Public-Key Cryptography through the Lens of Monoid Actions

Hart Montgomery [*] Sikhar Patranabis [†]

Abstract

We show that key exchange and two-party computation are exactly equivalent to monoid actions with certain structural and hardness properties. To the best of our knowledge, this is the first "natural" characterization of the mathematical structure inherent to any key exchange or two-party computation protocol, and the first explicit proof of the *necessity* of mathematical structure for public-key cryptography. We then utilize these characterizations to show a new black-box separation result, while also achieving a simpler and more general version of an existing black-box separation result. Concretely, we obtain the following results:

TWO-PARTY KEY EXCHANGE. We show that that *any* two-party noninteractive key exchange protocol is equivalent to the existence of an *abelian monoid* equipped with a natural hardness property, namely (distributional) *unpredictability*. More generally, we show that *any* k-round (two-party) key exchange protocol is essentially equivalent to the existence of a (distributional) unpredictable monoid with certain commutator-like properties. We then use a generic version of this primitive to show a simpler and more general version of Rudich's (Crypto '91) black-box separation of k-round and $(k+1)$-round key exchange.

TWO-PARTY COMPUTATION. We show that *any* maliciously secure two-party computation protocol is also equivalent to a monoid action with commutator-like properties and certain hardness guarantees. We then use a generic version of this primitive to show a black-box separation between k-round *semi-honest secure* two-party computation and $(k+1)$-round *maliciously secure* two-party computation. This yields the first black-box separation (to our knowledge) between k-round and $(k+1)$-round maliciously secure two-party computation protocols.

We believe that modeling cryptographic primitives as mathematical objects (and our approach of using such modeling for black-box separations) may have many other potential applications and uses in understanding what sort of assumptions and mathematical structure are necessary for certain cryptoprimitives.

[*] Linux Foundation. Email: hmontgomery@linuxfoundation.org

[†] IBM Research India. Email: sikhar.patranabis@ibm.com

Contents

1 Introduction

An important question in the theory of cryptography is also one of the simplest to state: what implies public-key cryptography? In particular, the idea of separating public-key cryptography from symmetric-key cryptography using mathematical structure has been around for quite some time: Barak mentions this in "The Complexity of Public-Key Cryptography" [Bar17]. As he puts it, "... it seems that you can't throw a rock without hitting a one-way function" but public-key cryptography is somehow "special." Barak implicitly argues that there is some mathematical structure inherent in public-key cryptography: "One way to phrase the question we are asking is to understand what type of structure is needed for public-key cryptography." However, formalizing this has proven to be difficult.

A number of works have shown connections between particular mathematical structures and cryptography. Hohenberger showed that pseudo-free groups had numerous cryptographic applications [Hoh03] which led to several follow-up works [Riv04, CFW11]. Other works [JQSY19, ADMP20] focused on building cryptography from "hard" group actions, and some papers focusing on Braid group cryptography have had interesting discussions on mathematical structure and cryptography [Gar08, AJJ12].

Characterizing Cryptoprimitives by Structure. There has been a line of work [AMPR19, AMP19, BKLS24] focused more directly on the characterization of cryptographic primitives by mathematical structure: roughly speaking, the authors of these papers show that certain primitives in the world of Minicrypt [Imp95] (i.e., one-way functions, pseudorandom generators, weak unpredictable functions, and weak pseudorandom functions) that are *homomorphic* between the input space (or the key space if it exists) and the output space directly imply the existence of many cryptographic primitives. However, these works are purely constructive and not very useful for separations: they show that simple primitives endowed with extra structure can be used to build powerful cryptographic primitives. In this paper, we ask the following question: is there a characterization of a cryptographic primitive in terms of a simply structured mathematical object that is *exactly* equivalent to the primitive?

Black-Box Separations. Another fundamental question in cryptography is to understand the power of a cryptographic primitive in terms of what other primitives are (im)possible to build from it in a black-box way. Understanding these implications lets us design new primitives, figure out attacks, and understand cryptographic primitives better in a complexity-theoretic sense.

Some of the oldest and most famous black-box separation results are about key exchange: in perhaps the most well-known work on black-box separations [IR89], Impagliazzo and Rudich showed how to separate key exchange (of any number of rounds) from one-way functions. In a follow-up work, Barak and Mahmoody [BM09] improved the result of Impagliazzo and Rudich, proving a "query-optimal" attack that nicely matched the known positive result: the famous Merkle puzzles [Mer78]. In addition, Rudich [Rud92] showed how to black-box separate k-round key exchange from $(k+1)$-round key exchange for any k. More recently, separations have helped us better understand things like MPC round complexity [ABG$^+$20] and indistinguishability obfuscation [GMM17a, GMM17b]. We refer to [Fis12] for a comprehensive survey of the enormous literature on black-box reductions and separations in cryptography, and present a more detailed treatment of related work in Section 1.3.

Relativizing Reductions. A well-studied approach to establishing black-box separations is to prove the impossibility of a *relativizing reduction* [RTV04] between certain primitives. In these sorts of separations, which aim to separate a "stronger" primitive from a "weaker" primitive, typically some oracle $\mathcal{O}$ or set of oracles with certain structure are assumed to exist. Generally speaking the structure of the oracle mimics the functionality of the "weaker" primitive in the separation result. It is then shown that, given $\mathcal{O}$ and some very powerful oracle (e.g. an NP-oracle), it is impossible to build the "stronger" primitive. The powerful NP-oracle (or sometimes an even more powerful oracle, like a PSPACE-oracle) serves to ensure that no hardness assumptions can be used other than what is inherent to the oracle $\mathcal{O}$. In these reductions, the oracle $\mathcal{O}$ can sometimes be stronger (or at least not necessarily equivalent to) the "weaker" primitive involved in the black-box separation. This extra slack has the potential to make black-box separation results much trickier since $\mathcal{O}$ may be "in between" the strong and weak primitives in terms of its power.

Separations using Mathematical Structure. In this work, we investigate the possibility of defining oracles that are *exactly* equivalent to generic versions of primitives that we want to separate. Such an approach potentially allows us to construct new or simplified separation results since oracles that are exactly equivalent to generic versions of primitives might be easier to separate than those that are not exactly equivalent. Moreover, characterizing cryptographic primitives in a way that leads to easily defined oracles may allow for interesting observations on what is necessary for building the primitives themselves in terms of both mathematical structure and hardness, which could be of independent interest, particularly for mathematicians looking to find new hardness assumptions and objects of cryptographic interest.

This motivates us to ask the following: can we characterize important cryptoprimitives *exactly* by their mathematical structure? And does characterizing cryptographic primitives in terms of the algebraic structure inherent to them enable new black-box separation results (or alternatively, enable simpler and more general versions of existing black-box separation results)?

1.1 Our Contributions

In this paper, we answer the above question in the affirmative. We present novel characterizations of common cryptographic primitives in terms of the mathematical structure that is inherent to such primitives. We additionally show that such characterization offers new possibilities for black-box separation results involving such primitives.

Concretely, we focus on two very popular and well-studied cryptographic primitives, namely two-party key exchange (abbreviated henceforth as KE) and two-party computation (abbreviated henceforth as 2-PC). We show that KE and 2-PC are exactly equivalent to monoid actions[1] with certain structural and hardness properties. To the best of our knowledge, this is the first "natural" characterization of the mathematical structure inherent to any KE or 2-PC protocol, and the first explicit proof of the *necessity* of mathematical structure for public-key cryptography. We then utilize these characterizations to show a new black-box separation result, while also achieving a more general version of an existing black-box separation result.

Structure of Key Exchange. Recently, *group actions* have become a popular concept in cryptography, as they have been used to model elliptic curve isogenies [BY91, Cou06, ADMP20]. Informally speaking, a group action is a tuple of a group $\mathbb{G}$, a set X, and an operation $\star$ where the identity (there is some identity element $e \in \mathbb{G}$ such that, for all $x \in X$, $e \star x = x$) and composability (for all $g, h \in \mathbb{G}$, $g \star (h \star x) = gh \star x$) axioms hold.

As shown in [ADMP20], we can define similar assumptions with group actions as is done with groups: for instance, informally speaking, the group action CDH problem (GA-CDH) is, for randomly sampled $g, h \in \mathbb{G}$ and $x \in X$, given $x, g \star x$, and $h \star x$, output $gh \star x$. Analogous to how can build key exchange from

[1] For unfamiliar readers, we explain and formally define these in our preliminaries and technical overview.

abelian groups where the CDH assumption holds, we can build key exchange from *abelian* group actions where the GA-CDH assumption holds: given a public set element x, Alice samples $g \in \mathbb{G}$ and sends $g \star x$ to Bob, Bob samples $h \in \mathbb{G}$ and sends $h \star x$ to Alice, and both Alice and Bob compute $gh \star x$. At a high level[1], this is how CSIDH [CLM+18] (and certain other families of isogeny-based assumptions) can be used to build key exchange.

Recall that a *monoid* is just a group where the property that elements have unique inverses is not required to hold. Similarly, we can define a *monoid action* as just the same thing as a group action except we use a monoid instead of a group. In this work, we show that *any* two-party non-interactive KE protocol is equivalent to the existence of an *abelian monoid action* equipped with a natural hardness property that is quite similar to a CDH assumption on monoid actions, namely (distributional) *unpredictability*. Our result is captured by the following (informal) theorem (see Theorems 2.5 and 2.4 for a more formal exposition):

Theorem 1.1 (Informal). *Any two-party non-interactive KE protocol is equivalent to a "hard" abelian monoid action, where the hardness assumption is distributional unpredictability.*

We generalize this result to show that *any* two-party k-round KE protocol is equivalent to the existence of a (distributional) unpredictable monoid action with certain commutator-like properties. This gives us what we consider to be the first "natural" characterization of KE with respect to mathematical structure. While it has long been folklore knowledge that some kind of structure is necessary for public-key cryptography, we believe that this is the first formalization of this idea. Moreover, it is only slightly weaker than a common abstraction–group actions–used to build popular key exchange protocols today.

We also emphasize that our result handles "noisy" key exchange protocols like those from LWE, and we explain this in detail later in the paper.

Revisiting KE Separation by Rounds. We consider a mathematically structured oracle representing a generic version of the monoid action above (which is, in turn, equivalent to a generic version of KE). We show how to use this oracle in order to black-box separate k-round key exchange and $(k+1)$-round key exchange for any polynomial k. This enables us to build a tighter, more rigorously formal, and more general version of the KE separation result due to Rudich [Rud92]. Our proof follows from applying our ideas on mathematically structured oracles to the proof frameworks used in [BM09].

Structure of Two-Party Computation. We show that *any* maliciously secure 2-PC protocol is also equivalent to a monoid action with certain commutator-like properties and certain hardness guarantees. The "hard" monoid action used in this characterization of 2-PC is (slightly) more complicated than the one we showed was equivalent to KE. Intuitively, rather than just using "random" monoid elements as we did above with key exchange, we can encode each player's secret information as well as the computation to be performed in the monoid elements themselves; the monoid action itself can be just to incorporate (but selectively hide) this information.

Thus, the main differences with the structural characterization of KE outlined above come from the facts that: (a) any 2-PC protocol must allow evaluating deterministic functions on the parties' inputs (a KE protocol, on the other hand, only outputs a random key to the parties involved), and (b) 2-PC has a very different notion of security as compared to KE. Consequently, the monoid action used in our characterization of 2-PC requires different structural and hardness properties as compared to its counterpart used in the characterization of KE. Our result is captured by the following (informal) theorem (see Theorem 4.4 for a more formal exposition):

[1] Due to [PR23], we can now consider CSIDH to be an *effective* group action.

Theorem 1.2 (Informal). *Any maliciously secure* 2*-PC protocol is equivalent to a monoid action satisfying certain commutator-like properties and certain (simulation-based) hardness guarantees.*

New Malicious 2**-PC Separation by Rounds.** As in the case of KE, we again consider a mathematically structured oracle representing a generic version of the monoid action above (which is, in turn, equivalent to a generic version of maliciously secure 2-PC with abort security). We show how to use this oracle in order to establish that is impossible to construct (in a black-box manner) a k-round *semi-honest secure* 2-PC protocol from *any* $(k+1)$-round *maliciously secure* 2-PC protocol. This yields the first black-box separation (to our knowledge) between k-round and $(k+1)$-round maliciously secure 2-PC protocols. Our result is captured by the following (informal) theorem (see Theorem 4.12 for a more formal exposition):

Theorem 1.3 (Informal). *For any* $k = \text{poly}(\lambda)$ *(where* λ *is the security parameter), there does not exist a black-box construction of* k*-round* semi-honest secure 2*-PC protocol from any* $(k+1)$*-round maliciously secure* 2*-PC protocol.*

Comparison with Known Results. We place our black-box separation result in the context of known black-box reductions and separations in the 2-PC literature. We begin by noting that [GKM$^+$00] showed a black-box separation between k-round and $(k+1)$-round (maliciously secure) oblivious transfer (OT). Coupled with the seminal result of Yao [Yao86] proving the (black-box) equivalence of k-round OT and k-round 2-PC for any $k \geq 2$ in the setting of *semi-honest* corruptions, this immediately yields a black-box separation of k-round 2-PC from $(k+1)$-round 2-PC for any $k \geq 2$ in the same setting. However, this does not yield a black-box separation result in the setting of malicious corruptions, which is our focus. In fact, extending the known black-box separation results to the setting of general maliciously secure 2-PC seems technically challenging, as outlined below.

Observe that an analogue of Yao's result (i.e., a round-preserving black-box reduction of k-round 2-PC to k-round OT for any $k \geq 2$) in the setting of malicious corruptions would immediately imply our result. However, to the best of our knowledge, such a reduction is only known for $k = 2$ rounds [IKO$^+$11, IKSS22], and it is not immediately clear how one might generalize these results to $k > 2$ rounds.

Concretely, suppose that there was a round-preserving black-box reduction of k-round 2-PC to k-round OT for *all* $k \geq 2$. Coupled with the black-box separation between k-round and $(k+1)$-round (maliciously secure) OT from [GKM$^+$00], this would immediately yield a black-box separation between maliciously secure k-round and $(k+1)$ 2-PC for all $k \geq 2$ (and thus imply our separation result). However, to the best of our knowledge, such a general round-preserving black-box reduction is not known for malicious corruptions.

Alternatively, suppose that there was a round-preserving black-box reduction of k-round 2-PC to k-round OT for *some* $k > 2$ (this is a seemingly weaker assumption as compared to the one for the first observation). Coupled with the black-box separation between k-round and $(k+1)$-round (maliciously secure) OT from [GKM$^+$00] and the known round-preserving black-box reduction of 2-round 2-PC to 2-round OT from [IKO$^+$11, IKSS22], this would immediately yield a black-box separation between maliciously secure 2-round and k-round 2-PC (but not k'-round 2-PC for some $k' \neq k$, thus implying a weaker version of our separation result). Unfortunately, such a round-preserving reduction is, in fact, not known in the malicious corruption setting for *any* $k > 2$. To summarize, our result is the *first* black-box separation of *maliciously secure* 2-PC by rounds and is, to the best of our knowledge, not subsumed by known black-box reductions and separations in the 2-PC literature.

An Observation on (Noisy) Multiparty NIKE. A natural question to ask is whether our approach to black-box separations using structural characterization extends to other similar cryptographic primitives, such as multiparty noninteractive key exchange (NIKE). More concretely, one could ask if there exists a structural characterization of k-party NIKE that would make it easy to black-box separate NIKE protocols by number of parties (more precisely, show a black-box separation between $(k + 1)$-party NIKE and k-party NIKE).

We give evidence that such a characterization is likely to require very different techniques (at least generally for all $k \geq 2$). In particular, we show that (for large enough k), a k-party NIKE protocol black-box implies a slightly weaker variant of a $(k + 1)$-party NIKE protocol. We call a this weaker variant a $(k + 1)$-party "2-noisy" NIKE protocol. Informally speaking, we say that a NIKE protocol is "ℓ-noisy" (for $\ell > 1$) if, instead of outputting a single shared key to all parties, the protocol outputs a total of ℓ candidate keys to each party with the following properties: (a) one of the ℓ keys received by each party is guaranteed to be shared by all parties, and (b) a passive eavesdropping (computationally bounded) adversary cannot predict (with non-negligible property) any of the ℓ candidate keys received by each party. For many practical applications (such as encryption), an ℓ-NIKE protocol in conjunction with a random oracle offers the same functionality as a regular NIKE protocol, albeit less efficiently. For example, in the case of encryption, the players could derive ℓ uncorrelated encryption keys by invoking the random oracle on the ℓ keys received from the ℓ-NIKE protocol, and then encrypt each message under each of the derived keys (one of which is guaranteed to be shared by all parties).

We show that, for large enough k, a k-party (regular) NIKE protocol implies (in a black-box manner) a $(k + 1)$-party 2-noisy NIKE protocol. While 2-noisy NIKE does not exactly meet the definition of regular NIKE and thus, our construction does not necessarily rule out the black-box separation of $(k + 1)$-party NIKE and k-party NIKE, it does offer strong evidence that such a separation will require very different techniques. As we discuss in Section 2.4 and Section 5, our observation rules out the possibility of using our black-box separation techniques, and more generally, the separation frameworks that we build upon [IR89, Rud92, BM09], to black-box separate NIKE by number of parties.

1.2 Implications of our Results

We show in this paper that structural characterizations of cryptoprimitives can be useful for black-box separation results, and we believe the separations that we have shown in this work only scratch the surface of what might be possible with these techniques. However, we also believe that characterizing cryptoprimitives *explicitly* by their structure could have extremely useful applications (even beyond black-box separations).

One of the most interesting and relevant areas of cryptographic research today is post-quantum cryptography. Many people today consider it worrisome that so many of our post-quantum cryptosystems are based on (essentially) a single hard problem: finding short vectors in lattices. In fact, NIST [oST22] said the following in their call for additional digital signatures: "NIST is primarily interested in additional general-purpose signature schemes that are not based on structured lattices." The text continues, later, "NIST is open to receiving additional submissions based on structured lattices, but is intent on diversifying the post-quantum signature standards." We consider it notable that an organization like NIST is willing to call out the lack of post-quantum assumption diversity so directly.

But from where do new cryptographic assumptions come? Historically, the most trusted assumptions have come from the world of mathematics[DH76, RSA78] [Ajt96, Cou06] and are usually based upon problems that mathematicians have been studying for decades. We hope that explicitly defining cryptoprimitives as mathematically structured objects will enable mathematicians and cryptographers to search more efficiently for potentially new (post-quantum) assumptions for public-key cryptography, since they will know exactly what kind of mathematical structure is required. Knowing that key exchange explicitly requires an abelian monoid action, for instance, substantially restricts the kind of mathematical assumptions that could be used to build key exchange, and thus may help researchers to narrow down the search for new key exchange constructions. A similar argument applies in the case of 2-PC (and, generally, any public-key cryptoprimitive).

Finally, we think that it is generally useful and interesting to study the relationship of cryptography and mathematical structure, and we can view our work here as continuing in the vein of [MP23], but with more concrete results.

1.3 Related Work

We present a full treatment of related work in this section.

Black-Box Separations. There has been a substantial amount of work done on cryptographic black-box separations and generic models. Some of the classical results include black-box separating collision-resistant hash functions (CRHFS) from general assumptions [Sim98], separating PKE and oblivious transfer (OT) [GKM+00], more general results on reducibility [RTV04], and generic cryptographic models [Sho97, Fis00]. We refer to [Fis12] for a survey of black-box reductions and separations in cryptography.

More recent results include an analysis of black box complexity of optimally-fair coin tossing [DLMM11, BHMO18, HMO18, MW20, MW21], black box separating CRHFs from hierarchical identity-based encryption (IBE) schemes [MM16], black-box separating the notions of PPAD Hardness and standard crypto assumptions [RSS17], showing limits on the usability of garbling for building PKE [GHMM18], and separating two-round secure computation from OT [ABG+20]. There has also been work done on separations between security notions [HK17] and compositions of reductions and the applicability to separations [CFM21]. Finally, quite a few recent works have focused on separations related to iO and advanced primitives [FS10, GKLM12, AS15, AS16, MMN+16, GMM17a, GMM17b, BDV17].

Non-Black-Box Constructions. There are many examples of non-black-box constructions of cryptographic primitives that manage to bypass black-box separation results. A notable example is Beaver's two-round OT extension protocol [Bea96], which bypasses a subsequently proposed black-box separation result for two-round OT extension in [GMMM18]. Similarly, [BOV03] presents a non-black box construction of non-interactive commitments from one-way functions, which bypasses a corresponding black-box separation result in [MP12]. Additionally, non-black-box constructions of functional encryption (FE) from indistinguishability obfuscation (iO) [BV15, AJ15] circumvent the result of [GMM17b] showing that it is impossible to build FE from iO in a non-black box manner.

In a breakthrough result, Döttling and Garg showed a non-black construction of (hierarchical) IBE from pairing-free CDH-hard groups [DG17a, DG17b], thereby bypassing known black-box separations between IBE and DDH-hard groups [BPR+08, PRV12]. In another work [KNT18], the known black-box separation between public-key FE and secret-key FE [AS15] was bypassed via a non-black construction.

Implications of Black-Box Separations. Black-box separations are especially useful in the sense that they indicate the necessity of resorting to non-black techniques when trying to build certain primitives from other primitives. In particular, for certain advanced cryptographic primitives such ad FE and iO, there exist many instances of non-black-box constructions of these primitives from other primitives/assumptions from which they have been black-box separated. However, in the case of certain simpler primitives (e.g. one-way functions and PKE), classical box-box separation results seem less likely to be circumvented. For example, a non-black-box construction of PKE from one-way functions would bypass [IR89, BM09] and collapse Cryptomania and Minicrypt into the same world [Imp95].

Other Related Work. We finally mention some other related work. The (black-box) impossibility of blind signatures from one-way permutations was shown in [KSY11], which was nice example of a separation between primitives. There have been a number of interesting results on one-way functions and trapdoor functions, including [GMR01, MM11]. Black-box separations have also been extensively studied in the context of commitment schemes and MPC [HK05, IKLP06, HHRS07], as well as oblivious transfer (OT) [GKM$^+$00, Hai08]. Finally, we refer to fundamental work on lower bounds on signatures from symmetric primitives [BM07] and generic oracles and oracle classes [BI87].

1.4 Paper Outline

The rest of the paper proceeds as follows. Section 2 presents a more detailed overview of our techniques. Section 3 presents the formal details of our first result, namely separating (two-party) KE by rounds, and is sub-organized as follows. Section 3.1 proves that any KE protocol is equivalent to the existence of an abelian monoid action equipped with certain natural hardness properties. Sections 3.2 and 3.3 build upon this equivalence result to present a proof of black-box separation of k-round KE from $(k+1)$-round KE. Section 4 presents the formal details of our second result, namely separating maliciously secure 2-PC by rounds, and is sub-organized as follows. Section 4.1 proves that any 2-PC protocol satisfying malicious security is equivalent to the existence of abelian monoid action equipped with certain additional structure and hardness properties. Sections 4.2 and 4.3 again build upon this equivalence result to present a proof of black-box separation of semi-honest secure k-round 2-PC from $(k+1)$-round 2-PC satisfying malicious security, where both protocols support computing symmetric functionalities. Section 4.4 generalizes this black-box separation separation result to 2-PC protocols for asymmetric functionalities. Finally, Section 5 presents some observations on (noisy) multiparty NIKE.

2 Technical Overview

In this section, we provide a more detailed technical overview of the results in the paper. A core component of our work will be *monoids* and *monoid actions*. Informally speaking, a monoid is a weakening of a group in the sense that the requirement of unique inverses does not hold. Similarly, a monoid action is a weakening of a *group action* [BY91, Cou06, CLM$^+$18, CLM$^+$18] where the group is replaced by a monoid. We define these objects somewhat informally below. For a thorough treatment, please see Definitions 3.1 and 3.2 in Section 3.1.

Definition 2.1 (Monoid). A monoid is defined as a tuple $(M, \oplus)$ where M is a set and $\oplus : M \times M \to M$ is an operation with the following properties:

- **Closure**: for all $g_1, g_2 \in M$, we have $g_1 \oplus g_2 \in M$.
- **(Left) Identity**: there exists an element $e \in M$ such that for all $g \in M$, we have $e \oplus g = g$.
- **Associativity**: for all $g_1, g_2, g_3 \in M$, we have $(g_1 \oplus g_2) \oplus g_3 = g_1 \oplus (g_2 \oplus g_3)$.

Finally, a monoid $(M, \oplus)$ is said to be *commutative* (or equivalently, *abelian*) if for any pair of elements $g_1, g_2 \in M$, we have $g_1 \oplus g_2 = g_2 \oplus g_1$.

Groups vs. Monoids. Any monoid where each element $g_1 \in M$ has some *unique* inverse $g_2 \in M$ such that $g_1 g_2 = e$ is a *group*, of which we assume all readers of this paper are familiar. So monoids are only a slight weakening of groups! To think more concretely, consider some field F. For some n, the set of $n \times n$ matrices over F endowed with the operation of standard matrix multiplication forms a monoid. The set of $n \times n$ *invertible* matrices over F endowed with the operation of matrix multiplication forms a group. The line between groups and monoids is often very thin, but, looking ahead, it is important that we use monoids and not groups in our protocols so that we can model "compressing" when we perform repeated operations.

Definition 2.2 (Monoid Action). A monoid $(M, \oplus)$ (as defined above) is said to *act on* a set X if there exists a map $\star : M \times X \to X$ that satisfies the following two properties:

1. **Identity:** If e is the identity element of M, then for any $x \in X$, we have $e \star x = x$.
2. **Compatibility:** For any $g, h \in M$ and any $x \in X$, we have $(g \oplus h) \star x = g \star (h \star x)$.

We use the notation $(M, X, \star)$ to denote a monoid action. Furthermore, we say that a monoid action $(M, X, \star)$ is a *commutative monoid action* if the monoid M is itself commutative. Moreover, we emphasize that the set X itself may be extremely unstructured: for instance, it may not be possible to efficiently sample a random element of X, and it may not even be possible to test if something is a member of X. There may be inputs (represented as bit strings) to monoid action computation algorithms that "work" in some way that are *not* set elements; but we are not concerned (at least from the point of view of the definition) with what happens on malformed operations.[1] We note that [ADMP20] has a thorough discussion on these limitations of sets in the context of *group* actions; these discussions also apply to our treatment of monoid actions.

String Concatenation Monoids and Monoid Actions. In our constructions and separations, we will use a natural class of monoids, called *string concatenation monoids*. If we let S denote the set of bit strings of length a multiple of some integer ℓ with a "null string" which we denote ϵ, then we can define a monoid $(S, \cdot)$ where $\cdot$ denotes the string concatenation operation. It is straightforward to see that the usual properties of a monoid hold:

- **Identity** : ϵ is the identity element.
- **Associativity** : for $a, b, c \in S, \quad a \cdot (b \cdot c) = a||b||c = (a \cdot b) \cdot c.$
- **Closure** : If $a, b \in S,$ then $a \cdot b \in S$ since $a \cdot b$ will have length a multiple of ℓ.

Given a string concatenation monoid, we can also define a *string concatenation monoid action*, which is a tuple $(M, X, \star)$ consisting of a string concatenation monoid, a set, and the usual mapping.

Further Extensions for Our Proofs. While a monoid action itself is just a manifestation of mathematical structure, we can endow monoid actions with various cryptographic properties as well: for instance, a monoid action is *one-way* if, given randomly chosen set elements $x_1, x_2 \in X$, it is hard to find an element $m \in M$ such that $m \star x_1 = x_2$.

We will also utlize what we call call *k-commutator* string concatenation monoid actions. Suppose we consider a string concatenation monoid action, but we add the constraint that, for some $k > 0$, for all $a, b \in M$, and all $x \in X$, we have

$$(a \cdot b)^k \star x = (b \cdot a)^k \star x.$$

[1] When we consider "cryptographic" monoids and monoid actions, our security definitions will rule out adversaries learning anything useful from these sorts of malformed inputs.

We note that such a string concatenation monoid action is still a valid monoid action, since it does not violate any of the axioms of a monoid action. In fact, when $k = 1$, it satisfies the standard notion of an abelian monoid action.

Looking ahead, we will also introduce another very useful constraint: the ability to limit the number of monoid operations performed. We can add another rule to our monoid (and monoid action) that does this: if the string in our string concatenation monoid becomes a certain length, we immediately map it to some terminal element $\perp$. Note that this extra rule doesn't violate the identity, closure, or associativity properties of the monoid (but it would not work for a group since we would be eliminating unique inverses). This extra rule will let us restrict the viable computations in monoids and monoid actions, which will be very useful for both modeling key exchange and arguing separations.

2.1 Key Exchange Is Equivalent to a "Hard" Abelian Monoid Action

We start by proving that two-party non-interactive key exchange (KE) is equivalent to an *abelian monoid action* with *distributional unpredictability*. To our knowledge, this constitutes the first proof that KE inherently requires algebraic structure, and provides a natural characterization of KE in terms of algebraically structured primitives.

Building Key Exchange. Group actions [BY91, Cou06] have a long history in cryptography and have recently seen increased interest due to the fact that secure elliptic curve isogeny-based protocols can often be modeled as group actions [CLM$^+$18, ADMP20]. Popular isogeny-based KE protocols such as CSIDH [CLM$^+$18] can be thought of as instantiations of *abelian* group actions where the *group action computational Diffie-Hellman problem* (GA-CDH problem) holds.

In this section, we show that this structure–an abelian group action where the GA-CDH problem holds–is *almost* necessary for the existence of KE! We only need to weaken the group (action) to a monoid (action), which only relaxes the requirement of the existence of unique inverses in the group. The "distributional unpredictability" requirement of the monoid action could certainly be rephrased as "the monoid action–CDH problem is hard" except for the fact that the CDH problem typically assumes that a challenge element is sampled *uniformly at random*, which may not be possible for certain monoids (the existence of unique inverses is assumed for many sampling algorithms). Since CDH is, at its heart, a computational unpredictability problem, we bake an efficient sampling algorithm for the monoid elements into our core requirement for KE and thus arrive at "computational unpredictability".

In [ADMP20], the authors define a KE protocol based on a *group* action,[1] and we show in this paper how to extend this protocol from groups to (abelian) monoid actions. Our protocol works as follows:

- Setup : Output $\mathsf{pp} = x \leftarrow X$.
- $\mathsf{Gen}_A(\mathsf{pp})$: Set $r_A = m_A \leftarrow M$ and output $s_A = m_A \star x$.
- $\mathsf{Gen}_B(\mathsf{pp})$: Set $r_B = m_B \leftarrow M$ and output $s_B = m_B \star x$.
- $\mathsf{Combine}_A(r_A, s_B)$: Output $k_{AB} = m_A \star s_B$.
- $\mathsf{Combine}_B(r_B, s_A)$: Output $k_{BA} = m_B \star s_A$.

[1] Constructing KE from group actions was known before [ADMP20] (and dates to at least 1997 [Cou06]), but we choose to mimic their presentation here.

If we simply replaced the monoid M with a group, then we would immediately recover the key exchange protocol from [ADMP20]. The authors of [ADMP20] focus on *regular* group actions[1], so their protocol and assumptions can be quite simply stated. Informally speaking, they rely on the group action-CDH assumption (GA-CDH), which states that, for a group action $(G, X, \star)$, $g, h \leftarrow G$ and $x \leftarrow X$, given x, $g \star x$, and $h \star x$, it is hard to construct $gh \star x$.

As we alluded earlier, using monoids instead of groups introduces some complications around sampling elements. For instance, sampling uniformly from a monoid could be difficult (the leftover hash Lemma [ILL89] holds over groups but not necessarily all monoids), and, looking ahead a little bit, the distributions over the monoid induced by KE protocols might also use distributions that aren't uniform over the monoid. Hence, we now describe a new primitive that we call a distributional *unpredictable* monoid action. Informally speaking, this is a monoid action where the "monoid action CDH problem" holds, but we have to be a little bit careful in defining this due to the reasons stated in the previous paragraph. More concretely, we take an abelian monoid action as defined above and endow it with a certain hardness property that we call *distributional unpredictability*. We describe this property in more details below.

Let $(M, X, \star)$ be a monoid action such that the set X supports efficient representation, and such that the "action operation" $\star$ is efficiently computable. Also let $\mathcal{D}_{M,b}$ for $b \in \{0, 1\}$ and $\mathcal{D}_X$ denote distributions over (subsets of) M and X, respectively, such that one can *efficiently* sample a monoid element $g \leftarrow \mathcal{D}_{M,0}$, a monoid element $h \leftarrow \mathcal{D}_{M,1}$ and a set element $x \leftarrow \mathcal{D}_X$ as per the distributions $\mathcal{D}_{M,0}$, $\mathcal{D}_{M,1}$ and $\mathcal{D}_X$, respectively. We define the following experiment (parameterized by the distributions $\mathcal{D}_{M,0}$, $\mathcal{D}_{M,1}$, and $\mathcal{D}_X$) between a challenger and a probabilistic polynomial-time adversary $\mathcal{A}$:

Experiment $\mathsf{Expt}_{\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$:

1. The challenger samples a pair of group elements (g, h) as $g \leftarrow \mathcal{D}_{M,0}$ and $h \leftarrow \mathcal{D}_{M,1}$, and a set element $x \leftarrow \mathcal{D}_X$, and provides the tuple $(x, g \star x, h \star x)$ to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ responds with a set element $y \in X$.

We say that the adversary $\mathcal{A}$ wins the experiment if $y = (g \oplus h) \star x$.

Definition 2.3 (Distributional Unpredictable Monoid Action (Definition 3.4, restated)). A monoid action $(M, X, \star)$ with an efficiently computable action operation is said to satisfy distributional unpredictability with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$ and with respect to some security parameter λ if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the experiment $\mathsf{Expt}_{\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$ is negligible in the security parameter λ.

This "distributional" unpredictable monoid action definition can just be thought of as an extension of the definition of a *weak* unpredictable group action from [ADMP20]–essentially, as we have suggested before, a group action where the GA-CDH problem is hard–to monoids, with the added caveat that we are not necessarily sampling uniformly over the monoid. Since we are primarily focused in this section on abelian monoid actions, we will refer to a *distributional unpredictable commutative monoid action* as a *DUCMA*. We are now in position to show the equivalence between DUCMA and KE.

[1] A regular group action is a group action that is *free* and *transitive*. Informally there is a (not generally efficiently computable) isomorphism between the group and the set in a regular group action.

Theorem 2.4 (DUCMA → Two-Party NIKE). *A two-party NIKE protocol can be built in a black-box manner from a secure DUCMA as in Definition 2.3.*

Proof. Our construction is the KE protocol as described above, where the starting set element $x \in X$ is sampled from $\mathcal{D}_X$, and the players A and B use the distributions $\mathcal{D}_{M,0}$ and $\mathcal{D}_{M,1}$ for sampling their monoid elements, respectively. With this in mind, the proof is almost immediate. Correctness follows from the commutativity of the monoid action, and the security proof is also simple: any adversary that can break the KE protocol can be used to break the security of the DUCMA, since the KE is essentially a protocol version of the DUCMA security experiment. □

We offer a more formal version of this proof in Section 3.1. We now prove the reverse statement, which is substantially more involved.

Theorem 2.5 (Two-Party NIKE → DUCMA). *Any two-party NIKE protocol can be used to build a DUCMA satisfying Definition 2.3.*

Proof. To prove this theorem, we show how to construct a monoid action $(M, X, \star)$ that satisfies the definition for DUCMA (Definition 2.3) with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$. We assume the existence of a two-party NIKE protocol consisting of the following algorithms:

- $\mathsf{Gen}_A : PP \times R_A \to S_A$.
- $\mathsf{Gen}_B : PP \times R_B \to S_B$.
- $\mathsf{Combine}_A : PP \times R_A \times S_B \to K$.
- $\mathsf{Combine}_B : PP \times R_B \times S_A \to K$.

Constructing the Monoid. We begin by describing how to construct the monoid $(M, \oplus)$ underlying the monoid action $(M, X, \star)$. Recall that any two-party NIKE protocol in our definition is associated with a pair of sets R_A and R_B, denoting the set of secret states for parties A and B, respectively. We define the following auxiliary sets:

$$R_{A,B} = \{r_A || r_B : r_A \in R_A, r_B \in R_B\}, \quad R_{B,A} = \{r_B || r_A : r_B \in R_B, r_A \in R_A\}.$$

At this point, we define the set M in the monoid $(M, \oplus)$ as:

$$M = R_A \cup R_B \cup R_{A,B} \cup R_{B,A} \cup \{e_M, \bot_M\},$$

where e_M is a special "identity" element and $\bot_M$ is a special "terminal" element. Next, we define the associated monoid operation $\oplus$ as follows:

- For any $r_A \in R_A$ and any $r_B \in R_B$, define $r_A \oplus r_B = r_B \oplus r_A := r_A \| r_B$.
- For any $\alpha \in M$, define $e_M \oplus \alpha = \alpha \oplus e_M := \alpha$.
- For any $(\alpha, \beta) \in M \times M$ such that $\alpha, \beta \neq e_M$ and $(\alpha, \beta) \notin R_A \times R_B$ and $(\alpha, \beta) \notin R_B \times R_A$, define $\alpha \oplus \beta = \bot_M$.

Lemma 2.6. *$(M, \oplus)$ is a commutative monoid.*

Proof. Closure, associativity and commutativity are immediate by construction. Also, e_M serves as the identity element for M. □

Remark 2.7. Note that for simplicity of exposition, we assume here that the sets R_A and R_B support efficient representations. In case this is not true, we equivalently represent an element r_A (resp., r_B) sampled according to the distribution $\mathcal{D}_A$ (resp., $\mathcal{D}_B$) using the random coins input to the sampling algorithm (any element that cannot be sampled according to these distributions does not appear in the monoid M).

Constructing the Set. Next, we define the set X as follows:

$$X = (PP \cup \{\perp_X\}) \times (S_A \cup \{\perp_X\}) \times (S_B \cup \{\perp_X\}) \times (K \cup \{\perp_X\})$$

where PP denotes the set of possible public parameters for the two-party NIKE protocol, S_A and S_B denote the set of possible output shares for A and B, respectively, K denotes the set of possible final keys on which A and B could agree, and $\perp_X$ is a special "terminal" symbol.

At a high level, a set element captures the gradual evolution of the public transcript of messages exchanged at various stages of the protocol, as well as the final computation of the shared key. In particular:

- A set element of the form $(\mathsf{pp}, \perp_X, \perp_X, \perp_X)$ represents the transcript of messages from the point of view of either party A or party B before the start of the protocol.
- A set element of the form $(\mathsf{pp}, \perp_X, s_B, \perp_X)$ represents the transcript of "received" messages from the point of view of party A after the first round of protocol execution.
- A set element of the form $(\mathsf{pp}, s_A, \perp_X, \perp_X)$ represents the transcript of "received" messages from the point of view of party B after the first round of protocol execution.
- A set element of the form $(\mathsf{pp}, s_A, s_B, k_{AB})$ represents the transcript of messages and the final secret key after the completion of the protocol (from the point of view of both parties A and B).

Defining the Action. We define the action $\star : M \times X \to X$. We make use of the functions associated with any two-party NIKE protocol as defined above to define the action operation $\star : M \times X \to X$ as follows:

- For any $x = (x_0, x_1, x_2, x_3) \in X$, define

$$e_M \star (x_0, x_1, x_2, x_3) := (x_0, x_1, x_2, x_3).$$

- For any $r_A \in R_A$ and $\mathsf{pp} \in PP$, define

$$r_A \star (\mathsf{pp}, \perp_X, \perp_X, \perp_X) := (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \perp_X, \perp_X).$$

- For any $r_B \in R_B$ and $\mathsf{pp} \in PP$, define

$$r_B \star (\mathsf{pp}, \perp_X, \perp_X, \perp_X) := (\mathsf{pp}, \perp_X, \mathsf{Gen}_A(\mathsf{pp}, r_B), \perp_X).$$

- For any $r_A \in R_A$, any $\mathsf{pp} \in PP$, and any $s_B \in S_B$, define

$$r_A \star (\mathsf{pp}, \perp_X, s_B, \perp_X) := (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), s_B, \mathsf{Combine}_A(\mathsf{pp}, r_A, s_B)).$$

- For any $r_B \in R_B$, any $\mathsf{pp} \in PP$, and any $s_A \in S_A$, define

$$r_B \star (\mathsf{pp}, s_A, \bot_X, \bot_X) := (\mathsf{pp}, s_A, \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_B(\mathsf{pp}, r_B, s_A)).$$

- For any $r_A \in R_A$, any $r_B \in R_B$, and any $\mathsf{pp} \in PP$ define

$$(r_A \| r_b) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) :=$$
$$(\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_A(\mathsf{pp}, r_A, \mathsf{Gen}_B(\mathsf{pp}, r_B))).$$

- All other action operations output the "terminal" element $(\bot_X, \bot_X, \bot_X, \bot_X)$.

Lemma 2.8 (Lemma 3.16, restated). *The monoid action $(M, X, \star)$ satisfies identity and compatibility if the two-party NIKE protocol satisfies correctness.*

Proof. We defer the detailed proof to Section 3.1 (see proof of Lemma 3.16). □

Putting together Lemma 2.6 and Lemma 2.8, we have that the tuple $(M, X, \star)$ is indeed a commutative monoid action. Finally, it follows immediately from the security of the two-party NIKE protocol that the group action $(M, X, \star)$ satisfies distributional unpredictability with respect to the distributions $\mathcal{D}_{M,b}$ for $b \in \{0, 1\}$ and $\mathcal{D}_X$ defined as follows:

$$\mathcal{D}_{M,0} := \mathcal{D}_A, \quad \mathcal{D}_{M,1} := \mathcal{D}_B, \quad \mathcal{D}_X := \mathcal{D}_{\mathsf{pp}},$$

where $\mathcal{D}_A$, $\mathcal{D}_B$ and $\mathcal{D}_{\mathsf{pp}}$ are the efficiently sampleable distributions over the sets R_A, R_B and PP in our definition of two-party NIKE.

This establishes that the group action $(M, X, \star)$ indeed satisfies the definition for DUCMA (Definition 2.3), and completes the proof of Theorem 3.13. □

On the Structure of the Monoid Action. At a first glance, the monoid action in our KE may seem very unnatural: while it is commutative, it only has two "levels" before all action computations map to the terminal element. But this restriction is necessary, since KE does not allow us to "chain" computations in general; the output of the KE may not be able to be used in any more mathematically structured computations. Moreover, there are some KE protocols which also only have two "levels": for instance, SIDH [DJP14] essentially allows computing a commutative square and nothing else. So our "restriction to the square" in terms of "levels" of the monoid action is strictly necessary, even if most KE protocols are more structured.[1]

On the Structure of the Monoid. Another intuitive question is the following: why do we need two separate sets R_A and R_B in our monoid? We use this distinction explicitly in our definition of the monoid operations to capture the fact that parties A and B apply the action between their respective randomness monoid elements to the initial set element in *potentially different orders* and make *potentially different contributions*, and thus follow two different paths to arrive at the same secret key at the end of the protocol. We use string concatenation to model this, and the ordering of elements is extremely important to define the action correctly.

In many KE protocols, the users perform identical actions, and in this case we could have just used one set. But in some cases, the users perform different actions that may need different amounts of (or structure in) the randomness, and this is why we need the slightly more complicated monoid description.

[1] We note that even though SIDH has been very recently broken [CD22], we still believe that it is interesting to examine its structure.

Extension to Multi-Round KE and PKE. We emphasize that we can also extend our results to multi-round KE (and hence, public-key encryption). In particular, we can use similar (but more technically involved) techniques to show that any $(2k-1)$-round KE is equivalent to a monoid action with what we call k-commutativity properties (i.e., for all $g, h \in M$ and all $x \in X$, we have $(gh)^k \star x = (hg)^k \star x$) with a security notion similar to distributional unpredictability. The key observation here is that even multi-round, interactive KE implies a relatively strong mathematical structure. We state the overall result in the following informal theorem.

Theorem 2.9 (Informal). *$(2k-1)$-round KE is equivalent to a monoid action where the k-commutators of the monoid are equal under black-box reductions.*

We defer the formal definitions of these properties and the detailed proof of Theorem 2.9 to Section 3.1.

Quantum Hardness and Non-Perfectly Correct KE. We note that inverting general monoid actions and similar problems are thought to be quantum-hard [CI14] and encompass similar problems over group actions [ADMP20]. Our results described above can immediately handle isogeny-based KE protocols.

For KE protocols with non-perfect correctness (e.g., LWE-based KE protocols), we only need to modify our framework slightly: instead of monoid actions that are perfectly commutative (or k-commutative), we instead use monoid actions that commute with reasonably large probability. This turns out to be a relatively minor change that does not affect any of our results since none of the security properties of the monoid action are affected, so we can model these primitives as well in our framework. We discuss this at the end of Section 3.1.

Implications on Structure. Our results provide further evidence for the unlikelihood of building public-key cryptography (and more generally, Cryptomania) from primitives with little or no algebraic structure (e.g., combinatorial symmetric-key primitives in Minicrypt, such as AES). Our results also yield the first natural characterization of KE based on algebraic structure, encompassing even less-structured protocols like SIDH [DJP14][1], which are otherwise impossible to capture via traditional abstractions.

2.2 Separating KE by Round

We now provide an outline of how we can use the above structural characterization of (multi-round) KE to separate KE by rounds. Our work here can be thought of as a more general, rigorously formal, and tighter version of the separation shown in [Rud92], where Rudich black-box separated k-round KE from $(k+1)$-round KE. Our separation result for 2-PC will follow from tweaking these arguments appropriately to fit the corresponding requirements.

Differences from Rudich's Separation [Rud92]. In his ground-breaking work [Rud92], Rudich separates k and $(k+1)$-round KE by constructing an oracle that enables $k+1$-round KE and then shows that it is black-box impossible to build k-round KE from this oracle.[2] Rudich shows that 2-round protocols built using only his oracle for 3-round KE have something that he defines as *restrictive form*. He then shows that an eavesdropper can always break protocols in restrictive form with constant probability.

[1] Even though SIDH has been recently broken [CD22], we still find its structure interesting.

[2] More precisely, Rudich black-box separates 3-round KE from 2-round KE, and leaves the extension to arbitrary $(k+1)$-round and k-round KE to the reader.

On the other hand, we formally show that *any* $(2k+1)$-round KE protocol is equivalent to a $(k+1)$-*commutator string concatenation monoid action* (abbreviated as $(k+1)$-SCMA). We then show that it is impossible to build a $2k$-round KE protocol from a $(k+1)$-SCMA oracle in a black-box manner. We also extend our formalism to separate $2k$-round and $(2k-1)$-round KE protocols. As discussed subsequently, viewing a KE protocol as an SCMA oracle allows our separation proof to achieve certain desirable properties as compared to [Rud92].

As an example of a difference between our generic SCMA oracle and Rudich's oracle, we note that our oracle allows for arbitrary public parameters (in addition to the oracle), while Rudich's oracle does not, which is important if we want to naturally capture the general form of all KE protocols. At a first glance, this may not seem important: one could just incorporate a fixed set of public parameters into the oracles. However, it has recently been shown in the context of group-based cryptography that the distinction between fixed or instance-based public parameters can be extremely important [BMZ19]. In addition, Rudich's framework and terms mirror that of [IR89], while our work here utilizes the more modern analysis of [BM09]. As a consequence, our separation proof leads to tighter lower bounds as compared to those in [Rud92].

2.2.1 Background: The Barak-Mahmoody Proof [BM09]

In [BM09], Barak and Mahmoody show how to improve the seminal result of Impagliazzo and Rudich [IR89], proving that KE (of any round length) cannot be built in a black-box way from random oracles (and hence, from one-way functions). Barak and Mahmoody specifically optimize for the number of random oracle queries that an eavesdropper Eve has to make in order to find the secret key by Alice and Bob during a KE protocol built from a random oracle. Both [IR89] and [BM09] use similar techniques, but for the discussion below, we prefer to use to the more modern presentation and proof techniques of [BM09].

The core idea of both of these works is the following: suppose Alice and Bob want to perform a KE in the presence of an eavesdropper Eve, and all parties have access to a random oracle. In addition, assume that Eve has access to an **NP** oracle (an oracle that can find witnesses for all statements in **NP**). Informally speaking, this is assumed to ensure that Alice and Bob cannot use any hardness assumptions other than what is provided by the random oracle. All parties start with the description of the KE protocol and any setup parameters.

Initially, Alice, Bob, and Eve all share the same amount of information. Alice and Bob can make queries to the random oracle without Eve knowing, which can potentially give them information that Eve does not have. But what [IR89] and [BM09] show is that, informally, if Eve queries all of the random oracle queries that Alice and Bob are (individually) likely to make, then Eve will likely end up querying all of the queries that *both* Alice and Bob make, which they refer to as *intersection queries*.

Both sets of authors then show that if Eve queries all of the intersection queries that Alice and Bob make during the protocol, Eve can efficiently recover the final shared key of Alice and Bob with reasonable probability. Intuitively, this seems quite simple at first glance: if Eve has all of the random queries that both Alice and Bob made, then the only thing that Alice and Bob have that Eve does not is the output of different random oracle queries, which should be totally random (and thus seem uncorrelated).

Unfortunately, this is not quite true: for instance, Bob could choose to randomly query the random oracle on 0 or 1 and send the output to Alice. Alice could query 1 and check if it is equal to what Bob sent. If these two queries are not equal, Alice knows that Bob queried the random oracle on 0 (and not 1) in the previous round, and thus shares a bit with Bob even though they did not query the same value. However, both [IR89] and [BM09] present detailed analyses (albeit using different techniques) of why such kinds of queries are not likely to help Alice and Bob. The core intuition, though, is the same: if Eve queries all of the "most likely" queries from Alice and Bob individually, then Eve finds all of the intersection queries that Alice and Bob make, and can (with some extra work) find the final key with noticeable probability.

2.2.2 Our Techniques

We now explain how we can apply the core argument of [BM09] to our structural observations on commutative (or k-commutative) monoid actions in order to separate KE by round. Our proofs rely crucially on generic oracles based on commutative monoids (or k-commutator monoid actions). In particular, we will use *string concatenation monoids and monoid actions* because they are extremely simple. We note that our proofs in the previous subsection utilized string concatenation monoid actions, and so our results on the equivalence between KE and monoid actions also hold for these specific types of monoid actions.

A k-commutator String Concatenation Monoid Action Oracle. In order to argue lower bounds on KE, we use a "generic version" of a k-round KE protocol. To do this, we define what we call a *generic string concatenation monoid action (SCMA) oracle*. We note that this "generic oracle technique" is analogous to previous work [IR89, BM09] where a random oracle was used as a "generic version" of a one-way function and [Rud92] where a "structured" random oracle was used except for the fact that we directly incorporate mathematical structure into our oracles (in a way more reminiscent of a generic group proof [FKL18]).

Our k-SCMA oracle will essentially work as a modified random oracle. We define it here in a way that isn't necessarily possible to compute efficiently, but we will be able to simulate it in our proofs. Let M be a string concatenation monoid. The k-SCMA oracle simulates a map of the form **M** : $M \times X \to X$, where X is some "set". In other words, we let the k-SCMA oracle **M** $(\cdot, \cdot)$ take as input a string from the monoid M and a "set element" in X, represented by either a previous output from the oracle or a predefined value for a "base point" x_0 given in the description of the oracle (this is analogous to giving out some initial elements in a generic group algorithm). For now, suppose that we have a single base point x_0 and that every set element in the action is "reachable" from x_0. These will be assumptions implicit in our proofs.

To evaluate **M** on input string $s \in M$ and set element $x \in X$, we first compute (or simulate computing) the string s' such that $x = s' \star x_0$. We then check to make sure that $\tilde{s} = s||s'$ is not over any limit in terms of length (i.e., $2k\ell$) and if it is, we will map to $\perp$. Otherwise, in most cases we will compute the random oracle on $s||s'$ and return that as the output of the oracle. If $\tilde{s}$ forms a string that has a "commutator element"–i.e. $\tilde{s}$ can be written as $(a||b)^k$ for some k, then we use some lexicographical metric on the bits of the element representation to choose one of the equivalent values of $\tilde{s}$ to use as the input to the random oracle.

In our actual proof, we will prove that, for any polynomially large k, a $2k$-round KE protocol cannot be built in a black-box way using only a $(k+1)$-SCMA oracle. Note that since a $(k+1)$-SCMA oracle implies a $(2k+1)$-round KE protocol, this gives us the logical result that a $2k$-round key exhange protocol cannot be built in a black box way from *any* $(2k+1)$-round KE protocol.

We finally note that the the construction of (multi-round) KE protocol from SCMA oracles is unconditionally (statistically) secure as it provably takes a super-polynomially large number of queries to break the corresponding hardness assumptions over an SCMA oracle. Therefore, this result should be interpreted along similar to a line of works on feasibility results based on idealized assumptions. For instance, one can easily show that certain idealized models such as generic group model (GGM) [Sho97] or algebraic group model (AGM) [FKL18] imply a KE protocol, and these results hold unconditionally.

Separating $2k$-round and $(2k+1)$-round KE. We now provide an abbreviated version of our result separating $2k$-round KE protocol from $(2k+1)$-round KE protocol. As alluded to earlier, we prove a tighter version of Rudich's result [Rud92], as captured by the following theorem.

Theorem 2.10 (KE Separation Theorem (Theorem 3.54, restated)). *Let Π be a $2k$-round KE protocol between Alice and Bob such that: (i) Alice and Bob make at most n_A and n_B queries, respectively, to a* *generic $(k+1)$-SCMA oracle, and use randomness r_A and r_B, respectively, and (ii) Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol. Then for every $0 < \delta < \rho$, there exists an attacker Eve that only has access to the public messages exchanged between Alice and Bob, makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-SCMA oracle, and produces an output s_E such that $\Pr[s_E = s_B] > \rho - \delta$.*

Proof Overview. As mentioned earlier, we essentially prove the impossibility of constructing a $2k$-round KE protocol where Alice and Bob only make queries to a $(k+1)$-SCMA oracle. At a high level, we rely on the observation that, except for the "commutative" queries – in other words, queries that evaluate an action of the form $(a \cdot b)^{k+1} \star x_0$ for some $a, b \in M$ – the $(k+1)$-SCMA oracle is no more powerful than any ordinary random oracle. Note that whenever Alice and Bob issue a query of the form (monoid element, set element), where the monoid element is a bit-string, there exists an equivalent execution of the KE protocol where they arrived at this query by "splitting up" their queries to the maximum extent possible (i.e., querying $\mathbf{M}(a, \mathbf{M}(b, x))$ instead of $\mathbf{M}(a \cdot b, x)$). In our proof, we will use a special form of a $2k$-round KE protocol where we "force" Alice and Bob to split up their queries in this manner. We argue that if there does not exist a secure KE protocol of this special form in $2k$ rounds, then there exists *no* secure KE protocol of $2k$ rounds. We briefly explain why this is the case below.

Given a secure $2k$-round KE protocol, we can create a new KE protocol of $2k$ rounds where Alice and Bob additionally make the "split-up" versions of their original queries to the $(k+1)$-SCMA oracle, and then simply ignore the outputs of these additional "split-up" queries. We then argue that this only incurs at most a polynomial blow-up in the number of queries as long as Alice and Bob make at most polynomially many queries, and also as long as k is at most polynomially large. Additionally, the view of the adversary Eve in this modified protocol is exactly the same as in the original protocol, as the distribution of the messages exchanged between Alice and Bob remains the same. This allows us to argue that any $2k$-round KE implies a $2k$-round KE protocol in the special form.

Observe that the contrapositive of this fact is that if there does not exist a secure KE protocol in $2k$ rounds where we split queries, then there does not exist *any* secure KE protocol of $2k$ rounds. This enables us to solely consider protocols in this special form. At a high level, we use the fact that Alice and Bob necessarily split their queries to the $(k+1)$-SCMA oracle to argue that if Alice and Bob issue a query that results in the same output, then there must exist a corresponding query made by *both* Alice *and* Bob where they used the same inputs. This reduces all queries where Alice and Bob received the same output to the "traditional" definition of intersection queries, and we can handle such queries using the [BM09] framework. We expand on our proof approach below.

Handling "Commutative" Queries. The crux of our proof lies in arguing that for a KE protocol in $2k$ rounds where Alice and Bob necessarily "split" their queries, a $(k+1)$-commutator oracle does not help Alice and Bob to ask "commutative" queries that Eve cannot also ask. To see why, recall how a $(k+1)$-SCMA $\mathbf{M} : M \times X \to X$ would be used to realize a $(2k+1)$-round KE.

- Given the base element x_0, Alice would sample some $a \in M$ and obtain $\mathbf{M}(a, x_0)$, while Bob would sample some $b \in M$ and obtain $\mathbf{M}(b, x_0)$. Alice and Bob would then exchange their first-round messages, where Alice sends $\mathbf{M}(a, x_0)$ to Bob and Bob sends $\mathbf{M}(b, x_0)$ to Alice.
- In the next round, Alice would obtain $\mathbf{M}(a \cdot b, x_0) = \mathbf{M}(a, \mathbf{M}(b, x_0))$, and Bob would obtain $\mathbf{M}(b \cdot a, x_0) = \mathbf{M}(b, \mathbf{M}(a, x_0))$. Alice and Bob would then exchange their second-round messages, where Alice sends $\mathbf{M}(a \cdot b, x_0)$ to Bob and Bob sends $\mathbf{M}(b \cdot a, x_0)$ to Alice.

Observe that by repeating this process for $(2k+1)$ rounds and asking a final query to the $(k+1)$-SCMA oracle, Alice and Bob would have obtained $\mathbf{M}((a \cdot b)^{k+1}, x_0) = \mathbf{M}((b \cdot a)^{k+1}, x_0)$, which they can use as the final secret key. Note that this computation requires the full $(2k+1)$ rounds[1].

Let us now examine what happens if Alice and Bob try to exploit the "commutative" property of the $(k+1)$-SCMA oracle in less than $(2k+1)$ rounds. They must generate some (effective) query of the form $\mathbf{M}((a \cdot b)^{k+1}, x_0)$ – which we call an *equivalence query* – with less than $(2k+1)$ rounds of communication. When "building up" to such an equivalence query that gives Alice and Bob the same final set element via two different query sequences in less than $(2k+1)$ rounds, Alice and Bob cannot only issue queries to the $(k+1)$-SCMA where the monoid element is either a or b like in the $(2k+1)$-round KE protocol outlined above. In particular, by the pigeonhole principle, at least one of Alice or Bob must compute a query involving both the elements a and b.

Suppose for the purpose of analysis that it is Alice that makes such a query to the $(k+1)$-SCMA oracle. In this case, we know that Alice must explicitly know both a and b. This is where we use the fact that our monoid is string concatenation: if it were some other monoid, it might be possible that Alice made a query of the form $\mathbf{M}(a \cdot b, x)$ or $\mathbf{M}(b \cdot a, x)$ without explicitly knowing a and b. Moreover, in this case Alice and Bob must both have queried some string including b *before* any equivalence queries were computed, and since Alice knows both a and b, she would be capable of computing *every possible* way to compute $\mathbf{M}((a \cdot b)^{k+1}, x_0)$ using the oracle $\mathbf{M}$. This is already strong evidence that the commutativity of the $(k+1)$-SCMA oracle is not useful (and thus we can rely on the core arguments of the [BM09] framework) – the main challenge lies in rigorously formalizing this intuition.

A Modification to Handle Equivalence Queries. Suppose we further specialize the form of the KE protocol as follows: if Alice (resp., Bob) computes a query of the form $\mathbf{M}(s, x)$ such that s is a string involving only two elements a and b in some alternating sequence (i.e. $ababa$), then she (resp., he) uses this as a "trigger" to additionally compute all possible equivalence queries that could lead to either $\mathbf{M}((a \cdot b)^{k+1}, x_0)$ or $\mathbf{M}((b \cdot a)^{k+1}, x_0)$. A simple counting argument shows that there are only $(4k+6)$ such equivalence queries. We refer to this special form of KE protocol where Alice and Bob necessarily ask these additional queries as *equivalence-complete*. As we did before with "splitting" queries, we can now use an analogous contrapositive argument to show that we *only* need to consider KE protocols that are equivalence-complete. We rigorously formalize this thought in Section 3.2 by stating and proving the following two lemmas, which essentially establish that *equivalence queries follow intersection queries*.

Lemma 2.11 (Lemma 3.59, restated). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob to a generic $(k+1)$-SCMA oracle $\mathbf{M} : M \times X \to X$ until round i of a $2k$-round KE protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) = ((s_A, x_A), (s_B, x_B)) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist monoid elements (i.e., strings) $s'_A, s'_B \in M$ such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \cdot s'_A = s_A \| s'_A = s_B \| s'_B = s_B \cdot s'_B.$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the KE protocol. Then there exists a set intersection queries $S \subset Q_A^{(i)} \cap Q_B^{(i)}$ such that if Eve asks each query in S, she asks a query that is equivalent to both the queries q_A and q_B.

[1] We note that if M is a countably infinite set, then a uniform distribution over M is not well-defined; in this case, we restrict to those distributions for which the set of all strings consisting of more than $2k$ elements has negligible density in the sample space.

Lemma 2.12 (Lemma 3.60, restated). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob to a generic $(k+1)$-SCMA oracle $\mathbf{M} : M \times X \to X$ till round i of a $2k$-round KE protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist monoid elements (i.e., strings) $a, b, s'_A, s'_B \in M$, such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \cdot s'_A = (a \cdot b)^{k+1}, \quad s_B \cdot s'_B = (b \cdot a)^{k+1},$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the KE protocol. Then we must have $q_A \in Q_A^{(i)} \cap Q_B^{(i)}$ or $q_B \in Q_A^{(i)} \cap Q_B^{(i)}$.

From the above lemmas, we know that if any equivalence query of the form $\mathbf{M}((a \cdot b)^{k+1}, x_0) = \mathbf{M}((b \cdot a)^{k+1}, x_0)$ is computed by Alice and Bob, one of them (assumed to be Alice) must have computed a query of the form $\mathbf{M}(s, x)$ such that s involves both a and b, and no other element. But this is precisely what we referred to as a "trigger" query, and by our definition of equivalence-complete KE, Alice necessarily computes all possible equivalence queries that could lead to either $\mathbf{M}((a \cdot b)^{k+1}, x_0)$ or $\mathbf{M}((b \cdot a)^{k+1}, x_0)$. Now, since Bob must query one of these equivalence queries to also arrive at $\mathbf{M}((b \cdot a)^{k+1}, x_0)$, there must exist an intersection query for Alice and Bob, and if Eve finds this query, she can also compute $\mathbf{M}((a \cdot b)^{k+1}, x_0) = \mathbf{M}((b \cdot a)^{k+1}, x_0)$. In other words, for any equivalence-complete KE protocol with $2k$ rounds, any equivalence query w.r.t. the $(k+1)$-SCMA oracle that can be computed within $2k$ rounds is also an intersection query. This again effectively reduces all equivalence queries that rely on the commutative property of the $(k+1)$-SCMA oracle to the "traditional" notion of intersection queries, and we can again handle such queries using the [BM09] framework.

The following auxiliary theorem captures this result, which we prove formally in Section 3.2 (the changes from Theorem 3.54 are highlighted in red).

Theorem 2.13 (Auxiliary KE Separation Theorem (Theorem 3.62, restated)). *Let Π be a k-round KE protocol between Alice and Bob such that: (i) Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-SCMA oracle, and use randomness r_A and r_B, respectively, (ii) the query pattern for Alice and Bob is equivalence-complete, and (iii) Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol. Then for every $0 < \delta < \rho$, there exists an attacker Eve that only has access to the public messages exchanged between Alice and Bob, makes at most $O(n_A n_B/\delta^2)$ queries to the generic $(k+1)$-SCMA oracle, and produces an output s_E such that $\Pr[s_E = s_B] > \rho - \delta$.*

Combining this with our earlier lemmas (Lemmas 2.11 and 2.12, showing that any general adversary can be modified into an adversary that only uses equivalence-complete query patterns) allows us to complete the proof of Theorem 2.10. We note here that while this is a high-level overview of the core idea of our proof. The actual analysis is significantly more involved; see Section 3.2 for more details.

Finishing the Separation Result. Finally, we point out that an argument very similar to the one outlined above allows us to separate $(2k-1)$-round KE from $2k$-round KE. In fact, the only change that we need to make is to slightly tweak the commutator-property of the $(k+1)$-SCMA oracle so that it allows secure $2k$-round KE protocol, but cannot be used to build a secure $(2k-1)$-round KE protocol. We elaborate more on this in Section 3.3.

We conclude the overview of our KE separation result by remarking that our characterization of KE in terms of generic string concatenation monoid action oracles is crucial to our separation result since it enables a simple yet rigorously formal framework for analyzing equivalence queries while fitting nicely with the proof techniques of [BM09]; such an analysis would be extremely cumbersome if one chose to work with generic KE protocols directly.

2.3 Analyzing 2-PC

In this subsection, we show how to model 2-PC as a special kind of monoid action. We then present an overview of our main novel separation result, namely, black-box separating 2-PC by rounds.

2.3.1 Modeling 2-PC as a "Hard" Monoid Action.

Earlier, we noted that (multi-round) KE was essentially just an interactive protocol where two parties sent messages back and forth and, at the end, managed to compute a shared secret key, which was a random value computed in two different ways. We modeled this as a monoid action where the only required structure was the k-commutativity.

If we think in such an abstract way, then 2-PC is not so different. Suppose we consider basic 2-PC: again, the parties send messages back and forth, and at the end compute the value of a function on their shared inputs. This process is very similar to KE except for the fact that we output a function evaluation instead of a random key (alternatively, we can think of KE as a special case of 2-PC where the "function" outputs a key based on the parties' randomness). It turns out we can also model this as an abelian monoid action, although with some extra properties. To the best of our knowledge, this is the first "natural" characterization of the mathematical structure inherent to any 2-PC protocol, and the first explicit proof of the *necessity* of mathematical structure for 2-PC.

The basic idea is as follows: to model KE as a monoid action, we used (essentially) random monoid elements. To model 2-PC, we utilize monoid elements that include randomness, an encoding of a player's inputs, and an encoding of the program to be computed. We note that the monoid elements themselves are never made public (both in KE and 2-PC), so we can use the action to effectively hide them. We explain this in more detail below.

The Monoid Structure. Let M be the string concatenation monoid containing the (sub)monoids A and B, where $A := I \times F \times R$, and $B = I \times F \times R'$, where all of the submonoids have string concatenation as their rule. We represent the parties' inputs with the set I, the function with the space F, and the parties' randomness *that they will use for the whole protocol* with the sets R and R'.

With this in mind, we can define a four-round 2-PC as an abelian monoid action $(M, X, \star)$ for the monoid M described above and the set X containing, at a minimum, all possible outputs of the functions represented by some $f \in F$ and any public parameters with the property that, for any $i_1, i_2 \in I$, $f \in F$, $r \in R$, and $r' \in R$ where $a := i_1 \times f \times r$ and $b := i_2 \times f \times r'$, and appropriately sampled $x \in X$, the following holds:

$$(a \cdot b \cdot a \cdot b) \star x = (b \cdot a \cdot b \cdot a) \star x = f\left(i_1, i_2\right).$$

This is only different than what we needed from KE in the structure of the monoid elements of M and the restrictions on the final output; the "broad picture" is entirely the same. So our natural thought was the following: can we use the ideas from our KE separation to also separate, say, 2-round 2-PC from 4-round 2-PC? While the structure of these two protocols is very similar, the security requirements are somewhat different. This presents the main technical hurdle in: (a) modeling 2-PC using structured primitives, and (b) using such a structural characterization to separate 2-PC by rounds.

Dealing with 2-PC Security Requirements. The trickiest part of extending our KE separations to cover 2-PC is how we handle the security requirements of 2-PC. We note that the structure of the monoid only helps us to define correctness; security must be described in terms of additional properties. In Section 4.1, we formally define all relevant notions of security using the standard simulation-based definitions; here we only have space to sketch out security properties.

Intuitively, our monoid action is secure in the setting of semi-honest corruptions if, given an adversary that can see a "full run" of the protocol from the perspective of one of the parties, the adversary cannot "learn anything" about the other party's input that cannot be guessed from the final output of the protocol. This turns out to be very similar to the KE notion of security, although the security property here would be closer to (some analogue of) the discrete log over the monoid action rather than a CDH-style assumption over the monoid action (which would be the rough approximation of security for KE).

In a malicious setting, a security definition is more complicated. Intuitively, we need that an adversary that controls Alice and has access to an "oracle" that takes as input a set element and outputs the action computation of Bob's secret monoid element on that set element cannot learn anything more about Bob's secret monoid element than is implied by the final output of the protocol. Technically speaking, we will actually prove security in a "malicious with abort" setting because it is impossible to have protocols secure against fully malicious adversaries in the standard model [Cle86].

A 2-PC Monoid Action Oracle. We can very naturally extend our generic k-SCMA oracle into a similarly defined oracle modified for 2-PC, which we call a k-SCMA$_{\text{2-PC}}$ oracle. Note that the only difference from a SCMA oracle is that we have adjusted the monoid elements (to incorporate the function) and the final output (to output the function evaluation rather than a key) as described above.

We emphasize that such a SCMA$_{\text{2-PC}}$ oracle implies *maliciously secure* (with abort) 2-PC: by definition, the intermediate computations in the oracle are random and thus reveal no extra information about parties' queries. Only an extremely lucky random guess would help an adversary. Moreover, it is possible to extract the "useful" portion of the corrupt party's input in a security game since, to get an output, it must send valid inputs to the SCMA$_{\text{2-PC}}$ oracle.

2.3.2 Extending the KE Separation to 2-PC.

With our SCMA$_{\text{2-PC}}$ oracle defined, it becomes quite straightforward to extend our KE separation to 2-PC. In fact, in our KE separation, we not only show that Eve can generate the final shared key of Alice and Bob, we also show that she can find the input monoid element of either Alice or Bob. Intuitively, this is because to make an intersection or equivalence query–and we show that Eve is likely to make all of these–Eve must know the monoid input element of either Alice or Bob. We can extend this argument almost immediately from the KE setting to the 2-PC setting.

On the other hand, if there are no equivalence queries, then, in order for the computation to be complete, one of the parties must have sent "enough" of their input for the other party to be able to evaluate the full computation on the SCMA$_{\text{2-PC}}$ oracle themselves, which also breaks the protocol. This is analogous to the KE case where Alice and Bob never use the full power of the SCMA oracle.

Note that this is the most for which we can hope: one (insecure) 2-PC protocol that is correct would be for Alice to send Bob her inputs "in the clear", and then Bob could do all of Alice's computations for her and then return Alice's output "in the clear" as well. In this case, it is impossible to learn anything useful about Bob's secrets. However, this is clearly enough to break all definitions of 2-PC security. Our main 2-PC separation result is summarized by the following theorem, which we prove rigorously in Section 4.2.

Theorem 2.14 (Main 2-PC Separation Theorem (Theorem 4.12, restated)). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob computing a function f such that: (i) Alice and Bob have inputs in_A and in_B, respectively, (ii) Alice and Bob make at most n_A and n_B queries to a generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, and use random tapes r_A and r_B, respectively, and (iii) Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol. Then for every $0 < \delta < \rho$, there exists* ***an attacker Eve*** *that corrupts Bob and makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, corresponding to which, with probability at least $\rho - \delta$,* ***there exists no probabilistic simulator*** *$\mathcal{S}$ that makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle such that*

$$\mathcal{S}^{\mathbf{M}(\cdot,\cdot)}\left(\mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B)\right) \overset{c}{\approx} V^{\Pi}_{\text{Eve}},$$

where V^{Π}_{Eve} denotes the view of Eve (consisting of the messages exchanged by Alice and Bob, Eve's queries to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, and Eve's own internal random coins, if any).

We remark that, as in our KE separation result, our characterization of 2-PC in terms of generic monoid action oracles is crucial to our 2-PC separation result since it enables porting the proof techniques of [BM09] and the additional analysis of equivalence queries from the setting of KE to 2-PC in a natural manner; such an analysis would be significantly more complicated if one chose to work with generic 2-PC protocols directly.

Proof Strategy. Our proof strategy is analogous to that for our KE separation proof, and involves showing the existence of an attacker Eve that recovers more information about the honest party Alice's input in_A than is revealed by the knowledge of Bob's input in_B and the function output $f(\mathsf{in}_A, \mathsf{in}_B)$. Consequently, an ideal-world simulator $\mathcal{S}$ can never simulate Eve's view since it can never obtain this additional information about Alice's input in_A (except with non-negligible probability) given only $(\mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B))$. Concretely, we prove the following auxiliary theorem, which in turn implies our main 2-PC separation theorem (Theorem 2.14).

Theorem 2.15 (Auxiliary 2-PC Separation Theorem (Theorem 4.13, restated)). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob such that: (i) Alice and Bob have inputs in_A and in_B, respectively, (ii) Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, and use random tapes r_A and r_B, respectively, and (iii) Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol. Then for every $0 < \delta < \rho$, there exists an attacker Eve that corrupts Bob and makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, such that Eve recovers, with probability at least $(\rho - \delta)$,* ***all*** *queries made by Alice to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle that are either identical to or are "equivalent" to the queries made by Bob to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle.*

To prove Theorem 2.15, we construct an attacker Eve that recovers (without loss of generality) the part of Alice's input that is relevant to the output of the function (more concretely, the secret monoid element representing Alice's input that is used in Alice's queries to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle). The proof is technically very similar to the proof of Theorem 2.9 used in our KE separation result, and is detailed in Section 4.2. Note that in our proposed attack strategy, Eve does not recover any parts of Alice's inputs that were not used by Alice to query the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle. In fact, it is impossible in general to recover any parts of Alice's input that are (potentially) irrelevant to the output, since Alice can (at least sometimes) start the interaction by first deleting the irrelevant parts of her input. We note, however, that recovering the part of Alice's input that is relevant to her output already constitutes an attack on the security of the 2-PC protocol since it allows Eve to learn potentially greater information than is leaked by the function output.

Theorem 2.15 implies Theorem 2.14. We briefly outline why Theorem 2.15 implies Theorem 2.14. Since the outputs of the generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle are (by definition) uniformly random and uncorrelated except for the commutator relation, Alice and Bob must issue certain intersection/equivalence queries to the oracle in order to arrive at the final output with high enough probability, and these queries must contain information about the parts of the inputs of Alice and Bob, respectively, that are relevant to the final function output. Now, Theorem 2.15 states that Eve recovers (with high enough probability) **all** of the intersection and equivalence queries made by Alice and Bob to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle based on their respective inputs. Thus, Eve manages to extract a query from the SCMA$_{\text{2-PC}}$ oracle that allows her to simulate the computation on Alice's input for any of Bob's inputs she likes, thus breaking 2-PC security.

Finally, we emphasize that, for perfect correctness to hold, Alice must use a query that (if it doesn't correspond to her correct input) must result in the exact same output for all possible inputs of Bob. Alice could, of course, use a query that corresponds to a different input than her "official" input in the protocol (as long as it gives the same results on all queries) in the process, but finding this again is clearly enough to break 2-PC security, as once again, Eve could simulate the computation on Alice's input for any of Bob's inputs.

Finishing the 2-PC Separation Result. Note that Theorem 2.14 black-box separates $2k$-round 2-PC from $(2k+1)$-round 2-PC for $k \geq 1$. A similar argument allows us to also black-box separate $(2k-1)$-round 2-PC from $2k$-round 2-PC for any $k \geq 2$, thus establishing the general black-box separation of k-round 2-PC from $(k+1)$-round 2-PC for any $k \geq 1$, which is the desired result. Analogous to the KE separation result, the only change that we need to make is to slightly tweak the commutator-property of the $(k+1)$-SCMA oracle so that it allows secure $2k$-round KE protocol, but cannot be used to build a secure $(2k-1)$-round KE protocol. We elaborate more on this in Section 4.3.

Extending to Asymmetric Functionalities. We can extend the above argument to handle 2-PC with asymmetric functionalities too. To do this, we just redefine F, so that, instead of representing a single function, it represents two functions $F := F_1 \times F_2$. We require that each party *puts its function first* in the monoid element that it use in the 2-PC protocol, so Alice's F will look like $F_{Alice} \times F_{Bob}$ and Bob's F will look like $F_{Bob} \times F_{Alice}$.

In our separation, we can just modify our oracle to check that the functions in each monoid element are properly formed. Then, at the end, we have the oracle only output the first listed function from the first monoid element applied to the initial set element. In this way, Alice and Bob only get the function output which they are supposed to receive. These changes to the oracle description require basically no changes to the overall proof and everything proceeds as before. We refer to Section 4.4 for a detailed treatment.

2.4 Observation on (Noisy) Multiparty NIKE

It is natural to ask if our approach to black-box separations using structural characterization extends to other similar cryptographic primitives, such as multiparty noninteractive key exchange (NIKE). We give evidence that such a characterization is likely to require very different techniques. In particular, we show that (for large enough k), any k-party NIKE protocol black-box implies a slightly weaker "noisy" variant of a $(k+1)$-party NIKE protocol, which we call "2-noisy" NIKE protocol. We informally describe this notion of multiparty NIKE below.

"Noisy" Multiparty NIKE. Informally speaking, we say that a k-party NIKE protocol is "ℓ-noisy" (for $\ell > 1$) if, instead of outputting a single shared key k to all parties, the protocol outputs a total of ℓ candidate keys $\mathsf{k}_1, \ldots, \mathsf{k}_\ell$ to each party with the following properties: (i) at least one of the ℓ keys received by each party is guaranteed to be shared by all parties, and hence can be treated as the shared secret key, and (ii) a passive eavesdropping (computationally bounded) adversary cannot predict (with non-negligible property) *any* of the ℓ candidate keys received by each party.

For many practical applications (such as encryption), an ℓ-NIKE protocol in conjunction with a random oracle offers the same functionality as a regular NIKE protocol, albeit inefficiently. For example, in the case of encryption, the players could derive ℓ uncorrelated encryption keys by invoking the random oracle on the ℓ keys received from the ℓ-NIKE protocol, and then encrypt each message under each of the derived keys (one of which is guaranteed to be shared by all parties).

Constructing $(k+1)$-party 2-noisy NIKE from k-party NIKE. We show a construction of $(k+1)$-party 2-noisy NIKE that uses a k-party (regular) NIKE protocol (in a black-box manner) and a (single-bit) randomness extractor Ext. We present an informal overview of the construction here. The full details appear in Section 5.

The construction proceeds as follows. The parties run $(k+1)$ instances of the k-party (regular) NIKE protocol *in parallel*, where the i^{th} NIKE instance does not involve party-i. Let k'_i be the shared key output by the i^{th} NIKE instance, and let $b_i = \mathsf{Ext}(k'_i) \in \{0,1\}$ be a bit extracted from this shared key. Observe that party-i obtains all bits b_j for $j \in [k+1]$ *except* the i^{th} bit b_i. It now derives two keys $k_{i,0}, k_{i,1} \in \{0,1\}^{k+1}$ as follows:

$$k_{i,0} = (b_1 \| \ldots \| b_{i-1} \| 0 \| b_{i+1} \| b_{k+1}) \,, \quad k_{i,1} = (b_1 \| \ldots \| b_{i-1} \| 1 \| b_{i+1} \| b_{k+1}) \,.$$

Finally, party-i outputs the pair of keys $(k_{i,0}, k_{i,1})$, one of which is guaranteed to be shared by all the parties.

Separating Multiparty NIKE by Number of Parties. Note that 2-noisy NIKE does not exactly meet the definition of regular NIKE and thus, our construction above does not necessarily rule out *any* black-box separation of NIKE by rounds. However, it does offer strong evidence that such a separation will somehow have to rely on the distinction between "noise-free" and "noisy" NIKE. In particular, it seems difficult to use our black-box separation techniques, as well as the black-box separation frameworks from [IR89, Rud92, BM09], to separate $(k+1)$-party NIKE and k-party NIKE. Note that all of these frameworks rely on the fact that an eavesdropping adversary Eve can make all of the queries to the oracle that the honest participants can make. Unfortunately, given a k-party NIKE oracle, any subset of k parties can issue a query to this oracle that Eve provably cannot make (in fact, our construction above crucially exploits this feature). Hence, we believe that a black-box separation of $(k+1)$-party NIKE and k-party NIKE would require entirely new techniques.

3 Analyzing Key Exchange

In this section, we present the formal details of our first technical contribution (and the starting point of our approach that revisits Rudich's black-box separation of KE by rounds), namely the proof that that two-party non-interactive KE (NIKE) is equivalent to an abelian monoid action with *distributional unpredictability*. We then describe formally how we can use the above structural characterization of (multi-round) KE to separate KE by rounds. As mentioned in the overview, our KE separation result can be thought of as a more general, simplified, and tighter version of the separation shown by Rudich in [Rud92].

3.1 Key Exchange and Commutative Monoid Action

In this section, we prove that *any* (two-party) non-interactive key exchange protocol is equivalent to the existence of an *abelian monoid action* equipped with a natural hardness property, namely (one-time) *unpredictability*. More generally, we show that *any* k-round key exchange protocol is essentially equivalent to the existence of a (one-time) unpredictable monoid action with certain commutator-like properties (we present a more precise formalization of this property subsequently).

To our knowledge, this is the first formal proof that public-key cryptography (and, more generally Cryptomania) requires explicit mathematical structure. It also appears to be the first "natural" characterization of the mathematical structure inherent to any key exchange protocol. We further note here that since public-key encryption is equivalent to two-round key exchange, our results also imply a characterization of the mathematical structure inherent to any public-key encryption scheme.

Monoids. We begin by recalling the standard algebraic definition of a monoid. At a high level, a monoid is a set equipped with an associative binary operation and an identity element. Another way of viewing a monoid is as a group where each element does not necessarily have a (unique) inverse. For the sake of completeness, we recall the formal definition below.

Definition 3.1 (Monoid). A monoid is defined as a tuple $(M, \oplus)$ where M is a set and $\oplus : M \times M \to M$ is an operation with the following properties:

- **Closure**: for all $g_1, g_2 \in M$, we have $g_1 \oplus g_2 \in M$.
- **(Left) Identity**: there exists an element $e \in M$ such that for all $g \in M$, we have $e \oplus g = g$.
- **Associativity**: for all $g_1, g_2, g_3 \in M$, we have $(g_1 \oplus g_2) \oplus g_3 = g_1 \oplus (g_2 \oplus g_3)$.

Finally, a monoid $(M, \oplus)$ is said to be *commutative* (or equivalently, *abelian*) if for any pair of elements $g_1, g_2 \in M$, we have $g_1 \oplus g_2 = g_2 \oplus g_1$.

Monoid Action. Having defined a monoid, we now define a *monoid action*. Informally, a monoid action is very similar to a group action (a mathematical object that has been previously studied in the context of cryptography [ADMP20]), except for the fact that the group is replaced by a monoid. We present the formal algebraic definition of a monoid action below.

Definition 3.2. (Monoid Action.) A monoid $(M, \oplus)$ (as defined above) is said to *act on* a set X if there exists a map $\star : M \times X \to X$ that satisfies the following two properties:

1. **Identity:** If e is the identity element of M, then for any $x \in X$, we have
$$e \star x = x.$$
2. **Compatibility:** For any $g, h \in M$ and any $x \in X$, we have
$$(g \oplus h) \star x = g \star (h \star x).$$

We use the notation $(M, X, \star)$ to denote a monoid action. Furthermore, we say that a monoid action $(M, X, \star)$ is a *commutative monoid action* if the monoid M is itself commutative.

Extending Monoid Actions to Monoids. It is a known (and to our knowledge, folklore) result that every monoid action can be extended to a monoid in a way that respects commutativity. This essentially implies that a (commutative) monoid action is not a fundamentally different algebraic/category-theoretic object as compared to a (commutative) monoid; they are, in fact, equivalent. We formalize this result below.

Lemma 3.3. *Any (commutative) monoid action $(M, X.\star)$ can be extended to a (commutative) monoid $\left(\widehat{M}, \widehat{\oplus}\right)$ in a structure-preserving manner.*

Proof. Let $(M, X, \star)$ be a monoid action where the monoid $(M, \oplus)$ acts on the set X. We first consider the case where $(M, \oplus)$ is non-commutative. In this case, the extended monoid $(\widehat{M}, \widehat{\oplus})$ is defined as follows:

- The set $\widehat{M}$ is defined as $\widehat{M} := M \cup X \cup \{\perp\}$, where $\perp$ is a special "terminal" element.
- The operation $\widehat{\oplus}$ is defined as follows:
  - For any $g, h \in M$, define $g \widehat{\oplus} h := g \oplus h$.
  - For any $g \in M$ and $x \in X$, define $g \widehat{\oplus} x := g \star x$.
  - For any $(\alpha, \beta) \in \widehat{M} \times \widehat{M}$ such that $(\alpha, \beta) \notin M \times M$ and $(\alpha, \beta) \notin M \times X$, define $\alpha \widehat{\oplus} \beta := \perp$.

It is straightforward to see that the tuple $(\widehat{M}, \widehat{\oplus})$ satisfies both closure and associativity. At a high level, this follows from the fact that any operation that is not semantically defined in the original group action maps to the terminal element $\perp$. Additionally, the (left) identity element e for the monoid M also serves as the (left) identity element for $\widehat{M}$. Hence, $(\widehat{M}, \widehat{\oplus})$ is a monoid, as desired.

We now consider the case where $(M, \oplus)$ is commutative. In this case, the extended monoid $(\widehat{M}, \widehat{\oplus})$ is defined in a commutativity-preserving manner as follows (the changes from the non-commutative case are highlighted in red):

- The set $\widehat{M}$ is again defined as $\widehat{M} := M \cup X \cup \{\perp\}$, where $\perp$ is a special "terminal" element.
- The operation $\widehat{\oplus}$ is now defined as follows:
  - For any $g, h \in M$, define $g \widehat{\oplus} h := g \oplus h$.
  - For any $g \in M$ and $x \in X$, define $g \widehat{\oplus} x := g \star x$ and $x \widehat{\oplus} g := g \star x$.
  - For any $(\alpha, \beta) \in \widehat{M} \times \widehat{M}$ such that $(\alpha, \beta) \notin M \times M$ and $(\alpha, \beta) \notin M \times X$ and $(\alpha, \beta) \notin X \times M$, define $\alpha \widehat{\oplus} \beta := \perp$.

It is again straightforward to see that the tuple $(\widehat{M}, \widehat{\oplus})$ satisfies closure. Additionally, the (left) identity element e for the monoid M still serves as the (left) identity element for $\widehat{M}$. So, it remains to argue associativity and commutativity.

To see that associativity is satisfied, observe that since $(M, \oplus)$ is commutative, for any $g, h \in M$ and $x \in X$, we have:

1. $x \widehat{\oplus} (g \widehat{\oplus} h) = (x \widehat{\oplus} g) \widehat{\oplus} h$, and

Experiment $\mathsf{Expt}_{\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$:

1. The challenger samples a pair of group elements (g, h) as $g \leftarrow \mathcal{D}_{M,0}$ and $h \leftarrow \mathcal{D}_{M,1}$, and a set element $x \leftarrow \mathcal{D}_X$, and provides the tuple $(x, g \star x, h \star x)$ to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ responds with a set element $y \in X$.

We say that the adversary $\mathcal{A}$ wins the experiment if $y = (g \oplus h) \star x$.

Figure 1: The Security Definition of a Distributional Unpredictable Monoid Action

2. $g \widehat{\oplus} (x \widehat{\oplus} h) = (g \widehat{\oplus} x) \widehat{\oplus} h$.

More concretely, we have

$$x \widehat{\oplus} (g \widehat{\oplus} h) = x \widehat{\oplus} (g \oplus h) = (g \oplus h) \star x = (h \oplus g) \star x = h \star (g \star x) = h \star (x \widehat{\oplus} g) = (x \widehat{\oplus} g) \widehat{\oplus} h.$$

$$g \widehat{\oplus} (x \widehat{\oplus} h) = g \widehat{\oplus} (h \star x) = g \star (h \star x) = h \star (g \star x) = h \star (g \widehat{\oplus} x) = (g \widehat{\oplus} x) \widehat{\oplus} h.$$

Any other operation that is not semantically defined in the original group action still maps to the terminal element $\perp$. Hence, $(\widehat{M}, \widehat{\oplus})$ satisfies associativity whenever $(M, \oplus)$ is both associative and commutative.

Finally, it is straightforward to see that $(\widehat{M}, \widehat{\oplus})$ is commutative whenever $(M, \oplus)$ is commutative. Hence, $(\widehat{M}, \widehat{\oplus})$ is a commutative monoid, as desired. □

3.1.1 Distributional Unpredictable Monoid Action

We now describe a new primitive that we call a distributional *unpredictable* monoid action. More concretely, we take a monoid action as defined above and endow it with a certain hardness property that we call distributional unpredictability. We describe this property in more details below.

Distributional Unpredictable Monoid Action. Let $(M, X, \star)$ be a monoid action such that the set X supports efficient representation, and such that the "action operation" $\star$ is efficiently computable. Also let $\mathcal{D}_{M,b}$ for $b \in \{0, 1\}$ and $\mathcal{D}_X$ denote distributions over (subsets of) M and X, respectively, such that one can *efficiently* sample a monoid element $g \leftarrow \mathcal{D}_{M,0}$, a monoid element $h \leftarrow \mathcal{D}_{M,1}$ and a set element $x \leftarrow \mathcal{D}_X$ as per the distributions $\mathcal{D}_{M,0}$,$\mathcal{D}_{M,1}$ and $\mathcal{D}_X$, respectively. We define the experiment $\mathsf{Expt}_{\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$ (parameterized by the distributions $\mathcal{D}_{M,0}$, $\mathcal{D}_{M,1}$, and $\mathcal{D}_X$) between a challenger and a probabilistic polynomial-time adversary $\mathcal{A}$ as in Figure 1.

Definition 3.4 (Distributional Unpredictable Monoid Action). A monoid action $(M, X, \star)$ with an efficiently computable action operation is said to satisfy distributional unpredictability with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$ and with respect to some security parameter λ if for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the experiment $\mathsf{Expt}_{\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$ is negligible in the security parameter λ.

Remark 3.5. For simplicity, we abstract out the details of the (efficient) sampling procedures that allow sampling a monoid element $g \leftarrow \mathcal{D}_{M,b}$ for $b \in \{0, 1\}$ and a set element $x \leftarrow \mathcal{D}_X$. We simply assume that these algorithms take as input the security parameter λ and some random coins r, and output elements as per the desired distributions.

Remark 3.6. Note that we do not necessarily require the distributions $\mathcal{D}_{M,b}$ for $b \in \{0,1\}$ and $\mathcal{D}_X$ to be the uniform distributions over M and X, respectively. This distinguishes our notion of distributional unpredictability from the more standard notion of *weak* unpredictability in the cryptographic literature, where these distributions would be necessarily uniform. Our definition can be viewed as a generalization of weak unpredictability in the context of monoid actions. We note that it is typically much easier to sample uniform elements in groups (where inverses exist) than in monoids.

Remark 3.7. Note that in the aforementioned definition, we do not assume that the monoid action $(M, X, \star)$ necessarily supports compact representations for elements in the monoid M. For example, in order to represent a monoid element g sampled according to the distribution $\mathcal{D}_{M,0}$, one could simply use the random coins input to the sampling algorithm as an equivalent compact representation for g (so long as the action computation is efficient using this alternative representation).

3.1.2 Two-Party Non-Interactive Key Exchange (NIKE)

We now formally define a two-party non-interactive key exchange (NIKE) protocol [BS20].

Definition 3.8 (Non-interactive Key Exchange (NIKE)). A NIKE protocol is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Setup}, \mathsf{A}_0, \mathsf{B}_0, \mathsf{A}_1, \mathsf{B}_1)$ defined as follows:

- Setup takes as input a security parameter λ and outputs the public parameters pp.
- A_0 takes as input the public parameters pp, and outputs a secret state r_A and a share s_A.
- B_0 takes as input the public parameters pp, and outputs a secret state r_B and a share s_B.
- A_1 takes as input $(\mathsf{pp}, r_A, s_A, s_B)$ to compute the "final key" k_{AB}.
- B_1 takes as input $(\mathsf{pp}, r_B, s_B, s_A)$ to compute the "final key" k_{BA}.

A NIKE protocol is essentially a single-round protocol between a pair of (non-uniform) probabilistic polynomial-time algorithms (informally referred to as "parties") $A = (\mathsf{A}_0, \mathsf{A}_1)$ and $B = (\mathsf{B}_0, \mathsf{B}_1)$, where the tuple

$$\tau = (\mathsf{pp}, s_A, s_B)$$

denotes represents the *public transcript* of messages exchanged between A and B.

Correctness. A NIKE protocol $(\mathsf{Setup}, \mathsf{A}_0, \mathsf{B}_0, \mathsf{A}_1, \mathsf{B}_1)$ is said to be correct if for any $\mathsf{pp} \leftarrow \mathsf{Setup}$, for any $(r_A, s_A) \leftarrow \mathsf{A}_0(\mathsf{pp})$ and any $(r_B, s_B) \leftarrow \mathsf{B}_0(\mathsf{pp})$, we have

$$k_{AB} = k_{BA},$$

where $k_{AB} = \mathsf{A}_1(\mathsf{pp}, r_A, s_A, s_B)$ and $k_{BA} = \mathsf{B}_1(\mathsf{pp}, r_B, s_B, s_A)$.

Security. A NIKE protocol $(\mathsf{Setup}, \mathsf{A}_0, \mathsf{B}_0, \mathsf{A}_1, \mathsf{B}_1)$ is said to be secure if for any $\mathsf{pp} \leftarrow \mathsf{Setup}$, for any $(s_A, s_A) \leftarrow \mathsf{A}_0(\mathsf{pp})$ and any $(s_B, s_B) \leftarrow \mathsf{B}_0(\mathsf{pp})$, and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

$$\Pr[\mathcal{A}(\mathsf{pp}, s_A, s_B) = k_{AB}] < \mathrm{negl}(\lambda),$$

where $k_{AB} = \mathsf{A}_1(\mathsf{pp}, r_A, s_A, s_B)$.

Representing NIKE as a Commutative Square. We now formulate a NIKE protocol as a *commutative square*, capturing the core property that two parties can compute the same secret key using two different sequences of computation. Let PP, R, S_A, S_B, R_A, R_B, and K denote *sets*. More specifically:

- We let PP denote the set of public parameters and R denote the set of possible random coins used by the setup algorithm to output some public parameters from the set PP.

- We also let S_A and S_B (resp., R_A and R_B) denote the set of possible output shares (resp., the set of possible secret states) for the parties A and B, respectively.

- Finally, we let K denote the set of possible final keys that the parties A and B could agree on at the end of the NIKE protocol.

Next, we define the following functions that map between these sets as below:

- $\mathsf{Setup} : 1^\lambda \times R \to PP$.

- $\mathsf{Gen}_A : PP \times R_A \to S_A$.

- $\mathsf{Gen}_B : PP \times R_B \to S_B$.

- $\mathsf{Combine}_A : PP \times R_A \times S_B \to K$.

- $\mathsf{Combine}_B : PP \times R_B \times S_A \to K$.

Finally, we impose the following correctness requirement on these functions: for any $\mathsf{pp} \in PP$, any $r_A \in R_A$ and any $r_B \in R_B$, we have

$$\mathsf{Combine}_A\left(\mathsf{pp}, r_A, \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right)\right) = \mathsf{Combine}_B\left(\mathsf{pp}, r_B, \mathsf{Gen}_A\left(\mathsf{pp}, r_A\right)\right).$$

Security. Let $\mathcal{D}_{\mathsf{pp}}, \mathcal{D}_A$ and $\mathcal{D}_B$ denote efficiently sampleable distributions over the sets PP, R_A, and R_B, respectively. Based on the above structural formulation, we say that a NIKE protocol is $(\mathcal{D}_{\mathsf{pp}}, \mathcal{D}_A, \mathcal{D}_B)$-secure if for any $\mathsf{pp} \leftarrow \mathcal{D}_{\mathsf{pp}}$, any $r_A \leftarrow \mathcal{D}_A$ and any $r_B \leftarrow \mathcal{D}_B$, and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

$$\Pr[\mathcal{A}(\mathsf{pp}, \mathsf{Gen}_A\left(\mathsf{pp}, r_A\right), \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right)) = \mathsf{Combine}_A\left(\mathsf{pp}, r_A, \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right)\right)] < \mathrm{negl}(\lambda).$$

Remark 3.9. For simplicity, we abstract out the details of the (efficient) sampling procedures that allow sampling as per the distributions $\mathcal{D}_{\mathsf{pp}}$, $\mathcal{D}_A$, and $\mathcal{D}_B$. We simply assume that these algorithms take as input the security parameter λ and some random coins r from the set R, and output elements as per the desired distributions.

Remark 3.10. One again, note that we do not necessarily require the distributions $\mathcal{D}_{\mathsf{pp}}$, $\mathcal{D}_A$, and $\mathcal{D}_B$ to be the uniform distributions over the sets PP, R_A, and R_B, respectively.

Remark 3.11. Note that in the aforementioned definition, we do not assume that the sets R_A and R_B necessarily support compact representations. For example, in order to represent an element r_A sampled according to the distribution $\mathcal{D}_A$, one could simply use the random coins input to the sampling algorithm as an equivalent compact representation for r_A (so long as all relevant Function computations are efficient using this alternative representation).

3.1.3 Equivalence of Distributional Unpredictable Commutative Monoid Action and NIKE

At this point, the astute reader might have already noticed the (almost) exact structural correspondence between a distributional unpredictable commutative monoid action and the commutative-square depicting a NIKE protocol. At a high level, one could simply use this "structural" correspondence to informally argue that these two primitives are, in fact, equivalent. However, formalizing this equivalence is more involved, as we show subsequently. In what follows, we use the acronym "DUCMA" to denote a distributional unpredictable commutative monoid action.

DUCMA implies NIKE. We first formally prove the easier direction, namely, DUCMA implies NIKE. More concretely, we state and prove the following theorem:

Theorem 3.12. *Any DUCMA satisfying Definition 3.4 implies a NIKE protocol.*

Proof. Let $(M, X, \star)$ be a DUCMA with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$ as per the structural formulation for DUCMA described earlier (Definition 3.4). We describe a construction of NIKE protocol satisfying the structural formulation for NIKE described earlier. Our protocol bears certain similarities with existing NIKE protocols based on cryptographic group actions (e.g. in [ADMP20]).

Set definitions: We define the following sets for the NIKE protocol:

- Define $PP := X$, where X denotes the set in the group action $(M, X, \star)$.

- Define the set of secret states for A and B as $R_A := M$ and $R_B := M$, respectively, where M denotes the monoid in the group action $(M, X, \star)$.

- Define the set of possible output shares for A and B as $S_A := X$ and $S_B := X$, respectively, where X is again the set in the group action $(M, X, \star)$.

- Define the set of possible final keys as $K := X$, where X is again the set in the group action $(M, X, \star)$.

Function definitions: We define the following functions for the NIKE protocol:

- $\mathsf{Setup} : 1^\lambda \times R \to PP$: Sample $x \leftarrow \mathcal{D}_X$ and output x.

- $\mathsf{Gen}_A : PP \times R_A \to S_A$: Sample $g_A \leftarrow \mathcal{D}_{M,0}$ using random coins r_A and output $s_A = g_A \star x$.

- $\mathsf{Gen}_B : PP \times R_B \to S_B$: Sample $g_B \leftarrow \mathcal{D}_{M,1}$ using random coins r_B and output $s_B = g_B \star x$.

- $\mathsf{Combine}_A : PP \times R_A \times S_B \to K$: Re-sample $g_A \leftarrow \mathcal{D}_{M,0}$ using random coins r_A and output the final key as $k_{AB} = g_A \star s_B$.

- $\mathsf{Combine}_B : PP \times R_B \times S_A \to K$: Re-sample $g_B \leftarrow \mathcal{D}_{M,1}$ using random coins r_B and output the final key as $k_{BA} = g_B \star s_A$.

Correctness and Security. Correctness and security of the NIKE protocol described above are immediate from the structural formulation for DUCMA described earlier (Definition 3.4). This completes the proof of Theorem 3.12. □

NIKE implies DUCMA. We now formally prove the more involved direction, namely, NIKE implies DUCMA. More concretely, we state and prove the following theorem:

Theorem 3.13. *Any NIKE protocol implies a DUCMA satisfying Definition 3.4.*

Proof. To prove this theorem, we show how to construct a monoid action $(M, X, \star)$ that satisfies the definition for DUCMA (Definition 3.4) with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$. We assume the existence of a NIKE protocol satisfying the structural formulation as outlined above, including all the relevant sets and functions.

Constructing the Monoid. We begin by describing how to construct the monoid $(M, \oplus)$ underlying the monoid action $(M, X, \star)$. Recall that in our structural formulation, any NIKE protocol is associated with a pair of sets R_A and R_B, denoting the set of secret states for parties A and B, respectively.

We define the following auxiliary sets:

$$R_{A,B} = \{r_A \| r_B : r_A \in R_A, r_B \in R_B\}, \quad R_{B,A} = \{r_B \| r_A : r_B \in R_B, r_A \in R_A\}.$$

At this point, we define the set M in the monoid $(M, \oplus)$ as:

$$M = R_A \cup R_B \cup R_{A,B} \cup R_{B,A} \cup \{e_M, \perp_M\},$$

where e_M is a special "identity" element and $\perp_M$ is a special "terminal" element. Next, we define the associated monoid operation $\oplus$ as follows:

- For any $r_A \in R_A$ and any $r_B \in R_B$, define

$$r_A \oplus r_B = r_B \oplus r_A := r_A \| r_B.$$

- For any $\alpha \in M$, define

$$e_M \oplus \alpha = \alpha \oplus e_M := \alpha.$$

- For any $(\alpha, \beta) \in M \times M$ such that $\alpha, \beta \neq e_M$ and $(\alpha, \beta) \notin R_A \times R_B$ and $(\alpha, \beta) \notin R_B \times R_A$, define

$$x \oplus y = \perp_M.$$

Lemma 3.14. $(M, \oplus)$ *is a commutative monoid.*

Proof. Closure, associativity and commutativity are immediate by construction. Also, e_M serves as the identity element for M. □

Remark 3.15. Note that for simplicity of exposition, we assume here that the sets R_A and R_B support compact representations. In case this is not true, we equivalently represent an element r_A (resp., r_B) sampled according to the distribution $\mathcal{D}_A$ (resp., $\mathcal{D}_B$) using the random coins input to the sampling algorithm (any element that cannot be sampled according to these distributions does not appear in the monoid M).

Constructing the Set. Next, we define the set X as follows:

$$X = (PP \cup \{\perp_X\}) \times (S_A \cup \{\perp_X\}) \times (S_B \cup \{\perp_X\}) \times (K \cup \{\perp_X\})$$

where:

- PP denotes the set of possible public parameters for the NIKE protocol.
- S_A and S_B denote the set of possible output shares for the parties A and B, respectively.
- K denotes the set of possible final keys that the parties A and B could agree on.
- $\perp_X$ is a special "terminal" symbol.

At a high level, a set element captures the gradual evolution of the public transcript of messages exchanged at various stages of the protocol, as well as the final computation of the shared key. In particular:

- A set element of the form $(\mathsf{pp}, \perp_X, \perp_X, \perp_X)$ represents the transcript of messages from the point of view of either party A or party B before the start of the protocol.
- A set element of the form $(\mathsf{pp}, \perp_X, s_B, \perp_X)$ represents the transcript of "received" messages from the point of view of party A after the first round of protocol execution.
- A set element of the form $(\mathsf{pp}, s_A, \perp_X, \perp_X)$ represents the transcript of "received" messages from the point of view of party B after the first round of protocol execution.
- A set element of the form $(\mathsf{pp}, s_A, s_B, k_{AB})$ represents the transcript of messages and the final secret key after the completion of the protocol (from the point of view of both parties A and B).

While we allow all other kinds of tuples in the set X from a syntactical point of view, they do not carry any semantic meaning. We enforce this in the manner in which we define the action operation, as described next.

Defining the Action. Finally, we define the action $\star : M \times X \to X$. We make use of the following functions associated with any NIKE protocol:

- $\mathsf{Gen}_A : PP \times R_A \to S_A$.
- $\mathsf{Gen}_B : PP \times R_B \to S_B$.
- $\mathsf{Combine}_A : PP \times R_A \times S_B \to K$.
- $\mathsf{Combine}_B : PP \times R_B \times S_A \to K$.

Given these functions, we define the action operation $\star : M \times X \to X$ as follows:

- For any $x = (x_0, x_1, x_2, x_3) \in X$, define

$$e_M \star (x_0, x_1, x_2, x_3) := (x_0, x_1, x_2, x_3).$$

- For any $r_A \in R_A$ and $\mathsf{pp} \in PP$, define
$$r_A \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) := (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \bot_X, \bot_X).$$
- For any $r_B \in R_B$ and $\mathsf{pp} \in PP$, define
$$r_B \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) := (\mathsf{pp}, \bot_X, \mathsf{Gen}_A(\mathsf{pp}, r_B), \bot_X).$$
- For any $r_A \in R_A$, any $\mathsf{pp} \in PP$, and any $s_B \in S_B$, define
$$r_A \star (\mathsf{pp}, \bot_X, s_B, \bot_X) := (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), s_B, \mathsf{Combine}_A(\mathsf{pp}, r_A, s_B)).$$
- For any $r_B \in R_A$, any $\mathsf{pp} \in PP$, and any $s_A \in S_A$, define
$$r_B \star (\mathsf{pp}, s_A, \bot_X, \bot_X) := (\mathsf{pp}, s_A, \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_B(\mathsf{pp}, r_B, s_A)).$$
- For any $r_A \in R_A$, any $r_B \in R_B$, and any $\mathsf{pp} \in PP$ define
$$(r_A \| r_b) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) := (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_A(\mathsf{pp}, r_A, \mathsf{Gen}_B(\mathsf{pp}, r_B))).$$
- All other action operations output the "terminal" set element $(\bot_X, \bot_X, \bot_X, \bot_X)$.

Lemma 3.16. *The monoid action $(M, X, \star)$ satisfies identity and compatibility if the NIKE protocol satisfies correctness.*

Proof. Identity is again immediate by construction. To prove compatibility, it suffices to prove that for any $r_A \in R_A$, any $r_B \in R_B$, and any $\mathsf{pp} \in PP$, we have

$$(r_A \oplus r_B) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) = r_A \star (r_B \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X)),$$
$$(r_B \oplus r_A) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) = r_B \star (r_A \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X)).$$

To see that this is indeed the case, observe that we have

$$\begin{aligned}
(r_A \oplus r_B) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) \quad &= (r_A \| r_B) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) \\
&= (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_A(\mathsf{pp}, r_A, \mathsf{Gen}_B(\mathsf{pp}, r_B))) \\
&= r_A \star (\mathsf{pp}, \bot_X, \mathsf{Gen}_B(\mathsf{pp}, r_B), \bot_X) \\
&= r_A \star (r_B \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X)).
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
(r_B \oplus r_A) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) \quad &= (r_A \| r_B) \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X) \\
&= (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_A(\mathsf{pp}, r_A, \mathsf{Gen}_B(\mathsf{pp}, r_B))) \\
&= (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \mathsf{Gen}_B(\mathsf{pp}, r_B), \mathsf{Combine}_B(\mathsf{pp}, r_B, \mathsf{Gen}_B(\mathsf{pp}, r_A))) \\
&= r_B \star (\mathsf{pp}, \mathsf{Gen}_A(\mathsf{pp}, r_A), \bot_X, \bot_X) \\
&= r_B \star (r_A \star (\mathsf{pp}, \bot_X, \bot_X, \bot_X)).
\end{aligned}$$

The second identity additionally exploits the following relationship

$$\mathsf{Combine}_A(\mathsf{pp}, r_A, \mathsf{Gen}_B(\mathsf{pp}, r_B)) = \mathsf{Combine}_B(\mathsf{pp}, r_B, \mathsf{Gen}_B(\mathsf{pp}, r_A)),$$

which holds whenever the NIKE protocol is correct. Hence, it follows that the monoid action $(M, X, \star)$ satisfies both identity and compatibility. This completes the proof of Lemma 3.16. □

Experiment $\mathsf{Expt}_{\ell,\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$:

1. The challenger samples a pair of group elements (g,h) as $g \leftarrow \mathcal{D}_{M,0}$ and $h \leftarrow \mathcal{D}_{M,1}$, and a set element $x \leftarrow \mathcal{D}_X$, and generates the following for each $i \in [\ell]$:

$$x_{i,0} = (g \oplus h)^{i-1} \star x, \quad x_{i,1} = (h \oplus g)^{i-1} \star x$$

$$x'_{i,0} = \left(g \oplus (h \oplus g)^{i-1}\right) \star x, \quad x'_{i,1} = \left(h \oplus (g \oplus h)^{i-1}\right) \star x.$$

   It then provides the following tuple to the adversary $\mathcal{A}$:

$$\left(x, \{x_{i,0}, x_{i,1}, x'_{i,0}, x'_{i,1}\}_{i\in[\ell]}\right)$$

2. The adversary $\mathcal{A}$ responds with a set element $y \in X$.

We say that the adversary $\mathcal{A}$ wins the experiment if $y = \left((g \oplus h)^{\ell}\right) \star x$.

Figure 2: The Security Definition for a Distributional ℓ-Unpredictable ℓ-Commutative Monoid Action

Putting together Lemma 3.14 and Lemma 3.16, we have that the group action $(M, X, \star)$ is indeed a commutative monoid action. Finally, it follows immediately from the security of the NIKE protocol that the group action $(M, X, \star)$ satisfies distributional unpredictability with respect to the distributions $\mathcal{D}_{M,b}$ for $b \in \{0,1\}$ and $\mathcal{D}_X$ defined as follows:

$$\mathcal{D}_{M,0} := \mathcal{D}_A, \quad \mathcal{D}_{M,1} := \mathcal{D}_B, \quad \mathcal{D}_X := \mathcal{D}_{\mathsf{pp}},$$

where $\mathcal{D}_A$, $\mathcal{D}_B$ and $\mathcal{D}_{\mathsf{pp}}$ are the efficiently sampleable distributions over the sets R_A, R_B and PP.

This establishes that the group action $(M, X, \star)$ indeed satisfies the definition for DUCMA (Definition 3.4), and completes the proof of Theorem 3.13. □

3.1.4 Generalization to Multi-Round Key Exchange

In this section, we generalize the aforementioned result to any ℓ-round key exchange protocol for $\ell \geq 1$. In particular, we show that any ℓ-round key exchange protocol is equivalent to a monoid action that satisfies a certain ℓ-commutator-like properties as well as a notion of *distributional ℓ-unpredictability*. For $\ell = 1$, these properties are exactly equivalent to commutativity and distributional unpredictability for monoid actions as described earlier, while for $\ell > 1$, these properties can be viewed as certain "naturally weakened" versions of commutativity and distributional unpredictability for monoid actions. We call this weakened primitive an distributional ℓ-unpredictable ℓ-commutative monoid action (abbreviated as ℓ-DUCMA).

Distributional ℓ-Unpredictable ℓ-Commutative Monoid Action (ℓ-DUCMA). Let $(M, X, \star)$ be a monoid action such that the set X supports efficient representation, and such that the "action operation" $\star$ is efficiently computable. Let $\mathcal{D}_{M,b}$ for $b \in \{0,1\}$ and $\mathcal{D}_X$ denote distributions over (subsets of) M and X, respectively, such that one can *efficiently* sample a monoid element $g \leftarrow \mathcal{D}_{M,0}$, a monoid element $h \leftarrow \mathcal{D}_{M,1}$ and a set element $x \leftarrow \mathcal{D}_X$.

Additionally, for any $g, h \in M$ and any $i \geq 1$, we define $(g \oplus h)^i$ as:

$$(g \oplus h)^i := \underbrace{(g \oplus h) \oplus (g \oplus h) \oplus \ldots \oplus (g \oplus h)}_{i\text{-times}}.$$

Note that when the monoid is not commutative, $(g \oplus h)^i$ and $(h \oplus g)^i$ can be distinct monoid elements. We additionally define $(g \oplus h)^0$ as:

$$(g \oplus h)^0 := e_M,$$

where e_M is the identity element for the monoid.

We now define the experiment $\mathsf{Expt}_{\ell,\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$ (parameterized by $\ell \geq 1$ as well as the distributions $\mathcal{D}_{M,0}$, $\mathcal{D}_{M,1}$, and $\mathcal{D}_X$) between a challenger and a probabilistic polynomial-time adversary $\mathcal{A}$ as in Figure 2.

Definition 3.17 (ℓ-DUCMA). A monoid action $(M, X, \star)$ with an efficiently computable action operation is said to be an ℓ-DUCMA with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$ and with respect to some security parameter λ if the following conditions are satisfied simultaneously:

- **ℓ-Commutativity:** For any $g, h \in M$ and any $x \in X$, we have

$$\left((g \oplus h)^\ell\right) \star x = \left((h \oplus g)^\ell\right) \star x.$$

- **Distributional ℓ-Unpredictability:** For any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the experiment $\mathsf{Expt}_{\ell,\mathcal{D}_{M,0},\mathcal{D}_{M,1},\mathcal{D}_X}$ is negligible in the security parameter λ.

Remark 3.18. As in the original definition of DUCMA, we again abstract out the details of the (efficient) sampling procedures that allow sampling a monoid element $g \leftarrow \mathcal{D}_{M,b}$ for $b \in \{0,1\}$ and a set element $x \leftarrow \mathcal{D}_X$. We simply assume that these algorithms take as input the security parameter λ and some random coins r, and output elements as per the desired distributions.

Remark 3.19. As in the original definition of DUCMA, we do not necessarily require the distributions $\mathcal{D}_{M,b}$ for $b \in \{0,1\}$ and $\mathcal{D}_X$ to be the uniform distributions over M and X, respectively.

Remark 3.20. As in the original definition of DUCMA, we do not assume that the monoid action $(M, X, \star)$ necessarily supports compact representations for elements in the monoid M. In particular, in order to represent a monoid element g sampled according to the distribution $\mathcal{D}_{M,0}$, one could simply use the random coins input to the sampling algorithm as an equivalent compact representation for g (so long as the action computation is efficient using this alternative representation).

ℓ-Round Key Exchange. We now define an ℓ-round key exchange (KE) protocol for $\ell \geq 1$. In the same vein as the NIKE definition, we define ℓ-round KE as a two-party protocol involving a pair of (non-uniform) probabilistic polynomial-time algorithms $A = \{\mathsf{A}_i\}_{i \in [0,\ell]}$ and $B = \{\mathsf{B}_i\}_{i \in [0,\ell]}$, where each individual algorithm A_i and B_i is formalized subsequently.

Before presenting the definition, we fix some notation. Let pp be the public parameters and let $s_{i,A}$ and $s_{i,B}$ be the message shares output by A and B, respectively, in round-i of the protocol (for each $i \in [\ell]$). We define a sequence of "transcript" variables $(\tau_0, \tau_1, \ldots, \tau_\ell)$ to maintain track of the messages exchanged between A, B, where for each $i \in [0, \ell]$, τ_i denotes the transcript of messages exchanged between parties A and B up until round-i. Formally, τ_i is defined as follows:

$$\tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{i,A}, s_{i,B}).$$

Definition 3.21 (ℓ-Round Key Exchange). An ℓ-round KE protocol is a tuple of probabilistic polynomial-time algorithms $\big(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\big)$ defined as follows:

- Setup takes as input a security parameter λ and output the public parameters pp.

- For each $i \in [0, \ell - 1]$, A_i takes as input the public parameters pp, a secret state $r_{i,A}$, and a transcript τ_i of the messages exchanged between parties A and B up until round-i, and outputs an updated secret state $r_{i+1,A}$ and a share $s_{i+1,A}$.

- For each $i \in [0, \ell - 1]$, B_i takes as input the public parameters pp, a secret state $r_{i,B}$, and a transcript τ_i of the messages exchanged between parties A and B up until round-i, and outputs an updated secret state $r_{i+1,B}$ and a share $s_{i+1,B}$.

- A_ℓ takes as input the public parameters pp, a secret state $r_{\ell,A}$, and a transcript τ_ℓ of the messages exchanged between parties A and B up until round-ℓ, and outputs the "final" key k_{AB}.

- B_ℓ takes as input the public parameters pp, a secret state $r_{\ell,B}$, and a transcript τ_ℓ of the messages exchanged between parties A and B up until round-ℓ, and outputs the "final" key k_{BA}.

Correctness. An ℓ-round KE protocol $\big(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\big)$ is said to be correct if for any $\mathsf{pp} \leftarrow \mathsf{Setup}$, and for any

$$(r_{i+1,A}, s_{i+1,A}) = \mathsf{A}_i(\mathsf{pp}, r_{i,A}, \tau_i), \quad (r_{i+1,B}, s_{i+1,B}) = \mathsf{B}_i(\mathsf{pp}, r_{i,B}, \tau_i),$$

for each $i \in [0, \ell - 1]$, we have

$$k_{AB} = k_{BA},$$

where $k_{AB} = \mathsf{A}_\ell(\mathsf{pp}, r_{\ell,A}, \tau_\ell)$ and $k_{BA} = \mathsf{B}_\ell(\mathsf{pp}, r_{\ell,A}, \tau_\ell)$, and where for each $i \in [0, \ell]$, the transcript τ_i is as defined earlier, namely:

$$\tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{i,A}, s_{i,B}).$$

Security. An ℓ-round KE protocol $\big(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\big)$ is said to be secure if for any $\mathsf{pp} \leftarrow \mathsf{Setup}$, and for any

$$(r_{i+1,A}, s_{i+1,A}) = \mathsf{A}_i(\mathsf{pp}, r_{i,A}, \tau_i), \quad (r_{i+1,B}, s_{i+1,B}) = \mathsf{B}_i(\mathsf{pp}, r_{i,B}, \tau_i),$$

for each $i \in [0, \ell - 1]$, and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

$$\Pr[\mathcal{A}(\mathsf{pp}, \tau_\ell) = k_{AB}] < \mathrm{negl}(\lambda),$$

where $k_{AB} = \mathsf{A}_\ell(\mathsf{pp}, r_{\ell,A}, \tau_\ell)$ and where the transcript τ_ℓ is as defined earlier, namely:

$$\tau_\ell = (\mathsf{pp}, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{\ell,A}, s_{\ell,B}).$$

Structural Formulation. We now formulate an ℓ-round KE protocol using a structural formulation that is again geared towards capturing the core property that two parties can compute the same secret key using two different sequences of computation across ℓ rounds of communication.

For ease of exposition, we make a (minor) alteration to our structural formulation for an ℓ-round KE protocol from the standard cryptographic definition presented earlier. In the structural formulation, we assume that the parties A and B commit to "some" random coins r_A and r_B at the beginning of the protocol, and then re-use these coins to generate their messages throughout the protocol. We note, however, this definition is essentially equivalent to the "lazy" randomness sampling strategy in the standard definition presented earlier; indeed, we can assume that the parties commit to some "master" random coins at the beginning of the protocol, and use these to derive the individual random coins to be used in each round (depending on the transcript of messages exchanged up until that round).

It turns out that this alternative definition (where the parties commit to some "master" random coins at the beginning of the protocol and re-use the same to generate messages throughout the protocol) makes it easier to capture the "natural" mathematical structure inherent to an ℓ-round KE protocol. Although this would result in "less practical" key exchanges and monoid actions, it allows us to only have to define two sampling distributions (one for each player) rather than 2ℓ (one for each player in each round) and lets us considerably simplify our proofs of equivalence later in this section. We illustrate this in more details subsequently.

Definition 3.22 (ℓ-Round KE (Structural Formulation)). Let PP, R, $\{S_{i,A}, S_{i,B}\}_{i\in[\ell]}$, $\{\Gamma_i\}_{i\in[0,\ell]}$, R_A, R_B, and K denote *sets*. More specifically:

- We let PP denote the set of public parameters and R denote the set of possible random coins used by the setup algorithm to output some public parameters from the set PP.

- For each $i \in [\ell]$, we let $S_{i,A}$ and $S_{i,B}$ denote the set of possible output shares in round-i for the parties A and B, respectively.

- For each $i \in [0, \ell]$, we let Γ_i denote the set of all possible transcripts of messages exchanged between the parties A and B until round i.

- We also let R_A and R_B denote the set of possible secret states for the parties A and B, respectively.

- Finally, we let K denote the set of possible final keys that the parties A and B could agree on at the end of the ℓ-round KE protocol.

Next, we define the following functions that map between these sets as below:

- $\mathsf{Setup} : 1^\lambda \times R \to PP$.

- $\{\mathsf{Gen}_{i,A} : PP \times R_A \times \Gamma_i \to S_{i+1,A}\}_{i\in[0,\ell-1]}$.

- $\{\mathsf{Gen}_{i,B} : PP \times R_B \times \Gamma_i \to S_{i+1,B}\}_{i\in[0,\ell-1]}$.

- $\mathsf{Combine}_A : PP \times R_A \times \Gamma_\ell \to K$.

- $\mathsf{Combine}_B : PP \times R_B \times \Gamma_\ell \to K$.

Finally, we impose the following correctness requirement on these functions: for any $\mathsf{pp} \in PP$, any $r_A \in R_A$ and any $r_B \in R_B$, letting

$$s_{i+1,A} = \mathsf{Gen}_{i,A}(\mathsf{pp}, r_A, \tau_i), \quad (s_{i+1}, B) = \mathsf{Gen}_{i,B}(\mathsf{pp}, r_B, \tau_i),$$

for each $i \in [0, \ell - 1]$, where $\tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{i,A}, s_{i,B})$, we have

$$\mathsf{Combine}_A(\mathsf{pp}, r_A, \tau_\ell) = \mathsf{Combine}_B(\mathsf{pp}, r_B, \tau_\ell),$$

where we again have $\tau_\ell = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B})$.

Security. Let $\mathcal{D}_{\mathsf{pp}}, \mathcal{D}_A$ and $\mathcal{D}_B$ denote efficiently sampleable distributions over the sets PP, R_A, and R_B, respectively. Based on the above structural formulation, we say that a ℓ-round KE protocol is $(\mathcal{D}_{\mathsf{pp}}, \mathcal{D}_A, \mathcal{D}_B)$-secure if for any $\mathsf{pp} \leftarrow \mathcal{D}_{\mathsf{pp}}$, any $r_A \leftarrow \mathcal{D}_A$ and any $r_B \leftarrow \mathcal{D}_B$, and for any probabilistic polynomial time algorithm $\mathcal{A}$, letting

$$s_{i+1,A} = \mathsf{Gen}_{i,A}(\mathsf{pp}, r_A, \tau_i), \quad (s_{i+1}, B) = \mathsf{Gen}_{i,B}(\mathsf{pp}, r_B, \tau_i),$$

for each $i \in [0, \ell - 1]$, where $\tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{i,A}, s_{i,B})$, we have

$$\Pr[\mathcal{A}(\mathsf{pp}, \tau_\ell) = \mathsf{Combine}_A(\mathsf{pp}, r_A, \tau_\ell)] < \mathrm{negl}(\lambda),$$

where we again have $\tau_\ell = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B})$.

Remark 3.23. As in the structural formulation for NIKE, we abstract out the details of the (efficient) sampling procedures that allow sampling as per the distributions $\mathcal{D}_{\mathsf{pp}}$, $\mathcal{D}_A$, and $\mathcal{D}_B$. We simply assume that these algorithms take as input the security parameter λ and some random coins r from the set R, and output elements as per the desired distributions.

Remark 3.24. As in the structural formulation for NIKE, we do not necessarily require the distributions $\mathcal{D}_{\mathsf{pp}}$, $\mathcal{D}_A$, and $\mathcal{D}_B$ to be the uniform distributions over the sets PP, R_A, and R_B, respectively.

Remark 3.25. As in the structural formulation for NIKE, we do not assume that the sets R_A and R_B necessarily support compact representations. Once again, in order to represent an element r_A sampled according to the distribution $\mathcal{D}_A$, one could simply use the random coins input to the sampling algorithm as an equivalent compact representation for r_A (so long as all relevant Function computations are efficient using this alternative representation).

Equivalence of ℓ-DUCMA and ℓ-round KE. We now formalize the equivalence of ℓ-DUCMA and ℓ-round KE. More concretely, we formally prove the more involved direction, namely, ℓ-round KE implies ℓ-DUCMA. The other direction (namely, ℓ-DUCMA implies ℓ-round KE) is relatively straightforward to show and essentially follows the same template as the construction of NIKE from DUCMA. Hence, we avoid detailing it.

ℓ-round KE implies ℓ-DUCMA. We state and prove the following theorem:

Theorem 3.26. *Any ℓ-round KE protocol satisfying Definition 3.22 implies an ℓ-DUCMA satisfying Definition 3.17.*

Proof. To prove this theorem, we show how to construct a group action $(M, X, \star)$ that satisfies the structural formulation for ℓ-DUCMA (Definition 3.17) with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$. We assume the existence of an ℓ-round KE protocol satisfying the corresponding structural formulation (Definition 3.22), including all the relevant sets and functions.

Constructing the Monoid. We begin by describing how to construct the monoid $(M, \oplus)$ underlying the monoid action $(M, X, \star)$. Recall that any ℓ-round KE protocol satisfying Definition 3.22 is associated with a pair of sets R_A and R_B, denoting the set of possible secret states for parties A and B, respectively. For any $r_A \in R_A$ and $r_B \in R_B$, define the following:

$$(r_A \| r_B)^i := \underbrace{r_A \| r_B \| r_A \| r_B \| \ldots \| r_A \| r_B}_{i\text{-times}},$$

$$(r_B \| r_A)^i := \underbrace{r_B \| r_A \| r_B \| r_A \| \ldots \| r_B \| r_A}_{i\text{-times}}.$$

Additionally, for any $r_A \in R_A$ and $r_B \in R_B$, define the following:

$$(r_A \| r_B)^0 = (r_B \| r_A)^0 := e_M,$$

where e_M is the special "identity" element. Next, we define the following auxiliary sets for each $i \in [\ell]$:

$$R_{A,B,i} = \{(r_A \| r_B)^i : r_A \in R_A, r_B \in R_B\}, \quad R_{B,A,i} = \{(r_B \| r_A)^i : r_B \in R_B, r_A \in R_A\}.$$

We also define the following auxiliary sets for each $i \in [\ell - 1]$:

$$R'_{A,B,i} = \{r_A \| (r_B \| r_A)^i : r_A \in R_A, r_B \in R_B\}, \quad R'_{B,A,i} = \{r_B \| (r_A \| r_B)^i : r_B \in R_B, r_A \in R_A\}.$$

At this point, we define the set M in the monoid $(M, \oplus)$ as:

$$M = R_A \cup R_B \cup \left(\bigcup_{i \in [\ell]} R_{A,B,i} \cup R_{B,A,i} \right) \cup \left(\bigcup_{i \in [\ell-1]} R'_{A,B,i} \cup R'_{B,A,i} \right) \cup \{e_M, \perp_M\},$$

where e_M is the special "identity" element and $\perp_M$ is a special "terminal" element.

Next, we define the associated monoid operation $\oplus$ as follows:

- For any $r_A \in R_A$ and any y such that $y \in R_{B,A,i}$ for $i \in [\ell - 1]$ or $y \in R_{B,A,i}$ for $i \in [\ell - 1]$, define
$$r_A \oplus y := r_A \| y.$$
- For any $r_B \in R_B$ and any y such that $y \in R_{A,B,i}$ for $i \in [\ell - 1]$ or $y \in R'_{A,B,i}$ for $i \in [\ell - 1]$, define
$$r_B \oplus y := r_B \| y.$$
- For any $\alpha \in M$, define $e_M \oplus \alpha := \alpha$.
- Any other possible monoid operation maps to the terminal element $\perp_M$.

Lemma 3.27. *$(M, \oplus)$ is a monoid.*

Proof. Closure and associativity are immediate by construction. Also, e_M serves as the (left) identity element for M. □

Remark 3.28. For $\ell = 1$, $(M, \oplus)$ is essentially a non-commutative version of same monoid that we constructed when proving that NIKE implies CUDMA.

Remark 3.29. Note that once again, for simplicity of exposition, we assume here that the sets R_A and R_B support compact representations. In case this is not true, we equivalently represent an element r_A (resp., r_B) sampled according to the distribution $\mathcal{D}_A$ (resp., $\mathcal{D}_B$) using the random coins input to the sampling algorithm (any element that cannot be sampled according to these distributions does not appear in M).

Constructing the Set. Next, we define the set X as follows:

$$X = (PP \cup \{\perp_X\}) \times (S_{A,1} \cup \{\perp_X\}) \times (S_{B,1} \cup \{\perp_X\}) \times \ldots \times (S_{A,\ell} \cup \{\perp_X\}) \times (S_{B,\ell} \cup \{\perp_X\}) \times (K \cup \{\perp_X\}).$$

where:

- PP denotes the set of possible public parameters for the NIKE protocol.
- For each $i \in [\ell]$, $S_{i,A}$ and $S_{i,B}$ denote the set of possible round-i output shares for the parties A and B, respectively.
- K denotes the set of possible final keys that the parties A and B could agree on.
- $\perp_X$ is a special "terminal" symbol.

As in the proof of NIKE implies DUCMA, a set element captures the gradual evolution of the public transcript of messages exchanged at various stages of the protocol, as well as the final computation of the shared key. In particular:

- A set element of the form $(\mathsf{pp}, \perp_X, \perp_X, \ldots, \perp_X, \perp_X, \perp_X)$ represents the transcript of messages from the point of view of either party A or party B before the start of the protocol.
- A set element of the form

  $$(\mathsf{pp}, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{i,A}, s_{i,B}, \perp_X, \perp_X, \ldots, \perp_X, \perp_X, \perp_X)$$

  represents the transcript of exchanged messages after round-i of protocol execution (from the point of view of both parties A and B).
- A set element of the form

  $$(\mathsf{pp}, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{\ell,A}, s_{\ell,B}, k_{AB})$$

  represents the transcript of messages and the final secret key after the completion of the protocol (from the point of view of both parties A and B).

While we allow all other kinds of tuples in the set X from a syntactical point of view, they do not carry any semantic meaning. We enforce this in the manner in which we define the action operation, as described next.

Defining the Action. Finally, we define the action $\star : M \times X \to X$. We make use of the following functions associated with any ℓ-round protocol as per Definition 3.22:

- $\{\mathsf{Gen}_{i,A} : PP \times R_A \times \Gamma_i \to S_{i+1,A}\}_{i \in [0,\ell-1]}$.
- $\{\mathsf{Gen}_{i,B} : PP \times R_B \times \Gamma_i \to S_{i+1,B}\}_{i \in [0,\ell-1]}$.
- $\mathsf{Combine}_A : PP \times R_A \times \Gamma_\ell \to K$.

- $\mathsf{Combine}_B : PP \times R_B \times \Gamma_\ell \to K$.

Given these functions, we define the action operation $\star : M \times X \to X$. We divide the operations into two types: **base actions** and **recursive actions.** We begin by defining the base action operations.

Base Action Operations:

- For any $x = (x_0, x_1, x_2, \ldots, x_{2\ell+1}) \in X$, define
$$e_M \star (x_0, x_1, x_2, \ldots, x_{2\ell+1}) := (x_0, x_1, x_2, \ldots, x_{2\ell+1}).$$
- For any $r_A \in R_A$ and any $\mathsf{pp} \in PP$, define
$$r_A \star (\mathsf{pp}, \perp_X, \perp_X, \perp_X, \ldots, \perp_X) := (\mathsf{pp}, s_{1,A}, \perp_X, \perp_X, \ldots, \perp_X).$$
where $s_{1,A} = \mathsf{Gen}_{0,A}(\mathsf{pp}, r_A)$.
- For any $r_B \in R_B$ and any $\mathsf{pp} \in PP$, define
$$r_B \star (\mathsf{pp}, \perp_X, \perp_X, \perp_X, \ldots, \perp_X) := (\mathsf{pp}, \perp_X, s_{1,B}, \perp_X, \ldots, \perp_X).$$
where $s_{1,B} = \mathsf{Gen}_{0,B}(\mathsf{pp}, r_B)$.
- For any $i \in [\ell - 1]$, any $r_A \in R_A$, any $\mathsf{pp} \in PP$, and any
$$\{s_{j,A} \in S_{j,A}, s_{j,B} \in S_{j,B}\}_{j \in [i-1]}, \quad s_{i,B} \in S_{i,B},$$
define
$$r_A \star (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[i-1]}, \perp_X, s_{i,B}, \perp_X, \ldots, \perp_X)$$
$$:= (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[i-1]}, s_{i,A}, s_{i,B}, s_{i+1,A}, \perp_X, \perp_X, \ldots, \perp_X),$$
where
$$s_{i,A} = \mathsf{Gen}_{i-1,A}(\mathsf{pp}, r_A, \tau_{i-1}), \quad s_{i+1,A} = \mathsf{Gen}_{i,A}(\mathsf{pp}, r_A, \tau_i),$$
where, as before, we have the transcript variables defined as
$$\tau_{i-1} = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{i-1,A}, s_{i-1,B}), \quad \tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{i,A}, s_{i,B}).$$
- For any $i \in [\ell - 1]$, any $r_B \in R_B$, any $\mathsf{pp} \in PP$, and any
$$\{s_{j,A} \in S_{j,A}, s_{j,B} \in S_{j,B}\}_{j \in [i-1]}, \quad s_{i,A} \in S_{i,A},$$
define
$$r_B \star (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[i-1]}, s_{i,A}, \perp_X, \perp_X, \ldots, \perp_X)$$
$$:= (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[i-1]}, s_{i,A}, s_{i,B}, \perp_X, s_{i+1,B}, \perp_X, \ldots, \perp_X),$$
where
$$s_{i,B} = \mathsf{Gen}_{i-1,B}(\mathsf{pp}, r_B, \tau_{i-1}), \quad s_{i+1,B} = \mathsf{Gen}_{i,B}(\mathsf{pp}, r_B, \tau_i),$$
where we again have the transcript variables τ_{i-1} and τ_i defined as before.

- For any $r_A \in R_A$, any $\mathsf{pp} \in PP$, and any
$$\{s_{j,A} \in S_{j,A}, s_{j,B} \in S_{j,B}\}_{j \in [\ell-1]}, \quad s_{\ell,B} \in S_{i,B},$$
define
$$r_A \star (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[\ell-1]}, \bot_X, s_{\ell,B}, \bot_X) := (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[\ell-1]}, s_{\ell,A}, s_{\ell,B}, k_{A,B}),$$
where
$$s_{\ell,A} = \mathsf{Gen}_{\ell-1,A}(\mathsf{pp}, r_A, \tau_{\ell-1}), \quad k_{A,B} = \mathsf{Combine}_A(\mathsf{pp}, r_A, \tau_\ell),$$
where, as before, we have the transcript variables defined as
$$\tau_{\ell-1} = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell-1,A}, s_{\ell-1,B}), \quad \tau_\ell = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B}).$$

- For any $r_B \in R_B$, any $\mathsf{pp} \in PP$, and any
$$\{s_{j,A} \in S_{j,A}, s_{j,B} \in S_{j,B}\}_{j \in [\ell-1]}, \quad s_{\ell,A} \in S_{i,A},$$
define
$$r_B \star (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[\ell-1]}, s_{\ell,A}, \bot_X, \bot_X) := (\mathsf{pp}, \{s_{j,A}, s_{j,B}\}_{j\in[\ell-1]}, s_{\ell,A}, s_{\ell,B}, k_{B,A}),$$
where
$$s_{\ell,B} = \mathsf{Gen}_{\ell-1,B}(\mathsf{pp}, r_B, \tau_{\ell-1}), \quad k_{B,A} = \mathsf{Combine}_B(\mathsf{pp}, r_B, \tau_\ell),$$
where we again have the transcript variables $\tau_{\ell-1}$ and τ_ℓ defined as before.

- All other base action operations of the form $r_A \star x$ for any $r_A \in R_A$ and any $x \in X$ output the "terminal" set element $(\bot_X, \bot_X, \ldots, \bot_X, \bot_X)$.

- Similarly, all other base action operations of the form $r_B \star x$ for any $r_B \in R_B$ and any $x \in X$ output the "terminal" set element $(\bot_X, \bot_X, \ldots, \bot_X, \bot_X)$.

Recursive Action Operations:

- For any $i \in [\ell]$, any $r_A \in R_A$, any $r_B \in R_B$, and any set element $x \in X$, define
$$((r_A \| r_B)^i) \star x := \underbrace{r_A \star (r_B \star (\ldots r_A \star (r_B \star x)))}_{i-\text{times}},$$
$$((r_B \| r_A)^i) \star x := \underbrace{r_B \star (r_A \star (\ldots r_B \star (r_A \star x)))}_{i-\text{times}}.$$

- For any $i \in [0, \ell - 1]$, any $r_A \in R_A$, any $r_B \in R_B$, and any set element $x \in X$, define
$$(r_A \| (r_B \| r_A)^i) \star x := r_A \star \left(\underbrace{r_B \star (r_A \star (\ldots r_B \star (r_A \star x)))}_{i-\text{times}} \right),$$
$$(r_B \| (r_A \| r_B)^i) \star x := r_B \star \left(\underbrace{r_A \star (r_B \star (\ldots r_A \star (r_B \star x)))}_{i-\text{times}} \right).$$

Lemma 3.30. *The monoid action $(M, X, \star)$ satisfies identity and compatibility if the NIKE protocol satisfies correctness.*

Proof. Identity and compatibility are again immediate by construction. □

Lemma 3.31. *The monoid action $(M, X, \star)$ satisfies ℓ-commutativity if the NIKE protocol satisfies correctness.*

Proof. To prove this, it suffices to show that for any $r_A \in R_A$, any $r_B \in R_B$, and any $\mathsf{pp} \in PP$, we have

$$((r_A \| r_B)^\ell) \star (\mathsf{pp}, \bot_X, \ldots, \bot_X) = ((r_B \| r_A)^\ell) \star (\mathsf{pp}, \bot_X, \ldots, \bot_X),$$

because any other ℓ-commutator-style expression maps to the all-$\bot_X$ terminal set element. Now, observe that we have the following by construction:

$$((r_A \| r_B)^\ell) \star (\mathsf{pp}, \bot_X, \ldots, \bot_X) = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B}, k_{A,B}),$$

$$((r_B \| r_A)^\ell) \star (\mathsf{pp}, \bot_X, \ldots, \bot_X) = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B}, k_{B,A}),$$

where

$$s_{i+1,A} = \mathsf{Gen}_{i,A}(\mathsf{pp}, r_A, \tau_i), \quad (s_{i+1}, B) = \mathsf{Gen}_{i,B}(\mathsf{pp}, r_B, \tau_i),$$

for each $i \in [0, \ell - 1]$, where $\tau_i = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{i,A}, s_{i,B})$. Also, we have

$$k_{A,B} = \mathsf{Combine}_A\left(\mathsf{pp}, r_A, \tau_\ell\right), \quad k_{B,A}\mathsf{Combine}_B\left(\mathsf{pp}, r_B, \tau_\ell\right),$$

where we again have $\tau_\ell = (\mathsf{pp}, s_{1,A}, s_{1,B}, \ldots, s_{\ell,A}, s_{\ell,B})$. Now, by the correctness of the NIKE protocol, we have

$$k_{A,B} = k_{B,A}. \qquad \square$$

This completes the proof of Lemma 3.31.

Lemma 3.32. *The monoid action $(M, X, \star)$ satisfies distributional ℓ-unpredictability if the NIKE protocol is secure.*

Proof. It follows immediately from the security of the NIKE protocol that the group action $(M, X, \star)$ satisfies distributional ℓ-unpredictability with respect to the distributions $\mathcal{D}_{M,b}$ for $b \in \{0, 1\}$ and $\mathcal{D}_X$ defined as follows:

$$\mathcal{D}_{M,0} := \mathcal{D}_A, \quad \mathcal{D}_{M,1} := \mathcal{D}_B, \quad \mathcal{D}_X := \mathcal{D}_{\mathsf{pp}},$$

where $\mathcal{D}_A$, $\mathcal{D}_B$ and $\mathcal{D}_{\mathsf{pp}}$ are the efficiently sampleable distributions over the sets R_A, R_B and PP in the structural formulation of NIKE. □

Putting together Lemma 3.27, Lemma 3.30, Lemma 3.31, and Lemma 3.32 establishes that the group action $(M, X, \star)$ indeed satisfies the structural formulation for ℓ-DUCMA (Definition 3.4) whenever the ℓ-round KE satisfies the corresponding structural formulation (Definition 3.22). This completes the proof of Theorem 3.26. □

Remark 3.33. We remark that a key exchange protocol (by definition) does not guarantee any security in presence of malicious parties, and it only considers the honest setting. Thus, for the aforementioned equivalence of key exchange protocol and unpredictable monoid action we *do not* need to consider the cases in which one (or more) parties do not follow the protocol (e.g., by sending an improperly formatted message to other parties). As a side note, this issues does not arise in case of a *noninteractive* key exchange since there is no interaction. We refer the reader to [FHKP13] for more details on the security models for NIKE. This makes it substantially easier for us to guarantee a correct monoid action, since we never come across circumstances where it is difficult to decide whether an operation should map to the terminal element or not (which would be the case if, for instance, we had to test set membership due to a malicious player).

Remark 3.34. We note that our results also hold in a natural sense for key exchange protocols that are not perfectly correct but satisfy overwhelming success probability (e.g., protocols based on Learning With Rounding [BPR12]). For these protocols, one can define an algebraic notion of "approximate equality" of set elements (see [AMPR19, AMP19] for more details) and prove an almost identical result to that of perfectly correct key exchange protocols. In other words, for these key exchange protocols, we form squares that "almost always commute." However, we chose to present our formal results based on key exchange protocols with perfect correctness for the ease of exposition.

3.1.5 String-Concatenation Monoid Action Oracles

We now define an unconditional variant of DUCMA, which we refer to as generic string concatenation monoid action (SCMA) oracle. Informally speaking, an SCMA oracle (with certain restrictions as outlined subsequently) is a DUCMA *in the strongest possible sense*, much like how a random oracle is one-way in the strongest possible sense (see [IR89] for a detailed exposition on the latter).

Definition 3.35 (Generic SCMA Oracle). A generic string concatenation monoid action (SCMA) oracle $\mathbf{M}(\cdot,\cdot)$ over an alphabet $\Sigma \subset \{0,1\}^*$ is a random variable whose values are functions $\mathbf{M} : \Sigma^* \times \{0,1\}^* \to \{0,1\}^*$ such that the following conditions hold:

1. For any $s \in \Sigma^*$ and any $x \in \{0,1\}^*$, $\mathbf{M}(s,x)$ is distributed independently of both $\mathbf{M}(\Sigma^* \setminus \{s\}, \{0,1\}^*)$ and $\mathbf{M}(\Sigma^*, \{0,1\}^* \setminus \{x\})$, subject to the restrictions that:
   (a) For any $x \in \{0,1\}^*$, we have $\mathbf{M}(\phi, x) = x$, where ϕ denotes the empty string element in Σ^*.
   (b) For any $a \in \Sigma$, any $s \in \Sigma^*$, and any $x \in \{0,1\}^*$, we have
   $$\mathbf{M}(a \| s, x) = \mathbf{M}(a, \mathbf{M}(s,x)).$$
2. For any $s \in \Sigma^*$ and any $x, y \in \{0,1\}^*$, $\Pr[\mathbf{M}(s,x) = y]$ is a rational number.

In this paper, we consider SCMA oracles that additionally satisfy certain commutative (or commutator-like) properties.

Definition 3.36 (Commutative SCMA Oracle). A generic SCMA oracle is said to be commutative if for any $a, b \in \Sigma$ and any $x \in \{0,1\}^*$, we have

$$\mathbf{M}(ab, x) = \mathbf{M}(ba, x).$$

Definition 3.37 (k-Commutator SCMA Oracle). A generic SCMA oracle is said to be a k-commutator (for $k \geq 1$) if for any $a, b \in \Sigma$ and any $x \in \{0,1\}^*$, we have

$$\mathbf{M}((ab)^k, x) = \mathbf{M}((ba)^k, x).$$

Remark 3.38. In the rest of the paper, we slightly abuse notation by using $|s|$ for any $s \in \Sigma^*$ to denote the number of symbols/elements in Σ that s contains, rather than the length of the bit-representation of s.

Restricted SCMA Oracles. We now introduce some restrictions of a generic SCMA oracle. We begin by defining a special set element, which we call the "initial" set element.

Definition 3.39 (Base Set Element). Let $\mathbf{M}(\cdot,\cdot)$ be a generic SCMA over the alphabet $\Sigma \subset \{0,1\}^*$. The k-base set element $x_0 \in \{0,1\}^*$ is a special set element such that for any $s_0, s_1 \in \Sigma^*$ such that $|s_0|, |s_1| < 2k$, we must have

$$\mathbf{M}(s_0, x_0) = \mathbf{M}(s_1, x_0) \implies s_0 = s_1.$$

In other words, for any $x \in \{0,1\}^*$ and any $\ell < 2k$, there exists *at most one* ℓ-length sequence of elements in Σ that acts on x_0 to yield x.

Definition 3.40 (Level of a Set Element). Let $\mathbf{M}(\cdot,\cdot)$ be a generic SCMA over the alphabet $\Sigma \subset \{0,1\}^*$, and let x_0 be a k-base set element as defined above for some $k \geq 1$. We define a corresponding "level" function $\mathsf{Level}_k : \{0,1\}^* \to \mathbb{Z}$ as follows:

$$\mathsf{Level}_k(x) = \begin{cases} \ell & \text{if } \exists s \in \Sigma^\ell \text{ such that } \ell < 2k \text{ and } \mathbf{M}(s, x_0) = x, \\ -1 & \text{otherwise.} \end{cases}$$

Remark 3.41. The level of any set element $x \in \{0,1\}^*$ is *unique* by the above definition, and hence the function Level_k is well-defined.

Remark 3.42. The level of the base set element x_0 is zero.

We now introduce a "two-layered" restriction of a generic SCMA oracle with a 1-base set element x_0 as defined above such that:

- The action of a monoid element $s \in \Sigma^*$ on any set element x is defined if and only if $\mathsf{Level}_1(x) \geq 0$, i.e., there exists some $s' \in \Sigma^*$ such that $\mathbf{M}(s', x_0) = x$. Any action computation on a set element x such that $\mathsf{Level}_1(x) = -1$ yields the symbol $\perp$.
- The action of a monoid element $s \in \Sigma^*$ on the base set element x_0 is allowed if and only if $s \in \Sigma^\ell$ for $\ell \leq 2$, i.e. s is either the empty string (which represents the identity element of the string concatenation monoid), or s is of the form $s = a$ or $s = ab$ for $a, b \in \Sigma$. In other words, we only allow at most two "layers" of action computation on x_0; any action computation that involves more layers yields the symbol $\perp$.

We call this a restricted SCMA oracle, and define it formally below.

Definition 3.43 (Restricted SCMA Oracle). A restricted SCMA oracle $\mathbf{M}(\cdot,\cdot)$ over an alphabet $\Sigma \subset \{0,1\}^*$ is a random variable whose values are functions $\mathbf{M} : \Sigma^* \times \{0,1\}^* \cup \{\perp\} \to \{0,1\}^* \cup \{\perp\}$ and which satisfies all of the properties of a generic SCMA oracle, with the following additional constraints:

1. For any $s \in \Sigma^*$, we have $\mathbf{M}(s, \perp) = \perp$.
2. For any $s \in \Sigma^*$ and any $x \in \{0,1\}^*$, we have $\mathbf{M}(s, x) = \perp$ if *either* of the following conditions holds:
    - **Either** $\mathsf{Level}_1(x) = -1$.
    - **Or** $|s| + \mathsf{Level}_1(x) > 2$ (where $|s|$ denotes the length of the string s).

Generic k-restricted SCMA Oracle. We now formally define a more general "k-layered" restriction of a generic SCMA oracle with a k-base set element x_0.

Definition 3.44 (Generic k-restricted SCMA Oracle). A generic k-restricted SCMA oracle $\mathbf{M}(\cdot, \cdot)$ over an alphabet $\Sigma \subset \{0,1\}^*$ is a random variable whose values are functions $\mathbf{M} : \Sigma^* \times \{0,1\}^* \cup \{\perp\} \to \{0,1\}^* \cup \{\perp\}$ and which satisfies all of the properties of a generic SCMA oracle, with the following additional constraints:

1. For any $s \in \Sigma^*$, we have $\mathbf{M}(s, \perp) = \perp$.
2. For any $s \in \Sigma^*$ and any $x \in \{0,1\}^*$, we have $\mathbf{M}(s, x) = \perp$ if *either* of the following conditions holds:
    - **Either** $\mathsf{Level}_k(x) = -1$.
    - **Or** $|s| + \mathsf{Level}_k(x) > 2k$ (where $|s|$ denotes the length of the string s).

In this paper, we consider k-restricted SCMA oracles that additionally satisfy certain commutator-like properties, defined formally below.

Definition 3.45 (k'-Commutator k-restricted SCMA Oracle). A generic k-restricted SCMA oracle with initial element x_0 is said to be a k'-commutator (for $k' \in [1, k]$) if for any $a, b \in \Sigma$, we have

$$\mathbf{M}\left((ab)^{k'}, x_0\right) = \mathbf{M}\left((ba)^{k'}, x_0\right).$$

In particular, we use k-restricted SCMA oracles that are also k-commutator. In the rest of the paper, when we refer to k-restricted SCMA oracles, we assume that they are additionally k-commutator by default (unless specified otherwise); hence, we do not explicitly specify the k-commutator property.

Remark 3.46. Our definition of a commutative SCMA oracle says that for monoid elements $a, b \in \Sigma$, and for a set element $x \in \{0,1\}^*$, we have: $\mathbf{M}((ab), x) = \mathbf{M}((ba), x)$. However, this *does not* necessarily imply that $ab = ba$, which is a significantly stronger requirement. In our definition, the monoid elements ab and ba are allowed to be distinct (and hence, a and b are allowed to be distinct), with the only requirement being that their action on the same set element $x \in \{0,1\}^*$ produces the same set element $y \in \{0,1\}^*$. The same holds for our general definition of k-commutator SCMA, where we have $\mathbf{M}\left((ab)^k, x\right) = \mathbf{M}\left((ba)^k, x\right)$, but *do not* enforce that $(ab)^k = (ba)^k$. In other words, we *do not* enforce that for any pair of set elements $(x, y) \in \{0,1\}^* \times \{0,1\}^*$, there exists a *unique* monoid element that maps x to y. Since we do not enforce the monoid elements themselves to be identical but only the output of their actions to be identical, there can be *exponentially many* monoid elements at *each level* of the generic string concatenation monoid.

Key Exchange from SCMA Oracle. We now state the following lemma.

Lemma 3.47. *There exists a construction of $(2k-1)$-round key exchange protocol from any k-restricted SCMA oracle $\mathbf{M}(\cdot,\cdot)$ over a sufficiently large alphabet Σ.*

Proof. The proof follows essentially immediately from the equivalence of NIKE and DUCMA described earlier. We describe it here for completeness. In the initial phase of the protocol we assume that two parties (Alice and Bob) have access to an base set element x_0. In addition, we assume that Alice (respectively Bob) has a random "private" monoid element a (respectively b), chosen randomly from from Σ.

Our protocol proceeds as follows. For each round $i \in [2k-1]$, we first describe how Alice computes her message and then we explain what Bob does in the ith round of the protocol.

- If $i = 1$, Alice queries the oracle on the input (a, x_0) and she receives a response $\mathbf{M}(a, x_0)$. She then sets $\mathsf{m}_{AB}^{(1)} = \mathbf{M}(a, x_0)$ and sends it to Bob.

- If $i = 1$, Bob queries the oracle on the input (b, x_0) and he receives a response $\mathbf{M}(b, x_0)$. He then sets $\mathsf{m}_{BA}^{(1)} = \mathbf{M}(b, x_0)$ and sends it to Alice.

- If $i > 1$ is odd where $i = 2t+1$, Alice queries the oracle on the input $(a, \mathsf{m}_{BA}^{(2t)})$ and she receives a response $\mathbf{M}(a, \mathsf{m}_{BA}^{(2t)})$. Observe that by construction we have
$$\mathsf{m}_{BA}^{(2t)} = \mathbf{M}((ba)^t, x_0).$$
She then sets $\mathsf{m}_{AB}^{(i)} = \mathbf{M}(a, \mathsf{m}_{BA}^{(2t)})$ and sends $\mathsf{m}_{AB}^{(i)}$ to Bob.

- If $i > 1$ is odd where $i = 2t+1$, Bob queries the oracle on the input $(b, \mathsf{m}_{AB}^{(2t)})$ and he receives a response $\mathbf{M}(b, \mathsf{m}_{AB}^{(2t)})$. Observe that by construction we have
$$\mathsf{m}_{AB}^{(2t)} = \mathbf{M}((ab)^t, x_0).$$
He then sets $\mathsf{m}_{BA}^{(i)} = \mathbf{M}(b, \mathsf{m}_{AB}^{(2t)})$ and sends $\mathsf{m}_{BA}^{(i)}$ to Alice.

- If $i > 1$ is even where $i = 2t$, Alice queries the oracle on the input $(a, \mathsf{m}_{BA}^{(2t-1)})$ and she receives a response $\mathbf{M}(a, \mathsf{m}_{BA}^{(2t-1)})$. Observe that by construction we have
$$\mathsf{m}_{BA}^{(2t-1)} = \mathbf{M}(b(ab)^{t-1}, x_0).$$
She then sets $\mathsf{m}_{AB}^{(i)} = \mathbf{M}(a, \mathsf{m}_{BA}^{(2t-1)})$ and sends $\mathsf{m}_{AB}^{(i)}$ to Bob.

- If $i > 1$ is even where $i = 2t$, Bob queries the oracle on the input $(b, \mathsf{m}_{AB}^{(2t-1)})$ and he receives a response $\mathbf{M}(b, \mathsf{m}_{AB}^{(2t-1)})$. Observe that by construction we have
$$\mathsf{m}_{AB}^{(2t-1)} = \mathbf{M}(a(ba)^{t-1}, x_0).$$
He then sets $\mathsf{m}_{BA}^{(i)} = \mathbf{M}(b, \mathsf{m}_{AB}^{(2t-1)})$ and sends $\mathsf{m}_{BA}^{(i)}$ to Bob.

Finally, Alice and Bob can compute the final shared secret as follows:

- Alice computes the final shared secret as $S_A = \mathbf{M}(a, \mathsf{m}_{BA}^{(2k-1)})$.
- Bob computes the final shared secret as $S_B = \mathbf{M}(b, \mathsf{m}_{AB}^{(2k-1)})$.

To argue correctness, observe that based on the description of the protocol above, we have

$$\mathsf{m}_{AB}^{(2k-1)} = \mathbf{M}(a(ba)^{k-1}, x_0), \quad \mathsf{m}_{BA}^{(2k-1)} = \mathbf{M}(b(ab)^{k-1}, x_0).$$

It follows that

$$S_A = \mathbf{M}((ab)^k, x_0) = \mathbf{M}((ba)^k, x_0) = S_B,$$

where we used the k-commutator property of SCMA. Thus, Alice and Bob arrive at the same value after following the protocol. □

We now sketch a proof of security of the protocol for $k = 1$. Our proof is similar to that of protocols in the generic group model. First observe that since the output of $\mathbf{M}$ is random subject to monoid axioms, it follows for any adversary that makes at most polynomially many queries to the oracle, i.e., at most $\mathrm{poly}(\log(|\Sigma|))$ many queries, we have

$$(x_0, \mathbf{M}(a, x_0), \mathbf{M}(b, x_0), \mathbf{M}(ab, x_0)) \overset{s}{\approx} (x_0, \mathbf{M}(a, x_0), \mathbf{M}(b, x_0), \mathbf{M}(u, x_0)),$$

where u is a randomly chosen element from Σ. To argue statistical security for the general case $k > 1$, first note that the messages sent by Alice/Bob have the form

$$\begin{aligned} \mathsf{m}_{AB}^{(i)} &= \mathbf{M}(a(ba)^t, x_0), \quad i = 2t + 1 \\ \mathsf{m}_{AB}^{(i)} &= \mathbf{M}(b(ab)^{t-1}, x_0), \quad i = 2t \\ \mathsf{m}_{BA}^{(i)} &= \mathbf{M}(b(ab)^t, x_0), \quad i = 2t + 1 \\ \mathsf{m}_{BA}^{(i)} &= \mathbf{M}(a(ba)^{t-1}, x_0), \quad i = 2t. \end{aligned}$$

In addition, since for a random (generic) k-restricted SCMA, the k'-commutator property does not hold if $k' < k$ (unless with negligible probability), one can argue that a strong form of DDH-like property holds, i.e.,

$$(x_0, \mathbf{M}(a, x_0), \mathbf{M}(b, x_0), \mathbf{M}(ab, x_0), \mathbf{M}(ba, x_0)) \overset{s}{\approx}$$
$$(x_0, \mathbf{M}(a, x_0), \mathbf{M}(b, x_0), \mathbf{M}(u, x_0), \mathbf{M}(u', x_0)),$$

where both u and u' are chosen randomly from Σ. By relying on this property, we can replace ab and ba with u and v in the tuples of messages sent by Alice and Bob, as shown above. It follows that

$$\begin{aligned} \mathsf{m}_{AB}^{(i)} &\overset{s}{\approx} \mathbf{M}(a(v)^t, x_0), \quad i = 2t + 1 \\ \mathsf{m}_{AB}^{(i)} &\overset{s}{\approx} \mathbf{M}(b(u)^{t-1}, x_0), \quad i = 2t \\ \mathsf{m}_{BA}^{(i)} &\overset{s}{\approx} \mathbf{M}(b(u)^t, x_0), \quad i = 2t + 1 \\ \mathsf{m}_{BA}^{(i)} &\overset{s}{\approx} \mathbf{M}(a(v)^{t-1}, x_0), \quad i = 2t. \end{aligned}$$

By setting $x_0 = \mathbf{M}(u^{t-1}, x_0)$ and $x_1 = \mathbf{M}(v^{t-1}, x_0)$, and relying once again on the DDH-like property it follows that the final secret is unpredictable for an eavesdropper, as required.

Remark 3.48. We remark that in the last step of the protocol, Alice and Bob only make a single query to the oracle in order to compute the final shared secret, and they do not exchange any messages. Therefore, we do not need to count the last step as an extra "round."

Remark 3.49. We note that each round consists of three sub-rounds as defined earlier. In the first two sub-round Alice and Bob query the oracle on their inputs respectively. Finally, in the last sub-round, Alice/Bob sends a message to the other party. We also remark that the protocol above works for semi-honest parties where the parties honestly follow the protocol. Since in each round there is no "illegal" query to the oracle, no party would send $\perp$ to the other party.

Remark 3.50. We note that in the construction of key exchange protocol above the results holds unconditionally (statistically) as there is no computational assumption over the k-restricted string concatenation monoid (SCMA). Therefore, this result should be interpreted along similar to a line of works on feasibility results based on idealized assumptions. For instance, one can easily show that certain idealized models such as generic group model (GGM) [Sho97] or algebraic group model (AGM) [FKL18] imply a key exchange protocol, and these results hold unconditionally. On the same vein, it has long been known how to construct noninteractive zero-knowledge proof from an interactive zero-knowledge protocol in the random oracle model (ROM) [FS87].

3.2 Separating $2k$-round Key Exchange from $(2k+1)$-round Key Exchange

Our (informal) goal is to black-box separate any $2k$-round KE protocol from any $(2k+1)$-round KE protocol. Subsequently, in Section 3.3, we show that the separation of $(2k+1)$-round KE from any $(2k+2)$-round KE follows analogously. Concretely, we establish the impossibility of a secure $2k$-round KE protocol where the participants Alice and Bob only make queries to a generic $(k+1)$-restricted SCMA oracle. Note that this immediately (black-box) separates $2k$-round KE from any $(2k+1)$-round KE protocol. In particular, we wish to establish that for any $2k$-round KE protocol where the participants Alice and Bob only make queries to a $(k+1)$-restricted SCMA oracle, there exists an attacker Eve that, given access access to the generic $(k+1)$-restricted SCMA oracle and to the the messages exchanged publicly between Alice and Bob during the protocol, also finds the final secret key that Alice and Bob agree on with non-negligible probability.

Before we formalize this goal, we define $2k$-round key exchange and introduce several notations for executions and probability distributions associated with a $2k$-round key exchange. In the rest of the section, when we refer to a generic $(k+1)$-restricted SCMA, we assume that it is $(k+1)$-commutator by default.

3.2.1 Round-Based Definition of $2k$-round Key Exchange

We begin by formally defining a $2k$-round key exchange protocol where the participants are Alice and Bob, and Eve is the adversary, all of whom have access to a $(k+1)$-restricted SCMA oracle. We assume w.l.o.g. that Alice, Bob, and Eve will never issue the same $(k+1)$-restricted SCMA oracle query twice. Also, we assume that Alice (resp., Bob) issues at most n_A (resp., n_B) $(k+1)$-restricted SCMA oracle queries.

Rounds and Sub-Rounds. Each round i (for $i \geq 1$) consists of a message $\mathsf{m}_{AB}^{(i)}$ sent from Alice to Bob and a message $\mathsf{m}_{BA}^{(i)}$ sent from Bob to Alice. Each round i consists of several sub-rounds (i,j) for $j \in [n_i+1]$ defined as follows:

- Each sub-round (i, j) for $j \in [n_i]$ begins with *either* Alice *or* Bob issuing a *single* (new) $(k+1)$-restricted SCMA oracle query, and ends with with Eve issuing her (new) oracle queries based on the set of messages exchanged between Alice and Bob so far, defined as

$$\mathsf{m}^{[i-1]} = \left\{\mathsf{m}_{AB}^{(1)}, \mathsf{m}_{BA}^{(1)}, \ldots, \mathsf{m}_{AB}^{(i-1)}, \mathsf{m}_{BA}^{(i-1)}\right\}.$$

  In these sub-rounds, Alice and Bob do not exchange any messages.

  Remark 3.51. The astute reader may observe that this restriction of a single oracle query by Alice or Bob in each sub-round matches the notion of "semi-normal form" for a key exchange protocol defined originally in [BM09], with the only difference being that [BM09] defined each round to involve a single query from either Alice or Bob, whereas we apply this restriction to each sub-round. This is because the analysis of [BM09] is agnostic of the number of rounds (indeed, their separation result holds for key exchange protocols with any polynomially many rounds), while our analysis crucially relies on the number of rounds (and is agnostic of the number of sub-rounds).

- Sub-round $(n_i + 1)$ involves the following steps that happen simultaneously:

  - Alice computes her message $\mathsf{m}_{AB}^{(i)}$ and sends it to Bob.
  - Simultaneously, Bob computes his message m_{BA} and sends it to Alice.

  While computing the above messages, both Alice and Bob only use their own oracle queries till round $(i-1)$, and the set of messages exchanged between Alice and Bob till round $(i-1)$, defined as

$$\mathsf{m}^{[i-1]} = \left\{\mathsf{m}_{AB}^{(1)}, \mathsf{m}_{BA}^{(1)}, \ldots, \mathsf{m}_{AB}^{(i-1)}, \mathsf{m}_{BA}^{(i-1)}\right\}.$$

We define the sub-rounds as above for ease of exposition, and for simplifying the attack analysis presented subsequently.

Queries and Views. We use the following notations to denote the queries and views of Alice, Bob, and Eve at the end of various sub-rounds:

- $Q_A^{(i,j)}$ (resp., $Q_B^{(i,j)}$ and $Q_E^{(i,j)}$): denotes the set of $(k+1)$-restricted SCMA oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i, j).

- $P_A^{(i,j)}$ (resp., $P_B^{(i,j)}$ and $P_E^{(i,j)}$): denotes the set of query-response pairs corresponding to the $(k+1)$-restricted SCMA oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i, j). More formally, for $\alpha \in \{A, B, E\}$, we have

$$P_\alpha^{(i,j)} = \left\{((s, x, y = \mathbf{M}(s, x))) : (s, x) \in Q_\alpha^{(i,j)}\right\}.$$

- $V_A^{(i,j)}$ (resp., $V_B^{(i,j)}$ and $V_E^{(i,j)}$): denotes the views of Alice (resp., Bob and Eve) by the end of sub-round (i, j). More formally, for $\alpha \in \{A, B\}$, we have

$$V_\alpha^{(i,j)} = \left(\mathsf{r}_\alpha, \mathsf{m}^{(i,j)}, P_\alpha^{(i,j)}\right),$$

where r_A (resp., r_B) denotes the internal randomness of Alice (resp., Bob). In addition, we have

$$V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

In particular, the view of Eve does not have any randomness since Eve does not use any randomness.

We adopt the notation $\mathcal{Q}(\cdot)$ from [BM09] to denote an operator that extracts the set of queries from any set of $(k+1)$-restricted SCMA oracle query-answer pairs or views; namely, for any set of query-response pairs P and any view $V = (r, \mathsf{m}, P)$, we have

$$\mathcal{Q}(P) = \mathcal{Q}(V = (r, \mathsf{m}, P)) = \{q = (s, x) : \exists y, (s, x, y) \in P\}.$$

Finally, we analogously use the notations $Q_A^{(i)}$ (resp,. $Q_B^{(i)}$ and $Q_E^{(i)}$), $P_A^{(i)}$ (resp,. $P_B^{(i)}$ and $P_E^{(i)}$) and $V_A^{(i)}$ (resp,. $V_B^{(i)}$ and $V_E^{(i)}$) to denote the set of queries asked by Alice (resp., Bob and Eve), the set of query-response pairs corresponding to the queries asked by Alice (resp., Bob and Eve), and the view of Alice (resp., Bob and Eve) at the end of all sub-rounds of round i in the KE protocol.

Executions and Distributions. A (full) execution of Alice, Bob, and Eve can be described by a tuple $(\mathsf{r}_A, \mathsf{r}_B, \mathbf{M})$, where r_A denotes Alice's random tape, r_B denotes Bob's random tape, and $\mathbf{M}$ denotes the generic $(k+1)$-restricted SCMA (note that Eve is deterministic). We denote by $\mathcal{E}$ the distribution over (full) executions, obtained by running the algorithms for Alice, Bob and Eve with uniformly chosen random tapes r_A, r_B, and a uniformly sampled generic $(k+1)$-restricted SCMA $\mathbf{M}$. We denote by $\Pr_{\mathcal{E}}[P_A^{(i,j)}]$ (resp., $\Pr_{\mathcal{E}}[P_B^{(i,j)}]$ and $\Pr_{\mathcal{E}}[P_E^{(i,j)}]$) the probability that $P_A^{(i,j)}$ (resp., $P_B^{(i,j)}$ and $P_E^{(i,j)}$) is the set of query-response pairs corresponding to the $(k+1)$-restricted SCMA oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i, j) during the execution.

For any (i, j), for any sequence of exchanged messages $\mathsf{m}^{(i,j)}$, and for any set of $(k+1)$-restricted SCMA oracle query-answer pairs $P_E^{(i,j)}$, we denote by $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ the joint distribution over the views $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ of Alice and Bob in their own (partial) executions up to just before the sub-round (i, j), conditioned on the event that:

1. the transcript of messages exchanged between Alice and Bob until this point being equal to $\mathsf{m}^{(i,j)}$, and
2. the set of all $(k+1)$-restricted SCMA oracle query-answer pairs corresponding to the queries issued by Eve until this point being equal to $P_E^{(i,j)}$.

We denote the probability of the aforementioned event by $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}]$. Similar to in [BM09], we use the distribution $\mathcal{V}(\mathsf{m}^{(i,j)})$ to essentially capture the conditional distribution of Alice's and Bob's views in the eyes of the attacker Eve who knows the public messages exchanged between Alice and Bob, and has learned all $(k+1)$-restricted SCMA oracle query-answer pairs described in $P_E^{(i,j)}$.

Intersection Queries and Equivalence Queries. We now formally define intersection and equivalence queries. Recall that for any (i, j), $Q_A^{(i,j)}$ (resp., $Q_B^{(i,j)}$) denotes the set of $(k+1)$-restricted SCMA oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i, j).

Intersection Queries. We define the set of intersection queries

$$Q_{A\cap B}^{(i,j)} = Q_A^{(i,j)} \cap Q_B^{(i,j)},$$

to be the set of *common* $(k+1)$-restricted SCMA oracle queries issued by *both* Alice *and* Bob until sub-round-(i,j).

Equivalence Queries. We now define the concept of *equivalence* queries with respect to the $(k+1)$-restricted SCMA oracle queries issued by Alice and Bob.

Definition 3.52 (Equivalence Queries). Let $q_A = (s_A, x_A)$ and $q_B = (s_B, x_B)$ be two queries issued by Alice and Bob to the $(k+1)$-restricted SCMA oracle. We say that q_A and q_B are *equivalent* queries if the following conditions hold simultaneously:

- $(s_A, x_A) \neq (s_B, x_B)$, $\mathbf{M}(s_A, x_A) \neq \perp$, $\mathbf{M}(s_B, x_B) \neq \perp$.
- One of the following two cases must be true (x_0 being the $(k+1)$-base set element for the $(k+1)$-restricted SCMA):
  - **Either** there exist $s'_A, s'_B \in \Sigma^*$ such that
  $$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = s_B \| s'_B.$$
  - **Or** there exist $a, b \in \Sigma$, and $s'_A, s'_B \in \Sigma^*$, such that
  $$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = (ab)^{k+1}, \quad s_B \| s'_B = (ba)^{k+1}.$$

Note that the first condition immediately implies that $\mathbf{M}(s_A, x_A) = \mathbf{M}(s_B, x_B)$. Additionally, the second condition also implies that

$$\begin{aligned} \mathbf{M}(s_A, x_A) &= \mathbf{M}(s_A \| s'_A, x) = \mathbf{M}((ab)^{k+1}, x) \\ &= \mathbf{M}((ba)^{k+1}, x) = \mathbf{M}(s_B \| s'_B, x) = \mathbf{M}(s_B, x_B). \end{aligned}$$

In other words, equivalence queries essentially depict two different sequences of queries to the $(k+1)$-restricted SCMA oracle leading to the same (valid) output, and the two possibilities mentioned above depict the only scenarios that could lead to such a "collision" between two different sequence of queries with non-negligible probability (this follows immediately from statistical independence properties of the outputs of a $(k+1)$-restricted SCMA oracle on uncorrelated inputs).

Remark 3.53. We remark here that we could also have some additional classes of equivalence queries that are essentially combinations of the above two cases. However, we avoid explicitly enumerating them since we do not need them for our eventual separation proof.

Next, we define the equivalence relation $\mathcal{R}_{A\equiv B}$ as follows:

$$\mathcal{R}_{A\equiv B} = \begin{cases} 1 & \text{if and only if } q_A \text{ and } q_B \text{ are equivalent,} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define the set of equivalence queries

$$Q_{A\equiv B}^{(i,j)} \quad = \{(q_A, q_B \in Q_A^{(i,j)} \times Q_B^{(i,j)} : \mathcal{R}_{A\equiv B}(q_A, q_B) = 1\},$$

to be the set of equivalence query-pairs (where each pair consists of a query issued by Alice and a query issued by Bob) until sub-round-(i,j).

Good Events. For any (i,j), for any sequence of exchanged messages $\mathsf{m}^{(i,j)}$, and for any set of $(k+1)$-restricted SCMA oracle query-answer pairs $P_E^{(i,j)}$ (corresponding to queries issued by Eve) such that $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$, we define the following:

- The event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is defined over the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and is said to hold if and only if:
$$Q_{A\cap B}^{(i,j)} \subseteq \mathcal{Q}(P_E^{(i,j)}),$$
where $Q_{A\cap B}^{(i,j)}$ and $Q_{A\equiv B}^{(i,j)}$ are determined by $Q_A^{(i,j)}$ and $Q_B^{(i,j)}$, which are in turn determined by sampling the views of Alice and Bob as
$$\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

- The event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is defined over the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and is said to hold if and only if:
$$Q_{A\cap B}^{(i,j)} \subseteq \mathcal{Q}(P_E^{(i,j)}) \quad \text{and} \quad \forall (q_A, q_B) \in Q_{A\equiv B}^{(i,j)}, q_A \in \mathcal{Q}(P_E^{(i,j)}) \vee q_b \mathcal{Q}(P_E^{(i,j)}),$$
where $Q_{A\cap B}^{(i,j)}$ and $Q_{A\equiv B}^{(i,j)}$ are again determined by $Q_A^{(i,j)}$ and $Q_B^{(i,j)}$, which are in turn again determined by sampling the views of Alice and Bob as
$$\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

Intuitively, the event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ indicates that Eve has issued all queries that have been issued by both both Alice and Bob, while the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ indicates that Eve has not only issued all queries that have been issued by both both Alice and Bob, but also at least one query from each pair of equivalence queries issued by Alice and Bob.

Finally, we denote by $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ the distributions obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the events $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, respectively.

3.2.2 The Main Separation Theorem for KE

Our goal is to prove the following main theorem.

Theorem 3.54 (Main Theorem for KE Separation). *Let Π be a $2k$-round KE protocol between Alice and Bob such that:*

- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted SCMA oracle, and use random tapes r_A and r_B, respectively.*

- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

Then for every $0 < \delta < \rho$, there exists an attacker Eve that only has access to the public messages exchanged between Alice and Bob, makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted SCMA oracle, and produces an output s_E such that $\Pr[s_E = s_B] > \rho - \delta$.

Before describing Eve's attack algorithm, we introduce a special form of $2k$-round KE (the existence of which is implied by any $2k$-round KE protocol). The special form of $2k$-round KE is introduced purely to make our attack analysis easier; our attack applies to any $2k$-round KE protocol.

3.2.3 KE with Equivalence Complete Query Pattern

We now introduce what we call an *equivalence complete* query pattern for Alice and Bob during an execution of a $2k$-round KE protocol, which essentially depicts a sequence of queries issued by Alice and Bob to the $(k+1)$-restricted SCMA oracle, albeit subject to certain constraints as described subsequently.

Definition 3.55 (Query Length). Let $\mathbf{M}(\cdot,\cdot)$ be a generic $(k+1)$-restricted SCMA oracle, and let (s, x) be a query to $\mathbf{M}$. Let $s = s_1 \| \ldots \| s_\ell$ be a "decomposition" of s such that each $s_i \in \Sigma^*$ for $i \in [\ell]$. We say that the "length" of the query (for this decomposition) is ℓ. Observe that, by the associative properties of the $(k+1)$-restricted SCMA oracle, we must have

$$\mathbf{M}(s, x) = \mathbf{M}(s_1, \mathbf{M}(s_2, \ldots, \mathbf{M}(s_\ell, x) \ldots)).$$

Remark 3.56. Note that the length of the query may vary depending on the decomposition of the string s, and may be different from $|s|$, which denotes the unique number of symbols from Σ in the string s.

Definition 3.57 (Equivalence Complete Query Pattern). Let Q be any set of queries to a $(k+1)$-restricted SCMA oracle, such that each query $q \in Q$ is of the form $q = (s, x) \in \Sigma^* \times \{0,1\}^*$. We say that Q is equivalence complete if the following conditions are satisfied (x_0 being the $(k+1)$-base set element of the generic $(k+1)$-SCMA oracle):

- Informally, for any query $q \in Q$, the query set Q also contains all the "split" versions of this query. Formally, for each $q = (s, x) \in Q$ such that $x = \mathbf{M}(s', x_0)$ and such that $s \| s' = a_1 \ldots a_\ell$ for $\ell > 1$ (where for each $j \in [\ell]$, we have $a_j \in \Sigma$), there exists a subset of "single-element" queries $S \subset Q$ of the form
$$S = \{q_1 = (s_1, x_1), \ldots, q_\ell = (s_\ell, x_\ell)\},$$
such that for each $j \in [\ell]$, we
$$s_j = a_j, \quad x_j = \mathbf{M}(a_{j+1}, \mathbf{M}(a_{j+2}, \ldots, \mathbf{M}(a_\ell, x_0) \ldots)).$$
- Informally, for any query $q \in Q$ that is a substring of either $(ab)^{k+1}$ or $(ba)^{k+1}$, and which potentially "triggers" a build-up to an equivalence query of the form $\mathbf{M}\left((ab)^{k+1}, x_0\right) = \mathbf{M}\left((ba)^{k+1}, x_0\right)$, the query set Q also contains all the possible ways to compute this equivalence query. Formally, for any $q = (s, x) \in Q$ such that $x = \mathbf{M}(s', x_0)$ and such that there exist distinct elements $a, b \in \Sigma$ such that
$$|s \| s'| > 2, \quad s \| s' \in \mathsf{SUBSTRING}\left((ab)^{k+1}\right) \cup \mathsf{SUBSTRING}\left((ba)^{k+1}\right),$$
where $\mathsf{SUBSTRING}\left((ab)^{k+1}\right)$ and $\mathsf{SUBSTRING}\left((ba)^{k+1}\right)$ denote the sets of all possible substrings of $(ab)^{k+1}$ and $(ba)^{k+1}$, respectively, we must have
$$S_0 \subset Q \wedge S_1 \subset Q,$$

where the query subsets S_0 and S_1 are defined as:

$$S_0 = \left\{ \widetilde{q} = (\widetilde{s}, x_0) : \widetilde{s} \in \mathsf{SUBSTRING}\left((ab)^{k+1} \right) \right\},$$

$$S_1 = \left\{ \widetilde{q} = (\widetilde{s}, x_0) : \widetilde{s} \in \mathsf{SUBSTRING}\left((ba)^{k+1} \right) \right\}.$$

Definition 3.58 (KE Protocol with Equivalence Complete Query Pattern). Let Π be any key exchange protocol as defined in Section 3.2.1. KE is said to have equivalence complete query pattern if for any round i, letting $Q_A^{(i)}$ and $Q_B^{(i)}$ denote the set of queried asked by Alice and Bob to the $(k+1)$-SCMA oracle, we have that $Q_A^{(i)}$ and $Q_B^{(i)}$ are both equivalence complete query patterns as per Definition 3.57.

Equivalence Queries Follow Intersection Queries. We now state and prove that for any $2k$-round KE protocol with equivalence complete query pattern where Alice and Bob make queries to a $(k+1)$-restricted SCMA oracle, for each equivalence query, there exists a corresponding intersection query such that if Eve makes this intersection query, she makes a query that is either identical to or equivalent to the original equivalence query. It is this special property of a KE protocol with equivalence complete query pattern that makes our subsequent attack analysis significantly simpler.

We note here that this step constitutes the core novelty of our attack analysis, and is necessitated by the additional algebraic structure that is inherent to a $(k+1)$-restricted SCMA oracle over and above a plain random oracle. In particular, the proofs of [IR89, BM09] do not require this additional analysis since any equivalence query is, by definition, an intersection query by default for a plain random oracle. However, since this is not the case for a $(k+1)$-restricted SCMA oracle, we additionally need to establish that Eve can "cover" all equivalence queries by identifying only the intersection queries. We formally prove this via Lemmas 3.59 and 3.60, that we state and prove below.

Lemma 3.59 (Equivalence Queries Follow Intersection Queries-1). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob till round i of a $2k$-round KE protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) = ((s_A, x_A), (s_B, x_B)) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist $s'_A, s'_B \in \Sigma^*$ such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = s_B \| s'_B.$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the KE protocol. Then there exists a set intersection queries

$$S = \{q_1, \ldots, q_\ell\} \subset Q_A^{(i)} \cap Q_B^{(i)},$$

such that if Eve asks each query in S, she asks a query that is equivalent to both the queries q_A and q_B.

Proof. Since Alice and Bob are only given the initial set-element x_0, they must have each issued a sequence of queries building up to the queries (s'_A, x_0) and (s'_B, x_0), respectively. By the definition of equivalence complete query pattern, they also issues all possible singleton queries leading up to these queries. In addition, they also issued all possible singleton queries building up to the queries (s_A, x_A) and (s_B, x_B), respectively. Suppose

$$s_A \| s'_A = s_B \| s'_B = a_1 a_2 \ldots a_\ell,$$

where for each $j \in [\ell]$, we have $a_j \in \Sigma$. Then, by definition of equivalence complete query pattern, there exists a set of queries of the form

$$S = \{q_1 = (s_1, x_1), \ldots, q_\ell = (s_\ell, x_\ell)\},$$

such that for each $j \in [\ell]$, we

$$s_j = a_j, \quad x_j = \mathbf{M}(a_{j+1}, \mathbf{M}(a_{j+2}, \ldots, \mathbf{M}(a_\ell, x_0) \ldots)),$$

such that $S \subset Q_A^{(i)} \cap Q_B^{(i)}$, and such that q_1 is equivalent to both q_A and q_B. This completes the proof of Lemma 3.59. □

Lemma 3.60 (Equivalence Queries Follow Intersection Queries-2). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob till round i of a $2k$-round KE protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist $a, b \in \Sigma$, and $s'_A, s'_B \in \Sigma^*$, such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = (ab)^{k+1}, \quad s_B \| s'_B = (ba)^{k+1},$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the KE protocol. Then we must have

$$q_A \in Q_A^{(i)} \cap Q_B^{(i)} \quad \text{or} \quad q_B \in Q_A^{(i)} \cap Q_B^{(i)}.$$

Proof. We will show that if Alice and Bob compute an equivalence query of the aforementioned form in at most $2k$ rounds, then either Alice or Bob must have computed a query that triggered the equivalence complete query pattern. Therefore, (at least) one of Alice and Bob will have computed the equivalence query in all possible ways, implying the existence of a corresponding intersection query by definition.

Based on the definition of equivalence query as outlined in Definition 3.52, in this scenario, Alice and Bob effectively compute an equivalence query of the form

$$(ab)^{k+1} \star x_0 = (ba)^{k+1} \star x_0,$$

given only the base set element x_0. To do this, they each must make queries of the form $\mathbf{M}\left(t_1, t_2 \star x\right)$ where $t_1 || t_2$ is a right substring of either $(ab)^{k+1}$ or $(ba)^{k+1}$ and send these back and forth between one another, constantly building t_2. Suppose we assume that if either Alice or Bob makes multiple queries of the above form in the same round that build upon one another, we replace them with a single query. Note that this will not change the final equivalence query or whether or not we have triggered an equivalence complete query pattern.

With this assumption, we may assume that Alice and Bob make no more than $2k$ queries of the form $q_i = \mathbf{M}\left(s_i, q_{i-1}\right)$ for $i \in [2k]$ such that

$$s_1 \| \ldots \| s_{2k} = (ab)^{k+1} \quad \text{or} \quad s_1 \| \ldots \| s_{2k} = (ba)^{k+1}.$$

If less than $2k$ queries are used by either Alice or Bob (or both), we simply assume that the extra s_i strings are empty strings.

By the pigeonhole principle, at least one of the s_i strings must contain a string concatenation of both a and b. Therefore, by the definition of equivalence complete query pattern (Definition 3.57) ,at least one of Alice and Bob must have computed all possible ways to compute that particular equivalence query, and hence made the corresponding queries to the $(k+1)$-restricted SCMA oracle. This completes the proof of Lemma 3.60. □

From any KE to KE with Equivalence Complete Query Pattern. Next, we show that any $2k$-round KE protocol (for polynomially large k) implies the existence of a $2k$-round KE protocol while incurring only a polynomial blow-up in the number of queries issued to the $(k+1)$-restricted SCMA oracle by Alice and Bob (assuming that Alice and Bob make at most polynomially many queries to the $(k+1)$-restricted SCMA oracle in the original $2k$-round KE protoco). More formally, we state and prove the following lemma.

Lemma 3.61. *Assuming the existence of any secure $2k$-round KE protocol (for polynomially large k) between Alice and Bob with correctness probability ρ such that Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted SCMA oracle such that n_A and n_B are at most polynomially large, there exists a secure $2k$-round KE protocol between Alice and Bob with correctness probability ρ such that the query pattern for Alice and Bob is equivalence complete, and such that Alice and Bob make at most $\mathrm{poly}(k, n_A)$ and $\mathrm{poly}(k, n_B)$ queries to a generic $(k+1)$-restricted SCMA oracle.*

Proof. Given any $2k$-round KE, we can immediately construct a $2k$-round KE with equivalence complete query pattern as follows: we allow Alice and Bob to behave exactly as in the original $2k$-round KE except that they additionally ask the extra queries entailed by the definition of equivalence complete query pattern, and ignore the corresponding responses of the $(k+1)$-restricted SCMA oracle to these additional queries. Since both Alice and Bob are PPT algorithms, the lengths of their queries are also poly-bounded. Hence, the blow-ups in the number of queries issued by Alice and Bob are at most $\mathrm{poly}(k, n_A)$ and $\mathrm{poly}(k, n_B)$, respectively. Note that neither changes the transcript of messages exchanged by Alice and Bob, nor does it change the view of Eve. This immediately implies that the following must hold:

- If the original $2k$-round KE is correct with probability ρ, then the new $2k$-round KE protocol with equivalence complete query pattern is also correct with the same probability ρ.
- If the original $2k$-round KE is secure against any PPT adversary Eve, then the new $2k$-round KE protocol with equivalence complete query pattern is also secure against any PPT adversary EVE.

This completes the proof of Lemma 3.61. □

3.2.4 Attacking KE with Equivalence Complete Query Pattern

At this point, we shift focus from the main theorem to the following auxiliary theorem.

Theorem 3.62 (Auxiliary Theorem). *Let Π be a $2k$-round KE protocol between Alice and Bob such that:*

- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted SCMA oracle, and use random tapes r_A and r_B, respectively.*
- *Π has an equivalence complete query pattern per Definition 3.57.*
- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

Then for every $0 < \delta < \rho$, there exists an attacker Eve that only has access to the public messages exchanged between Alice and Bob, makes at most $O(n_A n_B/\delta^2)$ queries to the generic $(k+1)$-restricted SCMA oracle, and produces an output s_E such that $\Pr[s_E = s_B] > \rho - \delta$.

We note that Theorem 3.62, together with Lemma 3.61, immediately implies Theorem 3.54, which is the main theorem that we originally set out to prove[1]. Hence, in the rest of the paper, we focus purely on proving Theorem 3.62 in the context of a $2k$-round KE with equivalence complete query pattern.

The Attack Algorithm. We now describe the algorithm that the attacker Eve uses to break any $2k$-round KE protocol with equivalence complete query pattern. We follow essentially the same attack strategy as used in [BM09]; the main difference lies in actually analyzing the attack algorithm in our setting, as presented subsequently. However, we summarize the attack strategy here for the sake of completeness.

The attack algorithm is parameterized by some constant $\epsilon > 0$, which we assume is smaller than $1/10$. Let (i, j) denote some sub-round of the KE protocol, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of messages between Alice and Bob until sub-round (i, j), and let $P_E^{(i,j)}$ denote the set of $(k + 1)$-restricted SCMA oracle query-answer pairs until sub-round (i, j) asked by Eve. At this point, Eve proceeds as follows during sub-round (i, j):

- If $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] = 0$, Eve aborts.
- Otherwise, as long as there is a query $q = (s, x)$ for $s \in \Sigma^{k+1}$ and x such that $\mathsf{Level}(x) \neq -1$ such that
$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}[q \in \mathcal{Q}(V_A^{(i,j)})] > \frac{\epsilon}{n_B},$$
or
$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}[q \in \mathcal{Q}(V_B^{(i,j)})] > \frac{\epsilon}{n_A},$$
Eve issues the lexicographically first such query q to the $(k + 1)$-restricted SCMA oracle and adds the query-response pair $(q, \mathbf{M}(q))$ to $P_E^{(i,j)}$.
- Eve continues in this way until there remains no additional query that Eve can ask, at which point she stops and waits for the next sub-round to commence.

Eventually, at the end of all sub-rounds of the final round $2k$ (when Eve is also done with asking her oracle queries), Eve samples
$$\left(V_A^{(2k)}, V_B^{(2k)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right),$$
computes Alice's final output s_A determined by $V_A^{(2k)}$, and outputs $s_E = s_A$ as its own output.

Note that Eve's algorithm above may ask much more than $n_A n_B$ queries. However, we will show that the probability that Eve needs to ask more than $O(n_A n_B/\epsilon^2)$ queries is bounded by $O(\epsilon)$, and hence we can stop Eve after asking these many queries without changing significantly her success probability.

Remark 3.63. As in the case of the attack algorithm of [BM09], our attacking algorithm above is not computationally efficient, as in general computing the probability distribution $\mathcal{V}\left(\mathsf{m}^k, P_E^{(k)}\right)$ could be a hard problem since it involves "inverting" the algorithms of Alice and Bob to a certain extent. But because computing these probabilities is in $\#\mathbf{P}$ we can use known techniques to approximate them with arbitrarily

[1] Note that the number of queries made by Eve when attacking the KE protocol with equivalence complete query pattern is actually independent of k; the factor of $\mathrm{poly}(k)$ blowup in the number of queries over and above any KE protocol (as in the statement of Theorem 3.54) is already implicit in the number of queries n_A and n_B in the statement of Theorem 3.62.

good precision using an NP-oracle. In particular this means that our attacker (as was the case in previous works) is computationally efficient in a relativized world in which **P = NP**, and hence our result also rules out relativizing reductions from any $(k+1)$-restricted SCMA to $2k$-round key exchange (and hence, relativizing reductions from $(2k+1)$-round KE to $2k$-round KE).

Analyzing Events. Our target is to prove Theorem 3.62. To do so, we first analyze some events for any $2k$-round KE protocol with equivalence complete query pattern. Recall that the event Good_0 holds if Eve has found all of the intersection queries, while event Good_1 holds if Eve has found all of the intersection *and* equivalence queries. We now state and prove the following lemma.

Lemma 3.64 ($\mathsf{Good}_0 \implies \mathsf{Good}_1$ **(Informal)).** *For any KE protocol with equivalence complete query pattern as described above, the event* Good_0 *holds if and only if the event* Good_1 *holds. In other words, if Eve finds all of the intersection queries during an execution of the KE protocol, it also finds all of the equivalence queries during the same execution of the KE protocol.*

More formally, we state and prove the following.

Lemma 3.65 ($\mathsf{Good}_0 \implies \mathsf{Good}_1$ **(Formal)).** *Given any KE protocol with equivalence complete query pattern as described above, let (i,j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i,j), and let $P_E^{(i,j)}$ denote some sequence of $(k+1)$-restricted SCMA oracle query-answer pairs until sub-round (i,j), such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. Then, we have*

$$\Pr_{\mathcal{E}}[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)] = 1.$$

Let $\mathcal{V}(\mathsf{m}^{(i,j)})$ denote the conditional distribution of Alice's and Bob's views in the eyes of the attacker Eve who knows the public messages exchanged between Alice and Bob, and has learned all $(k+1)$-restricted SCMA oracle query-answer pairs described in $P_E^{(i,j)}$. Finally, let $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ denote the distributions obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the events $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, respectively. Then, assuming Lemma 3.65, we also immediately obtain the following corollary.

Corollary 3.66. *$\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ are identical.*

Proof. Lemma 3.65 follows immediately from Lemmas 3.59 and 3.60. □

We define two additional events, which we call *fail* event and *long* event.

Fail Event. Given any $2k$-round KE protocol with equivalence complete query pattern, let (i,j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i,j), and let $P_E^{(i,j)}$ denote the sequence of $(k+1)$-restricted SCMA oracle query-answer pairs made by Eve until sub-round (i,j), such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. We define the event $\mathsf{Fail}^{(i,j)}$ to be the event that:

- **EITHER** the query (made by Alice or Bob) to the $(k+1)$-restricted SCMA oracle after this sub-round is an intersection query but is not contained in $P_E^{(i,j)}$.

- **OR** the query (made by Alice or Bob) to the $(k+1)$-restricted SCMA oracle after this sub-round is an equivalence query w.r.t. some query issued earlier by the other party, but $P_E^{(i,j)}$ does not contain a query that is either identical or equivalent to this query,

and this is the first instance of Eve missing either an intersection query or an equivalence query. Let the event $\mathsf{Fail} = \bigvee_{(i,j)} \mathsf{Fail}^{(i,j)}$ be the event that at some point during the $2k$-round KE protocol with equivalence query pattern, an intersection query is missed by Eve.

Long Event. We also denote by Long the event that Eve makes more than $O(n_A n_B/\epsilon^2)$ queries when attacking any $2k$-round KE protocol with equivalence complete query pattern.

Theorem 3.62 immediately follows from the following lemmas.

Lemma 3.67 (Attack is successful). *For any sub-round (i,j) of the KE protocol with equivalence complete query pattern, we have*

$$\Pr_{\mathcal{E}}[\mathsf{Fail}^{(i,j)}] = O\left(\frac{\epsilon}{(n_A+n_B)}\right).$$

Hence, by union bound, we have $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$.

Lemma 3.68 (Attack is efficient). *We have $\Pr_{\mathcal{E}}[\mathsf{Long}] = O(\epsilon)$.*

3.2.5 Proof of Lemma 3.67: The Attack is Successful

We prove Lemma 3.67 by proving the following stronger result.

Lemma 3.69. *For any sub-round (i,j) of the KE protocol with equivalence complete query pattern, let let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i,j), and let $P_E^{(i,j)}$ denote the sequence of $(k+1)$-restricted SCMA oracle query-answer pairs made by Eve until sub-round (i,j), such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. Then we have*

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}^{(i,j)} | \mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] = O\left(\frac{\epsilon}{(n_A+n_B)}\right).$$

To see why Lemma 3.69 implies Lemma 3.67, observe that $\mathsf{Fail}^{(i,j)}$ is the event that Eve fails to query an intersection query or an equivalence query for the first time in sub-round (i,j), and hence, Eve found all intersection queries and equivalence queries during the execution up until sub-round (i,j), meaning that $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ holds. Hence, we must have

$$\Pr_{\mathcal{E}}[\mathsf{Fail}^{(i,j)}] \le \Pr_{\mathcal{E}}\left[\mathsf{Fail}^{(i,j)} | \mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] = O\left(\frac{\epsilon}{(n_A+n_B)}\right),$$

which is precisely the statement of Lemma 3.67.

In what follows, we prove Lemma 3.69 by using a product characterization of the distribution $\mathcal{GV}_1$.

Product Characterization of $\mathcal{GV}_1$. Given any KE protocol with equivalence complete query pattern as described above, let (i, j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i, j), and let $P_E^{(i,j)}$ denote the set of $(k + 1)$-restricted SCMA oracle query-answer pairs until sub-round (i, j) asked by Eve, such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. Also, let $\mathcal{V}(\mathsf{m}^{(i,j)})$ denote the conditional distribution of Alice's and Bob's views in the eyes of the attacker Eve who knows the public messages exchanged between Alice and Bob, and has learned all $(k + 1)$-restricted SCMA oracle query-answer pairs described in $P_E^{(i,j)}$, and let $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ be the distributions obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the events $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, respectively.

We now show that the distribution $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is equal to the distribution obtained by taking some product distribution $\mathcal{A} \times \mathcal{B}$ and conditioning it on the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$. More formally, we state and prove the following lemma.

Lemma 3.70 (Product Characterization of $\mathcal{GV}_1$). *There exists a distribution $\mathcal{A}$ (resp., a distribution $\mathcal{B}$) over Alice's view (resp., Bob's view) upto sub-round (i, j) such that*

$$\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) = (\mathcal{A} \times \mathcal{B})|\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

Proof. We defer the proof of this lemma to later in Section 3.2.5. Our proof here follows very closely the proof of graph characterization (Lemma 4.5) of [BM09], except for some additional analysis with respect to equivalence queries at the very end of the proof. □

Having established the product characterization of $\mathcal{GV}_1$, we now turn to analyzing the distribution $\mathcal{GV}_0$, which is the distribution obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ (the event in which Eve only queries all intersection queries for Alice and Bob).

Product Characterization of $\mathcal{GV}_0$. The corollary below follows immediately from Lemma 3.70 and Corollary 3.66.

Corollary 3.71 (Product Characterization of $\mathcal{GV}_0$). *There exists a distribution $\mathcal{A}$ (resp., a distribution $\mathcal{B}$) over Alice's view (resp., Bob's view) upto sub-round (i, j) such that*

$$\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) = (\mathcal{A} \times \mathcal{B})|\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

Graph Characterization of $\mathcal{GV}_0$. The above product characterization implies that we can think of $\mathcal{GV}_0$ as a distribution over random edges of some bipartite graph G. Using an analysis very similar to that used in [BM09], we will show that every vertex in G is connected to most of the vertices on the other side.

Constructing the Graph. Given any KE protocol with equivalence complete query pattern as described above, let (i, j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i, j), and let $P_E^{(i,j)}$ denote the set of $(k + 1)$-restricted SCMA oracle query-answer pairs until sub-round (i, j) asked by Eve, such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. We construct a bipartite graph $G^{(i,j)}$ with vertex-sets $(\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)})$ and edge-set $E^{(i,j)}$ as follows:

- Every node $u \in \mathcal{U}_A^{(i,j)}$ corresponds to a view A_u of Alice (until sub-round (i,j)) that is in the support of the distribution $\mathcal{A}$ obtained from Lemma 3.70. We let the number of nodes corresponding to the view A_u to be proportional to $Pr_{\mathcal{A}}[A_u]$, meaning that the distribution $\mathcal{A}$ corresponds to the uniform distribution over the vertices in the partition $\mathcal{U}_A^{(i,j)}$.
- Every node $v \in \mathcal{U}_B^{(i,j)}$ similarly corresponds to a view B_v of Bob (until sub-round (i,j)) that is in the support of the distribution $\mathcal{B}$ obtained from Lemma 3.70. We again let the number of nodes corresponding to the view B_v to be proportional to $Pr_{\mathcal{B}}[B_v]$, meaning that the distribution $\mathcal{B}$ corresponds to the uniform distribution over the vertices in the partition $\mathcal{U}_B^{(i,j)}$.
- We define $Q_u = \mathcal{Q}(A_u) \setminus \mathcal{Q}\left(P_E^{(i,j)}\right)$ for $u \in \mathcal{U}_A^{(i,j)}$ to be the set of queries outside of those in $P_E^{(i,j)}$ that were asked by Alice in the view A_u.
- Similarly, we define $Q_v = \mathcal{Q}(B_v) \setminus \mathcal{Q}\left(P_E^{(i,j)}\right)$ for $v \in \mathcal{U}_B^{(i,j)}$ to be the set of queries outside of those in $P_E^{(i,j)}$ that were asked by Bob in the view B_v.
- We put an edge between a pair of nodes (u,v) (denoted by $u \sim v$) if and only if $Q_u \cap Q_v = \phi$.

Analyzing the Graph. We first state the following immediate corollary of Lemma 3.70 and Corollary 3.71.

Corollary 3.72. *Let $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ be sampled uniformly from the probability space $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$. Then the distribution of $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ is identical to the distribution of (A_u, B_v) sampled by picking a random edge (u,v) in the graph $G^{(i,j)}$ constructed as above, and letting A_u and B_v be the views of Alice and Bob associated with u and v, respectively.*

Next, we argue that the graph $G^{(i,j)}$ constructed as above is *dense*. More formally, we state and prove the following lemma:

Lemma 3.73. *Let $G^{(i,j)} = (\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)}, E^{(i,j)})$ be the graph constructed as above. Also, for any vertex $w \in (\mathcal{U}_A^{(i,j)} \cup \mathcal{U}_B^{(i,j)})$, let $\deg(w)$ denote the degree of the vertex w. Then, for each vertex $u \in \mathcal{U}_A^{(i,j)}$ and each vertex $v \in \mathcal{U}_B^{(i,j)}$, we have*

$$\deg(u) \geq (1-2\epsilon)|\mathcal{U}_B^{(i,j)}|, \quad \deg(v) \geq (1-2\epsilon)|\mathcal{U}_A^{(i,j)}|.$$

Proof. We defer the detailed proof of this lemma to later in Section 3.2.5. □

Finishing Proof of Lemma 3.69. Finally, we use the product characterization of $\mathcal{GV}_1$ and the graph characterization of $\mathcal{GV}_0$ to finish the proof of Lemma 3.69, and hence finish the proof of Lemma 3.67. We defer the detailed proof to later in Section 3.2.5.

Remaining Proofs. It remains to prove that Eve's attack is efficient, and that Eve eventually finds the secret key exchanged between Alice and Bob with noticeable probability. The first result follows from Lemma 3.68, and we present the detailed proof subsequently. We then prove formally that Eve finds the secret key with noticeable probability.

Proof of Lemma 3.70. We will show that for every pair of Alice's/Bob's views $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ in the probability space $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ that satisfy the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, the following holds:

$$\Pr_{\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right] = \alpha\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\alpha_A\alpha_B,$$

where α_A depends only on Alice's view $V_A^{(i,j)}$, and α_B only depends on Bob's view $V_B^{(i,j)}$. Hence, if we let $\mathcal{A}$ be the distribution such that $Pr_{\mathcal{A}}[V_A^{(i,j)}]$ is proportional to α_A, and if we let $\mathcal{B}$ be the distribution such that $Pr_{\mathcal{B}}[V_B^{(i,j)}]$ is proportional to α_B, then $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is proportional (and hence equal to) the distribution $(\mathcal{A}\times\mathcal{B})|\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$.

Analysis Step-1. Note that the tuple $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ lies in the support of the probability space $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$,i.e. we have

$$\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \in \mathsf{SUPPORT}\left(\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right).$$

Hence, if the views of Alice and Bob are indeed $V_A^{(i,j)}$ and $V_B^{(i,j)}$ respectively, then the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ must hold. In other words, we have

$$\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right] = \\ \Pr_{\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right]\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right].$$

Also, by definition, we have

$$\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right] = \frac{\Pr_{\mathcal{E}}(V_A^{(i,j)}, V_B^{(i,j)}, \mathsf{m}^{(i,j)}, P_E^{(i,j)})}{\Pr_{\mathcal{E}}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}.$$

Hence, we have

$$\Pr_{\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right] = \\ \frac{\Pr_{\mathcal{E}}(V_A^{(i,j)}, V_B^{(i,j)}, \mathsf{m}^{(i,j)}, P_E^{(i,j)})}{\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right]\Pr_{\mathcal{E}}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}.$$

Analysis Step-2: Analyzing the Denominator. The denominator of the expression on the right hand side is a function of only $\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, and so, we can define the function

$$\beta\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) = \Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right]\Pr_{\mathcal{E}}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

In what follows, we analyze the numerator of the expression on the right hand side.

Analysis Step-3: Analyzing the Numerator. Let $P_A^{(i,j)}$ and $P_B^{(i,j)}$ be the oracle query-answer pairs in the views of Alice and Bob, namely $V_A^{(i,j)}$ and $V_B^{(i,j)}$, respectively. Then, we claim that the numerator is given by

$$\Pr_{\mathcal{E}}\left(V_A^{(i,j)}, V_B^{(i,j)}, \mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) = 2^{-|\mathsf{r}_A|}2^{-|\mathsf{r}_B|}\Pr_{\mathcal{E}}\left(P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right),$$

where r_A and r_B are the random strings used by Alice and Bob, respectively, $P_A^{(i,j)}$ and $P_B^{(i,j)}$ denote the set of query-response pairs in the views $V_A^{(i,j)}$ and $V_B^{(i,j)}$ of Alice and Bob, respectively, and $\Pr_{\mathcal{E}}\left(P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right)$ denotes the probability that during an execution $\mathcal{E} = (\mathsf{r}_A, \mathsf{r}_B, \mathbf{M})$, the random oracle $\mathbf{M}$ is consistent with the set of query-response pairs in the set $P_A^{(i,j)} \cup P_B^{(i,j)} \cup P^{(i,j)_E}$. We justify next why our claim is correct.

Observe that the necessary and sufficient condition that

$$V_A^{(i,j)} = \left(\mathsf{r}_A, \mathsf{m}^{(i,j)}, P_A^{(i,j)}\right), \quad V_B^{(i,j)} = \left(\mathsf{r}_B, \mathsf{m}^{(i,j)}, P_B^{(i,j)}\right),$$

only happens if we sample a uniformly random execution $(\mathsf{r}'_A, \mathsf{r}'_B, \mathbf{M})$ such that all of the following hold simultaneously:

- $\mathsf{r}'_A = \mathsf{r}_A$ (which happens with probability $2^{-|\mathsf{r}_A|}$), and
- $\mathsf{r}'_B = \mathsf{r}_B$ (which happens with probability $2^{-|\mathsf{r}_B|}$), and
- $\mathbf{M}$ is consistent with the set of query-response pairs in the set $\left(P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right)$ (we analyze this probability subsequently).

Note that all of these conditions holding simultaneously ensures that Alice and Bob will indeed produce the transcript of messages $\mathsf{m}^{(i,j)}$.

Analyzing the Consistency Probability. We now analyze the probability that $\mathbf{M}$ is consistent with the set of query-response pairs in the set $\left(P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right)$. More formally, we analyze the probability expression

$$\Pr_{\mathcal{E}}\left[P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right].$$

By the definition of event Good_1, Eve queries all intersection and equivalence queries, i.e., we have

$$P_{A\cap B}^{(i,j)} \subseteq P_E^{(i,j)}, \quad P_{A\equiv B}^{(i,j)} \subseteq P_E^{(i,j)}.$$

It now follows from the definition of the generic $(k+1)$-restricted SCMA oracle $\mathbf{M}$ that the responses of $\mathbf{M}$ corresponding to the queries in the sets $\mathcal{Q}\left(P_A^{(i,j)} \setminus P_E^{(i,j)}\right)$ and $\mathcal{Q}\left(P_A^{(i,j)} \setminus P_E^{(i,j)}\right)$ are uniformly random and *independent* of the query-response pairs in the set $P_E^{(i,j)}$, since:

- Let $q_1 \in \mathcal{Q}\left(P_A^{(i,j)} \setminus P_E^{(i,j)}\right)$ and $q_2 \in \mathcal{Q}\left(P_E^{(i,j)}\right)$. Then, q_1 and q_2 are neither identical nor equivalent, and hence, the responses of $\mathbf{M}$ on q_1 and q_2 are independent.
- Similarly, let $q'_1 \in \mathcal{Q}\left(P_B^{(i,j)} \setminus P_E^{(i,j)}\right)$ and $q'_2 \in \mathcal{Q}\left(P_E^{(i,j)}\right)$. Then, q'_1 and q'_2 are neither identical nor equivalent, and hence, the responses of $\mathbf{M}$ on q'_1 and q'_2 are independent.

- Finally, let $q_1'' \in \mathcal{Q}\left(P_A^{(i,j)} \setminus P_E^{(i,j)}\right)$ and $q_2'' \in \mathcal{Q}\left(P_B^{(i,j)} \setminus P_E^{(i,j)}\right)$. Then, q_1'' and q_2'' are neither identical nor equivalent, and hence, the responses of **M** on q_1'' and q_2'' are independent.

This in turn implies that we have

$$\Pr_{\mathcal{E}}\left[P_A^{(i,j)} \cup P_B^{(i,j)} \cup P_E^{(i,j)}\right] = \Pr_{\mathcal{E}}\left[P_E^{(i,j)}\right] \cdot \Pr_{\mathcal{E}}\left[P_A^{(i,j)} \setminus P_E^{(i,j)}\right] \Pr_{\mathcal{E}}\left[P_B^{(i,j)} \setminus P_E^{(i,j)}\right].$$

Analysis Step-4: Putting Everything Together. Finally, by setting

$$\alpha_A = 2^{-|\mathsf{r}_A|} \Pr_{\mathcal{E}}[P_A^{(i,j)} \setminus P_E^{(i,j)}], \quad \alpha_B = 2^{-|\mathsf{r}_B|} \Pr_{\mathcal{E}}[P_B^{(i,j)} \setminus P_E^{(i,j)}]$$

and by setting

$$\alpha\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) = \frac{\Pr_{\mathcal{E}}[P_E^{(i,j)}]}{\beta\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)},$$

we have

$$\Pr_{\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[V_A^{(i,j)}, V_B^{(i,j)}\right] = \alpha\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\alpha_A\alpha_B.$$

This completes the proof of Lemma 3.70.

Proof of Lemma 3.73. The proof of Lemma 3.73 follows from the proofs of the following claims:

Claim 3.74. *Let $G^{(i,j)} = (\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)}, E^{(i,j)})$ be the graph constructed as above. Then, for each vertex $u \in \mathcal{U}_A^{(i,j)}$ and each vertex $v \in \mathcal{U}_B^{(i,j)}$, we have*

$$\sum_{w \in \mathcal{U}_B^{(i,j)}, w \not\sim u} \deg(w) \leq \epsilon |E^{(i,j)}|, \qquad \sum_{w' \in \mathcal{U}_A^{(i,j)}, w' \not\sim v} \deg(w') \leq \epsilon |E^{(i,j)}|.$$

Claim 3.75. *Let $G^{(i,j)} = (\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)}, E^{(i,j)})$ be the graph constructed as above. For any vertex $w \in (\mathcal{U}_A^{(i,j)} \cup \mathcal{U}_B^{(i,j)})$, define the set of edges*

$$E^{\not\sim}(w) = \{(u,v) \in E^{(i,j)} : u \not\sim w \wedge v \not\sim w\},$$

to be the set of edges that are not adjacent to any immediate neighbors of the vertex w in $G^{(i,j)}$. Then, for any vertex $w \in (\mathcal{U}_A^{(i,j)} \cup \mathcal{U}_B^{(i,j)})$, we have

$$\left|E^{\not\sim}(w)\right| \leq \epsilon |E|.$$

Claim 3.76. *Let $G^{(i,j)} = (\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)}, E^{(i,j)})$ be any non-empty bipartite graph such that for each vertex $w \in (\mathcal{U}_A^{(i,j)} \cup \mathcal{U}_B^{(i,j)})$, we have $\left|E^{\not\sim}(w)\right| \leq \epsilon|E|$ for some $\epsilon < 1/2$. Then, for each vertex $u \in \mathcal{U}_A^{(i,j)}$ and each vertex $v \in \mathcal{U}_B^{(i,j)}$, we have*

$$\deg(u) \geq (1-2\epsilon)|\mathcal{U}_B^{(i,j)}|, \quad \deg(v) \geq (1-2\epsilon)|\mathcal{U}_A^{(i,j)}|.$$

Proof of Claim 3.74. The probability that we choose a vertex w when we choose a random edge from $E^{(i,j)}$ is given by $\frac{\deg(w)}{|E^{(i,j)}|}$. Now, suppose that for some vertex $u \in \mathcal{U}_A^{(i,j)}$, we have

$$\sum_{w \in \mathcal{U}_B^{(i,j)}, w \not\sim u} \deg(w) > \epsilon |E^{(i,j)}|.$$

Then we have

$$\Pr_{(u,w) \leftarrow E^{(i,j)}} [|Q_u \cap Q_w| \neq \phi] > \epsilon.$$

Suppose that Alice issues at most n_A $(k+1)$-restricted SCMA oracle queries, i.e., we have $|Q_u| \leq n_A$. Hence, by the pigeonhole principle, there must exist $q \in Q_u$ such that

$$\Pr[q \in Q_v] > \epsilon / n_A.$$

But this is a contradiction, since then, by the definition of the attacker Eve, q must be in the set of queries corresponding to $P_E^{(i,j)}$, and hence, by definition, cannot be in the set Q_u. Hence, for each vertex $u \in \mathcal{U}_A^{(i,j)}$, we must have

$$\sum_{w \in \mathcal{U}_B^{(i,j)}, w \not\sim u} \deg(w) \leq \epsilon |E^{(i,j)}|.$$

By a similar argument, it follows that for any vertex $v \in \mathcal{U}_B^{(i,j)}$, we must have

$$\sum_{w' \in \mathcal{U}_A^{(i,j)}, w' \not\sim v} \deg(w') \leq \epsilon |E^{(i,j)}|.$$

This completes the proof of Claim 3.74.

Proof of Claim 3.75. Let $w \in (\mathcal{U}_A^{(i,j)} \cup \mathcal{U}_B^{(i,j)})$ be any vertex. Suppose that $w \in \mathcal{U}_A^{(i,j)}$. Then, we have

$$\left| E^{\not\sim}(w) \right| = \sum_{w' \in \mathcal{U}_B^{(i,j)}, w' \not\sim w} \deg(w') \leq \epsilon |E|.$$

Alternatively, suppose that $w \in \mathcal{U}_B^{(i,j)}$. Then, we have

$$\left| E^{\not\sim}(w) \right| = \sum_{w'' \in \mathcal{U}_A^{(i,j)}, w'' \not\sim w} \deg(w'') \leq \epsilon |E|.$$

This completes the proof of Claim 3.75.

Proof of Claim 3.76. To begin with, we define

$$\deg_A = \min\{\deg(u) : u \in \mathcal{U}_A^{(i,j)}\}, \quad \deg_B = \min\{\deg(v) : v \in \mathcal{U}_B^{(i,j)}\}.$$

Assume w.l.o.g. that

$$\frac{\deg_A}{|\mathcal{U}_B^{(i,j)}|} \leq \frac{\deg_B}{|\mathcal{U}_A^{(i,j)}|}.$$

Hence, it suffices to prove that $\frac{\deg_A}{|\mathcal{U}_B^{(i,j)}|} \geq (1-2\epsilon)$. Suppose that $\frac{\deg_A}{|\mathcal{U}_B^{(i,j)}|} < (1-2\epsilon)$, and let $u \in \mathcal{U}_A^{(i,j)}$ be the vertex such that $\deg(u) = \deg_A < (1-2\epsilon)|\mathcal{U}_B^{(i,j)}|$. Since for each $v \in \mathcal{U}_B^{(i,j)}$, we have $\deg(v) \leq |\mathcal{U}_A^{(i,j)}|$, we must have

$$|E^{(i,j)} \setminus E^{\not\sim}(u)| \leq \deg(u)|\mathcal{U}_A^{(i,j)}| = \deg_A |\mathcal{U}_A^{(i,j)}| \leq \deg_B |\mathcal{U}_B^{(i,j)}|.$$

On the other hand, since $\deg(u) < (1-2\epsilon)|\mathcal{U}_B^{(i,j)}|$, we must have

$$|E^{\not\sim}(u)| > 2\epsilon \deg_B |\mathcal{U}_B^{(i,j)}| \geq 2\epsilon |E^{(i,j)} \setminus E^{\not\sim}(u)|.$$

Now, we have

$$|E^{\not\sim}(u)| \leq \epsilon |E^{(i,j)}| = \epsilon \left(|E^{\not\sim}(u)| + |E^{(i,j)} \setminus E^{\not\sim}(u)| \right) < (\epsilon + 1/2)|E^{\not\sim}(u)|,$$

which is a contradiction for any $\epsilon < 1/2$ because the graph $G^{(i,j)}$ is non-empty. This completes the proof of Claim 3.76.

Finishing the Proof of Lemma 3.69. To finish the proof of Lemma 3.69, we first define an auxiliary fail event $\mathsf{Fail}'^{i,j}$ to be the event that the query (made by Alice or Bob) to the $(k+1)$-restricted SCMA oracle after this sub-round is an intersection query but is not contained in $P_E^{(i,j)}$. It is easy to see that, given Lemma 3.65, for any sub-round (i,j) of the KE protocol with equivalence complete query pattern, we have

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}^{(i,j)} | \mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \leq \Pr_{\mathcal{E}}\left[\mathsf{Fail}'^{(i,j)} | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right].$$

This follows immediately from the fact that the first time Eve fails to find an intersection query is also the first time Eve fails to find an intersection query or an equivalence query (since each equivalence query if preceded by a corresponding intersection query), and that the event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ holds if and only if the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ holds. Hence, to prove Lemma 3.69, it suffices to show that for any sub-round (i,j) of the KE protocol with equivalence complete query pattern,

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}'^{(i,j)} | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] = O\left(\frac{\epsilon}{(n_A + n_B)}\right).$$

Again, given any KE protocol with equivalence complete query pattern as described above, let (i,j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i,j), and let $P_E^{(i,j)}$ denote the set of $(k+1)$-restricted SCMA oracle query-answer pairs until sub-round (i,j) asked by Eve, such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. Assume without loss of generality that Bob issues a query q in sub-round (i,j), and let $V_B^{(i,j)}$ denote Bob's view up until sub-round (i,j). Now observe the following:

- By Lemma 3.71, the distribution $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ conditioned on getting $V_B^{(i,j)}$ as Bob's view is the same as the product distribution $(\mathcal{A} \times \mathcal{B})$ conditioned on the events $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ *and* getting $V_B^{(i,j)}$ as Bob's view, simultaneously. By the graph characterization of $\mathcal{GV}_0$ (Lemma 3.73), letting $G^{(i,j)} = (\mathcal{U}_A^{(i,j)}, \mathcal{U}_B^{(i,j)}, E^{(i,j)})$ be the graph constructed above, this is the same as randomly choosing an edge $(u,v) \leftarrow E^{(i,j)}$ conditioned on getting $V_B^{(i,j)}$ as Bob's view, and then choosing (A_u, B_v).

- It then follows that, conditioned on v such that $B_v = V_B^{(i,j)}$, the distribution of Alice's view is the same as choosing $u \leftarrow N(v)$ to be a random neighbor of v (here $N(v)$ denotes the set of all immediate neighbors of v), and then choosing A_u. Define the set S as:

  $$S = \{u \in \mathcal{U}_A^{(i,j)} : q \in A_u\}.$$

  Then we have the following:

  $$\Pr_{u \leftarrow N(v)}[q \in A_u] \leq \frac{|S|}{\deg(v)} \leq \frac{|S|}{(1-2\epsilon)|\mathcal{U}_A^{(i,j)}|} \leq \frac{|S||\mathcal{U}_B^{(i,j)}|}{(1-2\epsilon)|E^{(i,j)}|} \leq \frac{\sum_{u \in S} \deg(u)}{(1-2\epsilon)^2|E^{(i,j)}|}$$

  The second and fourth inequalities are because of Lemma 3.73. The third one is because $|E^{(i,j)}| \leq \left|\mathcal{U}_A^{(i,j)}\right| \left|\mathcal{U}_B^{(i,j)}\right|$.

- By the definition of the attack algorithm of Eve, the only queries asked by Eve are queries with probability of occurrence (in Bob's view) greater than ϵ/n_B. Hence, we must have

  $$\frac{\sum_{u \in S} \deg(u)}{|E^{(i,j)}|} \leq \frac{\epsilon}{n_B},$$

  which in turn implies that we have

  $$\Pr_{u \leftarrow N(v)}[q \in A_u] \leq \frac{\epsilon}{(1-2\epsilon)^2 n_B},$$

  which is $O\left(\frac{\epsilon}{n_B}\right)$ for $\epsilon < 1/10$.

Thus, we have

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}'^{(i,j)} | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) \wedge \mathsf{B}\right] = O\left(\frac{\epsilon}{n_B}\right),$$

where B denotes the event that Bob issues the query in sub-round (i,j). Similarly, an analogous argument can be used to prove that

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}'^{(i,j)} | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) \wedge \mathsf{A}\right] = O\left(\frac{\epsilon}{n_A}\right),$$

where A denotes the event that Alice issues the query in sub-round (i,j). Hence, we have

$$\begin{aligned}\Pr_{\mathcal{E}}\left[\mathsf{Fail}'^{(i,j)} | \mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] &= O\left(\frac{\epsilon}{n_A} \cdot \frac{n_A}{n_A + n_B} + \frac{\epsilon}{n_B} \cdot \frac{\epsilon}{n_A + n_B}\right) \\ &= O\left(\frac{\epsilon}{n_A + n_B}\right).\end{aligned}$$

This completes the proof of Lemma 3.69, and hence, the proof of Lemma 3.67.

3.2.6 Proof of Lemma 3.68: The Attack is Efficient

We now present the proof of Lemma 3.68, which establishes that the attack is efficient.

Proof Overview. We follow a strategy similar to [BM09] to prove that the attack is efficient by crucially relying on the fact that the attack is successful. Recall that in her algorithm, Eve follows the following strategy: at any given sub-round of the protocol, Eve keeps making the lexicographically first query q that has "significant" probability of appearing in either Alice's query set or Bob's query set, until all such queries are exhausted. Also recall that this probability is based on the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ (where $\mathsf{m}^{(i,j)}$ denotes the set of messages exchanged between Alice and Bob until sub-round (i,j), and $P_E^{(i,j)}$ denotes the set of $(k+1)$-restricted SCMA oracle query-answer pairs until sub-round (i,j) asked by Eve), conditioned on the event that Eve has not missed any intersection or equivalence queries up until this point (i.e. the event Good_1). Now, since we have proven that the event Good_1 happens with high probability (Lemma 3.65), this implies that queries with a significant probability of occurrence according the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ conditioned on Good_1 also have a significant probability of occurrence under the real distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$. Intuitively, we use this to bound the number of queries that Eve has to make by arguing that each query that Eve makes decreases the (nonzero) expected number of unknown queries. The formal proof is detailed below.

A Bad Event. For the formal proof, we begin by defining an additional event, which we refer to as a "bad" event. Let (i,j) denote some sub-round of the KE protocol, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of messages between Alice and Bob until sub-round (i,j), and let $P_E^{(i,j)}$ denote some sequence of $(k+1)$-restricted SCMA oracle query-answer pairs until sub-round (i,j) learned by Eve. We use $\mathsf{Bad}^{(i,j)}$ to denote the event that

$$\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\neg\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] > \frac{1}{2}.$$

We also define the probability space $\widehat{\mathcal{E}}$ to denote the same execution probability space as $\mathcal{E}$ with the difference that for any sub-round (i,j), Eve stops asking more queries at sub-round (i,j) if the event $\mathsf{Bad}^{(i,j)}$ occurs (the behavior of Alice and Bob remains unchanged). Note that $\mathcal{E}$ and $\widehat{\mathcal{E}}$ are identical as long as $\mathsf{Bad}^{(i,j)}$ does not happen, and so we have

$$\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}].$$

More generally speaking, for any event D whose definition depends on the behavior of Eve, we have

$$\Pr_{\mathcal{E}}[\mathsf{Bad} \vee \mathsf{D}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad} \vee \mathsf{D}].$$

The proof of Lemma 3.68 follows from the following steps:

- **Step-1:** We first show the following:

  $$\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon) \implies \Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon).$$

  Since our analysis of the success probability of the attack already established that $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$ (Lemma 3.67), we have

  $$\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon).$$

- **Step-2:** We then show the following: $\Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}] = O(\epsilon)$.

Observe that

$$\Pr_{\mathcal{E}}[\mathsf{Long}] \leq \Pr_{\mathcal{E}}[\mathsf{Long} \vee \mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Long} \vee \mathsf{Bad}] \leq \Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}] + \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}].$$

Hence, we have $\Pr_{\mathcal{E}}[\mathsf{Long}] = O(\epsilon)$, which is precisely the statement of Lemma 3.68.

Step-1: Bounding $\Pr_{\mathcal{E}}[\mathsf{Bad}]$. We state and prove the following lemma.

Lemma 3.77. *If* $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$ *then we must have* $\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon)$.

Proof. We present a proof by contradiction, which follows closely the proof of Lemma 6.4 in [IR89] and the proof of Lemma 4.7 in [BM09]. We present the proof in the context of our attack for the sake of completeness.

Assume that $\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Omega(\epsilon)$. We will show that this implies $\Pr_{\mathcal{E}}[\mathsf{Fail}] = \Omega(\epsilon)$. When we run the attack, instead of sampling the whole randomness $(r_A, r_B, \mathbf{M}) \leftarrow \mathcal{E}$ (for Alice, Bob, and the oracle) at the beginning, we can choose some parts of the system first (according to their final distribution), and then choose the rest of the system from their distribution conditioned on the chosen parts (this can be viewed as a generalization of the popular "lazy oracle sampling" method). In particular, we proceed as follows:

- Run the execution of the key exchange protocol as well as the attack algorithm for Eve till an arbitrary sub-round (i, j) such that $\mathsf{m}^{(i,j)}$ is the set of messages exchanged between Alice and Bob until sub-round (i, j), and $P_E^{(i,j)}$ is the set of $(k+1)$-restricted SCMA oracle query-answer pairs until sub-round (i, j) asked by Eve. Pretend that at this point, we have sampled $\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, and the rest of the description of the execution is not chosen yet.
- Sample $\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, and set $V_A^{(i,j)}$ and $V_B^{(i,j)}$ to be the "real" views of Alice and Bob until sub-round (i, j).
- Continue running the execution of the key exchange protocol as well as the attack algorithm for Eve from this point onwards conditioned on $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ (the views of Alice and Bob so far), and $\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ (the view of Eve so far).

Observe that the choice of (i, j) in the aforementioned simulation can be chosen arbitrarily. In particular, we could set it to the particular sub-round (i, j) where the event Bad happens for the first time. If the event Bad never happens, then we sample the views of Alice and Bob at the very end of the protocol execution. Now recall that Bad happens when

$$\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\neg\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] < \frac{1}{2}.$$

Since $\neg\mathsf{Good}_1 \subset \mathsf{Fail}$, we have

$$\Pr_{\mathcal{E}}[\mathsf{Fail} | \neg\mathsf{Good}_1] = 1.$$

So if Bad happens for the first time at sub-round (i, j), and we choose the views of Alice and Bob from $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, it must be the case that Fail will hold for this particular execution of the system with probability at least $1/2$. So we have

$$\Pr_{\mathcal{E}}[\mathsf{Fail}] \geq \frac{1}{2}\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Omega(\epsilon),$$

as desired. This completes the proof of Lemma 3.77. □

Since our analysis of the success probability of the attack already established that $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$ (Lemma 3.67), we have the following corollary.

Corollary 3.78. *We have* $\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon)$.

Step-2: Bounding $\Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}]$**.** We state and prove the following lemma.

Lemma 3.79. *We have* $\Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}] = O(\epsilon)$.

Proof. We prove that the expected number of queries asked by Eve in an execution sampled from the distribution $\widehat{\mathcal{E}}$ is $O(n_A n_B/\epsilon)$. Our proof follows closely the proof of Lemma 4.8 in [BM09]. We present the proof in the context of our attack for the sake of completeness.

By definition, in any sub-round (i, j), as long as there is a query $q = (s, x)$ for $s \in \Sigma^{k+1}$ and x such that $\mathsf{Level}(x) \neq -1$ such that

$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}[q \in \mathcal{Q}(V_A^{(i,j)})] > \frac{\epsilon}{n_B},$$

or

$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}[q \in \mathcal{Q}(V_B^{(i,j)})] > \frac{\epsilon}{n_A},$$

Eve issues the lexicographically first such query q to the $(k + 1)$-restricted SCMA oracle and adds the query-response pair $(q, \mathbf{M}(q))$ to $P_E^{(i,j)}$. Also, as long as Eve does not stop asking queries, we have

$$\Pr_{\widehat{\mathcal{E}}}\left[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] > \frac{1}{2}.$$

Hence, if Eve asks a query q in sub-round (i, j) conditioned on $\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, we must have

$$\begin{aligned}
&\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \widehat{\mathcal{V}}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[q \in \mathcal{Q}(V_A^{(i,j)}) \cup \mathcal{Q}(V_B^{(i,j)})\right] \\
&\geq \Pr_{\widehat{\mathcal{E}}}\left[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \cdot \\
&\qquad \Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \widehat{\mathcal{GV}_1}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[q \in \mathcal{Q}(V_A^{(i,j)}) \cup \mathcal{Q}(V_B^{(i,j)})\right] \\
&= \Omega\left(\frac{\epsilon(n_A + n_B)}{n_A n_B}\right),
\end{aligned}$$

where $\widehat{\mathcal{V}}$ and $\widehat{\mathcal{GV}_1}$ are defined analogously to $\mathcal{V}$ and $\mathcal{GV}_1$, albeit with respect to the modified probability distribution $\widehat{\mathcal{E}}$.

Now, define the random variable Y_ℓ to be 1 if Eve asks at least ℓ queries and the ℓ-th query that she makes was asked before by either Alice or Bob. It is easy to see that $\sum_\ell Y_\ell \leq (n_A + n_B)$ since Alice and Bob make at most n_A and n_B queries, respectively. Hence,

$$\sum_\ell \mathbb{E}(Y_\ell) = \mathbb{E}\left(\sum_\ell Y_\ell\right) \leq (n_A + n_B).$$

Claim 3.80. *Let p_ℓ be the probability that Eve asks the ℓ-th query. Then we have*

$$p_\ell = O\left(\frac{n_A n_B \, \mathbb{E}(Y_\ell)}{\epsilon(n_A + n_B)}\right).$$

Since $\sum_\ell p_\ell$ is the expected number of queries asked by Eve, assuming the aforementioned claim is true, we have

$$\sum_\ell p_\ell = O\left(\frac{n_A n_B \sum_\ell \mathbb{E}(Y_\ell)}{\epsilon(n_A + n_B)}\right) = O\left(\frac{n_A n_B}{\epsilon}\right),$$

which proves Lemma 3.79. Hence, it only remains to prove the above claim.

Proof of Claim. Define the random variable Y_ℓ^q to be 1 if the ℓ-th query that Eve asks is q and q was asked before by either Alice or Bob. Then $\mathbb{E}[Y_\ell] = \sum_q \mathbb{E}[Y_\ell^q]$. Suppose that the ℓ-th query was issued in the (i,j)-th sub-round. We have

$$\begin{aligned}
\mathbb{E}[Y_\ell^q] &= \sum_{\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)} \Pr\left[V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \cdot \\
&\qquad \Pr\left[q \in Q_A^{(i,j)} \cup Q_B^{(i,j)} | V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \\
&= \gamma \sum_{\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)} \Pr\left[V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right],
\end{aligned}$$

where $\gamma = \Omega\left(\frac{\epsilon(n_A+n_B)}{n_A n_B}\right)$. Hence, we have

$$\begin{aligned}
\mathbb{E}[Y_\ell] &= \sum_q \mathbb{E}[Y_\ell^q] \\
&= \gamma \cdot \sum_{\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)} \Pr\left[\text{Eve queries some } q \text{ as its } \ell\text{-th query} | V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \cdot \\
&\qquad \Pr\left[V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] \\
&= \gamma p_\ell \\
&= \Omega\left(\frac{p_j \epsilon(n_A + n_B)}{n_A n_B}\right).
\end{aligned}$$

which in turn implies that

$$p_j = O\left(\frac{n_A n_B \, \mathbb{E}[Y_\ell]}{\epsilon(n_A + n_B)}\right), \qquad \square$$

as desired. This completes the proof of our claim and, hence, the proof of Lemma 3.79.

Finally, together with Lemma 3.77 and Corollary 3.78, the proof of Lemma 3.79 completes the proof of Lemma 3.68.

3.2.7 Finishing the Attack: Eve finds the Key

Finally, we formally prove that Eve actually finds the secret key exchanged by Alice and Bob. The proof is very similar to the proof of Theorem 6.2 in [IR89] and the proof of Theorem 5.2 in [BM09]. We present the proof in the context of our attack on any $2k$-round KE with equivalence complete query pattern for the sake of completeness.

We assume in the last round of the $2k$-round KE with equivalence complete query pattern, Alice sends a special message LAST to Bob. Let the random variables $V_A^{(2k)}$, $V_B^{(2k)}$ and $V_E^{(2k)}$ be the distributions of the views of Alice, Bob, and Eve at the end of the execution, where

$$V_E^{(2k)} = \left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right).$$

In order to find the secret Eve runs the attack of Section 3.2.4 and at the end of round $2k$ (when Alice has sent the message LAST to Bob, and Eve has asked her queries from the oracle), Eve samples

$$\left(\widehat{V}_A^{(2k)}, \widehat{V}_B^{(2k)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right),$$

computes Alice's final output $s_A = s\left(\widehat{V}_A^{(2k)}\right)$, and outputs $s_E = s_A$ as its own output. We need to prove that

$$\Pr[s_E = s_B] > \rho - \delta,$$

for some $\delta = O(\epsilon)$. Let $\widehat{V}$ be the random variable generated by sampling

$$\left(\widehat{V}_A^{(2k)}, \widehat{V}_B^{(2k)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right),$$

and choosing $\widehat{V}_A^{(2k)}$ from it. We will show that

$$\mathsf{SD}\left(\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right), \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right) = O(\epsilon),$$

which in turn implies that

$$\left|\Pr\left[s\left(V_A^{(2k)}\right) = s\left(V_B^{(2k)}\right)\right] - \Pr\left[s\left(\widehat{V}\right) = s\left(V_B^{(2k)}\right)\right]\right| = O(\epsilon).$$

For any triple of the form (V_A, V_B, V_E), we say that:

- the event $\mathsf{Good}_0\left(V_A, V_B, V_E\right)$ holds if $\mathcal{Q}\left(V_A\right)$ and $\mathcal{Q}\left(V_B\right)$ have no intersection query that does not also appear in V_E, and
- the event $\mathsf{Good}_1\left(V_A, V_B, V_E\right)$ holds if $\mathcal{Q}\left(V_A\right)$ and $\mathcal{Q}\left(V_B\right)$ have no intersection query that does not appear in V_E *and* no equivalence query-pair such that V_E does not have a corresponding query equivalent to this pair.

The proof of the fact that Eve finds the key now follows from the following claims.

Claim 3.81. *We claim that* $\Pr[\neg\mathsf{Good}_1\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon)$.

Proof. The proof of this claim follows immediately from the proof of Theorem 3.62. □

Claim 3.82. *We claim that* $\Pr[\neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon)$.

Proof. We argue this claim as follows. It follows from Lemmas 3.59 and 3.60 that for any $2k$-round KE protocol with equivalence complete query pattern,

$$\Pr\left[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) \middle| \neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right] = 1,$$

and hence

$$\Pr\left[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right] = \Pr\left[\neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right].$$

Now suppose we fix $V_E^{(2k)} = \left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right)$ and sample $\widehat{V}$ as above. Then $\widehat{V}$ is independent of $V_B^{(2k)}$, and hence, any query q such that $q \in \mathcal{Q}\left(V_B^{(2k)}\right)$ and $q \notin \mathcal{Q}\left(V_E^{(2k)}\right)$ has probability at most ϵ/n_B of appearing in $\mathcal{Q}\left(\widehat{V}\right)$ (this follows from Eve's strategy of choosing queries in the attack). Hence, we must have

$$\Pr[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon),$$

which completes the proof of this claim. □

Claim 3.83. *Finally, we claim that*

$$\mathsf{SD}\Big(\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) | \mathsf{Good}_1\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right), \\ \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) | \mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\Big) = O(\epsilon).$$

Proof. We argue this claim based on Lemma 3.73. Let $G^{(2k)} = (\mathcal{U}_A^{(2k)}, \mathcal{U}_B^{(2k)}, E^{(2k)})$ be the graph characterization of $\mathcal{GV}_0\left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right)$. Then we have the following:

- The distribution of $V_A^{(2k)}$ in $\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right)$ conditioned on the event $\mathsf{Good}_0\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right)$ is the same as A_u sampled as follows: choose a vertex $v \in \mathcal{U}_B^{(2k)}$ conditioned on $B_v = V_B^{(2k)}$, then choose a uniformly random neighbor of v as $u \leftarrow N(v)$, and output A_u.
- Similarly, the distribution of $\widehat{V}$ in $\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)$ conditioned on the event $\mathsf{Good}_0\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right)$ is the same as A_u sampled as follows: choose a vertex $v \in \mathcal{U}_B^{(2k)}$ conditioned on $B_v = V_B^{(2k)}$, then choose a random edge $(u, v') \leftarrow E^{(2k)}$ conditioned on $v' = v$, and then output A_u (this is the same as randomly choosing neighbor of v as $u \in N(v)$ such that the choosing probability is proportional to $\deg(u)$, and then outputting A_u).
- By Lemma 3.73, we have for each $u \in \mathcal{U}_A^{(2k)}$

  $$(1 - 2\epsilon)\left|V_B^{(2k)}\right| \leq \deg(u) \leq \left|V_B^{(2k)}\right|,$$

  and hence, since $\epsilon < 1/10$, using techniques similar to those used in the proof of Theorem 5.2 in [BM09], one can show that

  $$\mathsf{SD}\Big(\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) | \mathsf{Good}_0\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right), \\ \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) | \mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\Big) \leq 2\epsilon.$$

Finally, it again follows from Lemmas 3.59 and 3.60 that for any $2k$-round KE protocol with equivalence complete query pattern,

$$\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_1 \left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) = \left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_0 \left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right),$$

and

$$\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_1 \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) = \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_0 \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right),$$

and hence, we have

$$\mathsf{SD}\Big(\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_1 \left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right), \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) |\mathsf{Good}_1 \left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) \Big) \leq 2\epsilon.$$

This completes the proof of the claim, and hence the proof of successful key-recovery by Eve. □

3.3 Separating $(2k-1)$-round Key Exchange from $2k$-round Key Exchange

In this section, we argue that we can also black-box separate $(2k-1)$-round key exchange from $2k$-round key exchange. The argument is almost identical to the separation of $2k$-round Key Exchange from $(2k+1)$-round key exchange, with the exception of some minor tweaks to the $(k+1)$-commutator property of a $(k+1)$-restricted SCMA oracle, and our core argument that for any KE protocol with equivalence complete query pattern, each equivalence query is also essentially an intersection query. The rest of the proof structure as well as the arguments surrounding attack success (detailed in Section 3.2.5, and Sections 3.2.5, 3.2.5, and 3.2.5), attack efficiency (detailed in Section 3.2.6), and the final key-finding probability (detailed in Section 3.2.7) remain essentially unchanged.

Changing the k-Commutator Property Slightly. For $k \geq 1$, suppose that we tweak the k-commutator property of a $(k+1)$-commutator oracle $\mathbf{M}(\cdot, \cdot)$ slightly as follows: instead of requiring that $\mathbf{M}((ab)^{k+1}, x_0) = \mathbf{M}((ba)^{k+1}, x_0)$ (x_0 being the base set element), we now require that

$$\mathbf{M}(b\|(ab)^k, x_0) = \mathbf{M}(a\|(ba)^k, x_0)$$

It is easy to see that in this case, a $(k+1)$-commutator oracle implies a $2k$-round key exchange as follows:

- Given a base element x_0, Alice would sample some $a \in M$ and obtain $\mathbf{M}(a, x_0)$, while Bob would sample some $b \in M$ and obtain $\mathbf{M}(b, x_0)$. Alice and Bob would then exchange their first-round messages, where Alice sends $\mathbf{M}(a, x_0)$ to Bob and Bob sends $\mathbf{M}(b, x_0)$ to Alice.
- In the next round, Alice would obtain $\mathbf{M}(ab, x_0) = \mathbf{M}\left(a, \mathbf{M}\left(b, x_0\right)\right)$, and Bob would obtain $\mathbf{M}(ba, x_0) = \mathbf{M}\left(b, \mathbf{M}\left(a, x_0\right)\right)$. Alice and Bob would then exchange their second-round messages, where Alice sends $\mathbf{M}(ab, x_0)$ to Bob and Bob sends $\mathbf{M}(ba, x_0)$ to Alice.

Observe that by repeating this process for $2k$ rounds and asking a final query to the $(k+1)$-SCMA oracle, Alice and Bob would have obtained $\mathbf{M}(a\| \, (ba)^k \, , x_0) = \mathbf{M}(b\| \, (ab)^k \, , x_0)$, which they can use as the final secret key. Note that this computation requires the full $2k$ rounds[1].

Arguing Impossibility of $(2k-1)$-round Key Exchange. Now let's look at what happens if Alice and Bob try to exploit the "commutative" property of the $(k+1)$-SCMA oracle in less than $2k$ rounds. Again, they must generate some equivalence query-pair of the form $\mathbf{M}(a\|(ba)^k, x_0) = \mathbf{M}(b\|(ab)^k, x_0)$ with less than $2k$ rounds of communication. Once again, note that when "building up" to such an equivalence query that gives Alice and Bob the same final set element via two different query sequences in less than $2k$ rounds, Alice and Bob cannot only issue queries to the $(k+1)$-SCMA where the monoid element is either a or b like in the $2k$-round key exchange protocol outlined above. In particular, by the pigeonhole principle, at least one of Alice or Bob must compute a query involving both the elements a and b.

At this point, we can the same core argument as in the separation of $2k$-round key exchange from $(2k+1)$-round key exchange to establish that even in this case, as long as the $(2k-1)$-round key exchange protocol is in a special form that "forces" Alice and Bob to make all "split" versions of their queries and at least one of Alice or Bob to compute all possible ways of computing an equivalence query as soon as there is a "trigger" query where the monoid element is a substring of either $(ab)^k$ or $(ba)^k$, any equivalence query w.r.t. the $(k+1)$-SCMA oracle that can be computed within $(2k-1)$ rounds is also an intersection query.

This again effectively reduces all equivalence queries that rely on the (modified) commutative property of the $(k+1)$-SCMA oracle to the "traditional" notion of intersection queries, and we can again handle such queries using the [BM09] framework, as detailed in Section 3.2.

4 Analyzing Malicious Two-Party Computation by Rounds

In this section, we present the formal details of our main novel black-box separation result, namely separating maliciously secure two-party computation (2-PC) by rounds. We begin with the formal details of our proof that maliciously (abort) secure 2-PC is equivalent to a monoid action that have certain commutator-like properties and satisfy certain hardness assumptions. We then describe formally how we can use the above structural characterization of 2-PC to separate 2-PC by rounds.

4.1 Two-Party Computation and Commutative Monoid Action

In this section, we prove that *any* ℓ-round (two-party) computation protocol (for deterministic functions) is equivalent to an ℓ-distributional commutative monoid action equipped with certain additional structural properties and a stronger security notion as compared to the distributional unpredictability security notion satisfied by any ℓ-DUCMA. We refer to this specially structured ℓ-DUCMA with stronger security notions as an ℓ-DCMA$_{\text{2-PC}}$. Before defining ℓ-DCMA$_{\text{2-PC}}$, we first formally define an ℓ-round 2-PC protocol. For simplicity, we first focus on 2-PC protocols for symmetric functionalities (i.e., where both parties receive the same output). Subsequently, in Section 4.4, we show a generalization of our approach to the case of 2-PC protocols for asymmetric functionalities (i.e., where both parties receive potentially different outputs).

We note that the proofs in this section are very similar if not almost identical to those for key exchange, so we frequently defer details to that section.

[1] We again note that if M is a countably infinite set, then a uniform distribution over M is not well-defined; in this case, we restrict to those distributions for which the set of all strings consisting of more than $2k$ elements has negligible density in the sample space.

Defining an ℓ-Round 2-PC Protocol. We now define an ℓ-round 2-PC protocol for $\ell \geq 1$. In the same vein as our KE definition, we define ℓ-round 2-PC as a two-party protocol involving a pair of (non-uniform) probabilistic polynomial-time algorithms $A = \{\mathsf{A}_i\}_{i\in[0,\ell]}$ and $B = \{\mathsf{B}_i\}_{i\in[0,\ell]}$, where each individual algorithm A_i and B_i is formalized subsequently.

Before presenting the definition, we fix some notation. Let I denote the set of all possible inputs for parties A and B in a 2-PC protocol, and let F denote the set of all possible functions f computable by the protocol. Finally, let R_A and R_B denote the set of all possible random coins used by parties A and B.

Definition 4.1 (ℓ-Round 2-PC). An ℓ-round 2-PC protocol is a tuple of probabilistic polynomial-time algorithms $\Pi = \left(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\right)$ defined as follows:

- Setup takes as input a security parameter λ and output the public parameters pp.

- For each $i \in [0, \ell-1]$, A_i takes as input the public parameters pp, the private input $\mathsf{in}_A \in I$, the function $f \in F$, a secret state $r_{i,A} \in R_A$, and a transcript τ_i of the messages exchanged between parties A and B up until round-i, and outputs an updated secret state $r_{i+1,A}$ and a message $s_{i+1,A}$.

- For each $i \in [0, \ell-1]$, B_i takes as input the public parameters pp, the private input $\mathsf{in}_B \in I$, the function $f \in F$, a secret state $r_{i,B}$, and a transcript τ_i of the messages exchanged between parties A and B up until round-i, and outputs an updated secret state $r_{i+1,B}$ and a message $s_{i+1,B}$.

- A_ℓ takes as input the public parameters pp, the private input $\mathsf{in}_A \in I$, the function $f \in F$, a secret state $r_{\ell,A}$, and a transcript τ_ℓ of the messages exchanged between parties A and B up until round-ℓ, and outputs the "final" output y_{AB}.

- B_ℓ takes as input the public parameters pp, the private input $\mathsf{in}_B \in I$, the function $f \in F$, a secret state $r_{\ell,B}$, and a transcript τ_ℓ of the messages exchanged between parties A and B up until round-ℓ, and outputs the "final" output y_{BA}.

Correctness. An ℓ-round 2-PC protocol $\Pi = \left(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\right)$ is said to be correct if for any $\mathsf{pp} \leftarrow \mathsf{Setup}$, any pair of inputs $\mathsf{in}_A, \mathsf{in}_B \in I$, any function $f \in F$, and any

$$(r_{i+1,A}, s_{i+1,A}) = \mathsf{A}_i(\mathsf{pp}, \mathsf{in}_A, f, r_{i,A}, \tau_i), \quad (r_{i+1,B}, s_{i+1,B}) = \mathsf{B}_i(\mathsf{pp}, \mathsf{in}_A, f, r_{i,B}, \tau_i),$$

for each $i \in [0, \ell-1]$, we have

$$y_{AB} = y_{BA} = f(\mathsf{in}_A, \mathsf{in}_B),$$

where $y_{AB} = \mathsf{A}_\ell(\mathsf{pp}, \mathsf{in}_A, f, r_{\ell,A}, \tau_\ell)$ and $y_{BA} = \mathsf{B}_\ell(\mathsf{pp}, \mathsf{in}_A, f, r_{\ell,A}, \tau_\ell)$, and where for each $i \in [0, \ell]$, the transcript τ_i is defined as:

$$\tau_i = (\mathsf{pp}, f, s_{1,A}, s_{1,B}, s_{2,A}, s_{2,B}, \ldots, s_{i,A}, s_{i,B}).$$

Semi-Honest Security. An ℓ-round 2-PC protocol $\Pi = \left(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\right)$ is said to be computationally secure against static semi-honest adversaries if there exist PPT simulators $\mathcal{S}_A$ and $\mathcal{S}_B$ such that for any security parameter $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}$, any pair of inputs $\mathsf{in}_A, \mathsf{in}_B \in I$, any function $f \in F$, we have

$$\mathcal{S}_A\left(1^\lambda, \mathsf{pp}, \mathsf{in}_A, f(\mathsf{in}_A, \mathsf{in}_B)\right) \overset{c}{\approx} \left(V_A^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B), \mathsf{out}_A^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B)\right),$$

$$\mathcal{S}_B\left(1^\lambda, \mathsf{pp}, \mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B)\right) \overset{c}{\approx} \left(V_B^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B), \mathsf{out}_B^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B)\right),$$

where V_A^Π (resp., V_B^Π) denotes the view of party A (resp., party B) and out_A^Π (resp., out_A^Π) denotes the output of protocol Π for party A (resp., party B), with the views of parties A and B being defined as

$$V_A^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B) = \left(\{r_{i,A}\}_{i\in[\ell]}, \tau_\ell\right), \quad V_B^\Pi(1^\lambda, \mathsf{pp}, \mathsf{in}_A, \mathsf{in}_B) = \left(\{r_{i,B}\}_{i\in[\ell]}, \tau_\ell\right).$$

Malicious Security. An ℓ-round 2-PC protocol $\Pi = \left(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\right)$ is said to be computationally secure against static malicious adversaries if for any PPT static malicious adversary $\mathcal{A}$ corrupting party B (without loss of generality), there exists a PPT simulator $\mathcal{S}$ such that for any security parameter $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}$, any input $\mathsf{in}_A \in I$, and any function $f \in F$, we have

$$\mathsf{real}_{\Pi,\mathcal{A}}(\lambda, \mathsf{pp}; \mathsf{in}_A) \overset{c}{\approx} \mathsf{ideal}_{f,\mathcal{S}}(\lambda, \mathsf{pp}; \mathsf{in}_A),$$

where the distributions are defined via the following experiments:

- $\mathsf{real}_{\Pi,\mathcal{A}}(\lambda, \mathsf{pp}; \mathsf{in}_A)$: Run the protocol Π on the security parameter λ, where party A runs the protocol honestly using its input in_A, and the messages of the corrupt party B are chosen by the adversary $\mathcal{A}$. Let y denote the output of party A, and let V denote the view of the adversary. Output (V, y).
- $\mathsf{ideal}_{f,\mathcal{S}}(\lambda, \mathsf{pp}; \mathsf{in}_A)$: Run the simulator $\mathcal{S}$ until it outputs an input in_B for the corrupt party B. Compute $y = f(\mathsf{in}_A, \mathsf{in}_B)$ and provide y to $\mathcal{S}$. Let V^* denote the final output of the simulator $\mathcal{S}$. Output (V^*, y).

Malicious Security with Abort. An ℓ-round 2-PC protocol $\Pi = \left(\mathsf{Setup}, \{\mathsf{A}_i, \mathsf{B}_i\}_{i\in[0,\ell]}\right)$ is said to satisfy *security with abort* against static malicious adversaries if for any PPT static malicious adversary $\mathcal{A}$ corrupting party B (without loss of generality), there exists a PPT simulator $\mathcal{S}$ such that for any security parameter $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \mathsf{Setup}$, any input $\mathsf{in}_A \in I$, and any function $f \in F$, we have

$$\mathsf{real}_{\Pi,\mathcal{A}}(\lambda, \mathsf{pp}; \mathsf{in}_A) \overset{c}{\approx} \mathsf{ideal}^{\mathsf{abort}}_{f,\mathcal{S}}(\lambda, \mathsf{pp}; \mathsf{in}_A),$$

where the distributions are defined via the following experiments:

- $\mathsf{real}_{\Pi,\mathcal{A}}(\lambda, \mathsf{pp}; \mathsf{in}_A)$: Run the protocol Π on the security parameter λ, where party A runs the protocol honestly using its input in_A, and the messages of the corrupt party B are chosen by the adversary $\mathcal{A}$. Let y denote the output of party A, and let V denote the view of the adversary. Output (V, y).
- $\mathsf{ideal}^{\mathsf{abort}}_{f,\mathcal{S}}(\lambda, \mathsf{pp}; \mathsf{in}_A)$: Run the simulator $\mathcal{S}$ until it outputs an input in_B for the corrupt party B. Compute $y = f(\mathsf{in}_A, \mathsf{in}_B)$ and provide y to $\mathcal{S}$. If the simulator $\mathcal{S}$ chooses to abort, set $y^* = \bot$. Else, set $y^* = y$. Let V^* denote the final output of the simulator $\mathcal{S}$. Output (V^*, y^*).

Structural Formulation. We now formulate an ℓ-round 2-PC protocol using a structural formulation that is geared towards capturing the core property that two parties can compute the same function output using two different sequences of computation across ℓ rounds of communication.

For ease of exposition, we make a (minor) alteration to our structural formulation for an ℓ-round 2-PC protocol from the standard cryptographic definition presented earlier. In the structural formulation, we assume that the parties A and B commit to "some" random coins r_A and r_B at the beginning of the protocol, and then re-use these coins to generate their messages throughout the protocol. We note, however, this definition is essentially equivalent to the "lazy" randomness sampling strategy in the standard definition presented earlier; indeed, we can assume that the parties commit to some "master" random coins at the beginning of the protocol, and use these to derive the individual random coins to be used in each round (depending on the transcript of messages exchanged up until that round). We emphasize that we do not need to assume any computational assumptions here: the "master" random coins could just be enough random coins to last through the whole protocol.

Similar to our alternative structural formulation of 2-party NIKE, it turns out that this alternative definition (where the parties commit to some "master" random coins at the beginning of the protocol and re-use the same to generate messages throughout the protocol) makes it easier to capture the "natural" mathematical structure inherent to an ℓ-round 2-PC protocol. Although this would result in "less practical" 2-PC protocols, it allows us to only have to define two sampling distributions (one for each player) rather than 2ℓ (one for each player in each round) and lets us considerably simplify our proofs of equivalence later in this section. We illustrate this in more details subsequently.

Definition 4.2 (ℓ-Round 2-PC (Structural Formulation)). Let PP, R, I, F, $\{S_{i,A}, S_{i,B}\}_{i\in[\ell]}$, $\{\Gamma_i\}_{i\in[0,\ell]}$, R_A, R_B, and Y denote *sets*. More specifically:

- We let PP denote the set of public parameters and R_A and R_B denote the set of possible random coins used by the setup algorithm to output some public parameters from the set PP.

- Let I and F denote the set of all possible inputs and the set of all possible functions, as defined earlier.

- For each $i \in [\ell]$, we let $S_{i,A}$ and $S_{i,B}$ denote the set of possible round-messages output in round-i by the parties A and B, respectively.

- For each $i \in [0, \ell]$, we let Γ_i denote the set of all possible transcripts of messages exchanged between the parties A and B until round i.

- We also let R_A and R_B denote the set of possible secret states for the parties A and B, respectively.

- Finally, we let Y denote the set of possible final outputs for the parties A and B at the end of the ℓ-round 2-PC protocol.

Next, we define the following functions that map between these sets as below:

- $\mathsf{Setup} : 1^\lambda \times R \to PP$.

- $\{\mathsf{Gen}_{i,A} : PP \times I \times F \times R_A \times \Gamma_i \to S_{i+1,A}\}_{i\in[0,\ell-1]}$.

- $\{\mathsf{Gen}_{i,B} : PP \times I \times F \times R_B \times \Gamma_i \to S_{i+1,B}\}_{i\in[0,\ell-1]}$.

- $\mathsf{Combine}_A : PP \times I \times F \times R_A \times \Gamma_\ell \to Y$.

- $\mathsf{Combine}_B : PP \times I \times F \times R_B \times \Gamma_\ell \to Y$.

Correctness, semi-honest simulation-based security and malicious security (with and without abort) are as defined analogously to the cryptographic definitions presented earlier.

Distributional Simulation-Secure CMA for 2-PC (ℓ-DCMA$_{\text{2-PC}}$). We now define an ℓ-DCMA$_{\text{2-PC}}$ as follows.

Definition 4.3 (ℓ-DCMA$_{\text{2-PC}}$). A monoid action $(M, X, \star)$ is an ℓ-DCMA$_{\text{2-PC}}$ if it satisfies following additional structural properties and security properties:

- Structural properties:
  - The monoid $(M, \oplus)$ is a string concatenation monoid structured as $M = M_A \cup M_B$ where
$$M_A = I \times F \times R_A, \quad M_B = I \times F \times R_B,$$
    such that both of the sub-monoids M_A and M_B are individually string concatenation monoids themselves.
  - The set X is structured as
$$X = PP \times \left(\bigcup_{i \in [\ell]} S_{i,A} \cup \bigcup_{i \in [\ell]} S_{i,B} \cup \{\perp\} \right) \times (Y \cup \{\perp\}) .$$
  - For any public parameters $\mathsf{pp} \in PP$, any pair of inputs $\mathsf{in}_A, \mathsf{in}_B \in I$, any function $f \in F$, and any pair of randomnesses $(r_A, r_B) \in R_A \times R_B$, letting
$$\begin{aligned} g = (\mathsf{in}_A, f, r_A) \in M_A \quad &, \qquad h = (\mathsf{in}_B, f, r_B) \in M_B, \\ x = (\mathsf{pp}, \perp, \perp) \in X \quad &, \qquad y = (\mathsf{pp}, \perp, f(\mathsf{in}_A, \mathsf{in}_B)) \in X. \end{aligned}$$
    we have
$$(g \oplus h)^{\ell} \star x = (h \oplus g)^{\ell} \star x = y.$$

- Distributional simulation security:

  An ℓ-DCMA$_{\text{2-PC}}$ is said to satisfy distributional simulation security with respect to the triplet of distributions $(\mathcal{D}_{M,0}, \mathcal{D}_{M,1}, \mathcal{D}_X)$ (where $\mathcal{D}_{M,0}$ and $\mathcal{D}_{M,1}$ are distributions over M and $\mathcal{D}_X$ is a distribution over the set X) if there exist PPT simulators $\overline{\mathcal{S}}_A$ and $\overline{\mathcal{S}}_B$ such that for any security parameter $\lambda \in \mathbb{N}$, any $g \leftarrow \mathcal{D}_{M,0}$, any $h \leftarrow \mathcal{D}_{M,1}$, and any $x \leftarrow \mathcal{D}_X$, letting
$$x_{i,0} = (g \oplus h)^{i-1} \star x, \quad x_{i,1} = (h \oplus g)^{i-1} \star x,$$
$$x'_{i,0} = \left(g \oplus (h \oplus g)^{i-1}\right) \star x, \quad x'_{i,1} = \left(h \oplus (g \oplus h)^{i-1}\right) \star x,$$
  for each $i \in [\ell]$, and letting
$$y = (g \oplus h)^{\ell} \star x = (h \oplus g)^{\ell} \star x,$$
  we have
$$\overline{\mathcal{S}}_A \left(1^{\lambda}, x, g, y\right) \overset{c}{\approx} \overline{\mathcal{S}}_B \left(1^{\lambda}, x, h, y\right) \overset{c}{\approx} \left(x, \{x_{i,0}, x_{i,1}, x'_{i,0}, x'_{i,1}\}_{i \in [\ell]}, y\right) .$$

ℓ-DCMA$_{\text{2-PC}}$ and ℓ-round 2-PC are Equivalent. We state the following theorem:

Theorem 4.4. *Any ℓ-round 2-PC protocol satisfying Definition 4.2 implies an ℓ-DCMA$_{\text{2-PC}}$ satisfying Definition 4.3, and vice versa.*

The construction of ℓ-round 2-PC given an ℓ-DCMA$_{\text{2-PC}}$ is reasonably straightforward and follows the template of the construction of (ℓ-round) KE given an (ℓ-)DUCMA. The construction of ℓ-DCMA$_{\text{2-PC}}$ given an ℓ-round 2-PC is more involved, but again follows the template of the construction of ℓ-DUCMA given any ℓ-round KE protocol, as outlined in the proof of Theorem 3.26. Hence, we do not detail the proof any further.

String-Concatenation Monoid Action Oracles for 2-PC. We extend our definition of a generic string concatenation monoid action oracles (SCMA) in order to model 2-PC. We refer to this extension of SCMA as SCMA$_{\text{2-PC}}$. Informally speaking, an SCMA$_{\text{2-PC}}$ oracle (with certain restrictions as outlined subsequently) is a DCMA$_{\text{2-PC}}$ *in the strongest possible sense*, much like how an SCMA oracle is a DUCMA in the strongest possible sense.

Definition 4.5 (Generic SCMA$_{\text{2-PC}}$ Oracle). An SCMA$_{\text{2-PC}}$ oracle $\mathbf{M}(\cdot,\cdot)$ over an alphabet $\Sigma \subset \{0,1\}^*$ is an SCMA oracle with additional property that the alphabet Σ is structured as $\Sigma = \Sigma_A \cup \Sigma_B$ where $\Sigma_A, \Sigma_B \subset \{0,1\}^*$.

Generic k-restricted SCMA$_{\text{2-PC}}$ Oracle. We now formally define a more general "k-layered" restriction of a generic SCMA$_{\text{2-PC}}$ oracle with a k-base set element x_0 (where k-base element is as defined earlier for SCMA oracles).

Definition 4.6 (Generic k-restricted SCMA$_{\text{2-PC}}$ Oracle). A generic k-restricted SCMA$_{\text{2-PC}}$ oracle $\mathbf{M}(\cdot,\cdot)$ over an alphabet $\Sigma \subset \{0,1\}^*$ is a random variable whose values are functions $\mathbf{M} : \Sigma^* \times \{0,1\}^* \cup \{\perp\} \to \{0,1\}^* \cup \{\perp\}$ and which satisfies all of the properties of a generic SCMA$_{\text{2-PC}}$ oracle, with the following additional constraints:

1. For any $s \in \Sigma^*$, we have $\mathbf{M}(s, \perp) = \perp$.
2. For any $s \in \Sigma^*$ and any $x \in \{0,1\}^*$, we have $\mathbf{M}(s, x) = \perp$ if *either* of the following conditions holds:
   - **Either** $\mathsf{Level}_k(x) = -1$.
   - **Or** $|s| + \mathsf{Level}_k(x) > 2k$ (where $|s|$ denotes the length of the string s).
   - **Or** s not of the from $s = a_1 b_1 a_2 b_2 \ldots$ or $s = b_1 a_1 b_2 a_2 \ldots$ for $a_i \in \Sigma_A$ and $b_i \in \Sigma_B$.

In this paper, we consider k-restricted SCMA$_{\text{2-PC}}$ oracles that additionally satisfy certain commutator-like properties, defined formally below.

Definition 4.7 (k'-Commutator k-restricted SCMA$_{\text{2-PC}}$ Oracle). A generic k-restricted SCMA$_{\text{2-PC}}$ oracle over an alphabet $\Sigma = \Sigma_A \cup \Sigma_B$ with initial element x_0 is said to be a k'-commutator (for $k' \in [1, k]$) if for any $a \in \Sigma_A, b \in \Sigma_B$, we have

$$\mathbf{M}\left((ab)^{k'}, x_0\right) = \mathbf{M}\left((ba)^{k'}, x_0\right).$$

In particular, we use k-restricted SCMA$_{\text{2-PC}}$ oracles that are also k-commutator. In the rest of the paper, when we refer to k-restricted SCMA$_{\text{2-PC}}$ oracles, we assume that they are additionally k-commutator by default (unless specified otherwise); hence, we do not explicitly specify the k-commutator property.

Semi-Honest Secure 2-PC from SCMA Oracle. We now state the following lemma.

Lemma 4.8. *There exists a construction of semi-honest $(2k-1)$-round 2-PC protocol from any k-restricted SCMA oracle $\mathbf{M}(\cdot,\cdot)$ over a sufficiently large alphabet Σ.*

Proof. The proof of this lemma is very similar to the proof of Lemma 3.47 and is hence not detailed.

Maliciously Secure 2-PC from SCMA Oracle. We now state the following lemma.

Lemma 4.9. *There exists a construction of $(2k-1)$-round 2-PC protocol satisfying malicious security with abort from any k-restricted SCMA oracle $\mathbf{M}(\cdot,\cdot)$ over a sufficiently large alphabet Σ.*

Proof Overview. The proof of this lemma is builds upon the proof of Lemma 4.8 for the existence of a semi-honest secure $(2k-1)$-round 2-PC protocol, except that we need a way for the simulator to extract the input of the corrupt party (concretely, the simulator needs to extract the monoid element representing the input of the corrupt party that is used in the various queries to the k-restricted SCMA oracle). This, however, is immediate from the following observation: in the real world, if the adversary does not abort and the honest party receives some output $y = f(\mathsf{in}_A, \mathsf{in}_B)$ corresponding to some input in_B used by the adversary, then the messages sent to the honest party by the adversary $\mathcal{A}$ must embed information about the monoid element representing in_B. Additionally, any message that the adversary sends to the honest party must be the output of a query to the k-restricted SCMA oracle (since there is no other way of generating valid set elements corresponding to the k-restricted SCMA oracle). In the ideal world, the simulator can thus observe all the queries issued by the adversary to the k-restricted SCMA oracle, thus extracting any input monoid element used by the adversary with non-negligible probability. The remainder of the simulation strategy is identical to that in the proof of Lemma 4.8, and is hence not detailed.

4.2 Separating $2k$-round 2-PC from $(2k+1)$-round Maliciously Secure 2-PC

Our (informal) goal is to black-box separate any $2k$-round 2-PC protocol from any $(2k+1)$-round maliciously secure 2-PC protocol. Subsequently, in Section 4.3, we show that the separation of $(2k+1)$-round 2-PC protocol from any $(2k+2)$-round maliciously secure 2-PC protocol follows analogously. Concretely, we establish the impossibility of a secure $2k$-round 2-PC protocol where the participants Alice and Bob only make queries to a generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle (which in turn implies a maliciously secure $(2k+1)$-round 2-PC protocol, as demonstrated earlier). Note that this immediately (black-box) separates $2k$-round 2-PC from any $(2k+1)$-round 2-PC protocol.

In particular, we wish to establish that for any $2k$-round 2-PC protocol where the participants Alice and Bob only make queries to a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, there exists an attacker Eve that corrupts Bob and *recovers the input* of the honest party Alice with non-negligible probability. Note that the corruption by Eve is *semi-honest*; in fact, it suffices for Eve to only have access to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle and the messages exchanged publicly between Alice and Bob. This allows us to prove an even stronger result, namely that it is impossible to construct any $2k$-round *semi-honest secure* 2-PC protocol from any $(2k+1)$-round maliciously secure 2-PC protocol in a black-box manner.

Before we formalize this goal, we define $2k$-round 2-PC and introduce several notations for executions and probability distributions associated with a $2k$-round . In the rest of the section, when we refer to a generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$, we assume that it is $(k+1)$-commutator by default. We note that this is analogous to our strategy for black-box separation of key exchange as well. Additionally, in the rest of the section, when we refer to any 2-PC protocol, we assume malicious corruptions by default.

4.2.1 Round-based Definition of $2k$-round 2-PC

We begin by formally defining a $2k$-round 2-PC protocol where the participants are Alice and Bob, and Eve is the adversary (corrupting either Alice or Bob in a semi-honest manner), all of whom have access to a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle. We assume w.l.o.g. that Alice, Bob, and Eve will never issue the same $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query twice. Also, we assume that Alice (resp., Bob) issues at most n_A (resp., n_B) $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries. Throughout this section, we abuse notation and redefine several notations from the context of key exchange to the context of 2-PC below.

Rounds and Sub-Rounds. We assume that Alice has input in_A and Bob has input in_B. Each round i (for $i \geq 1$) consists of a message $\mathsf{m}_{AB}^{(i)}$ sent from Alice to Bob and a message $\mathsf{m}_{BA}^{(i)}$ sent from Bob to Alice. Each round i consists of several sub-rounds (i,j) for $j \in [n_i+1]$ defined as follows:

- Each sub-round (i,j) for $j \in [n_i]$ begins with *either* Alice *or* Bob issuing a *single* (new) $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query, and ends with with Eve issuing her (new) oracle queries based on the set of messages exchanged between Alice and Bob so far, defined as

$$\mathsf{m}^{[i-1]} = \left\{\mathsf{m}_{AB}^{(1)}, \mathsf{m}_{BA}^{(1)}, \ldots, \mathsf{m}_{AB}^{(i-1)}, \mathsf{m}_{BA}^{(i-1)}\right\}.$$

  In these sub-rounds, Alice and Bob do not exchange any messages.

- Sub-round (n_i+1) involves the following steps that happen simultaneously:

  - Alice computes her message $\mathsf{m}_{AB}^{(i)}$ and sends it to Bob.
  - Simultaneously, Bob computes his message m_{BA} and sends it to Alice.

  While computing the above messages, both Alice and Bob only use their own oracle queries till round $(i-1)$, and the set of messages exchanged between Alice and Bob till round $(i-1)$, defined as

$$\mathsf{m}^{[i-1]} = \left\{\mathsf{m}_{AB}^{(1)}, \mathsf{m}_{BA}^{(1)}, \ldots, \mathsf{m}_{AB}^{(i-1)}, \mathsf{m}_{BA}^{(i-1)}\right\}.$$

We define the sub-rounds as above for ease of exposition, and for simplifying the attack analysis presented subsequently.

Queries and Views. We use the following notations to denote the queries and views of Alice, Bob, and Eve at the end of various sub-rounds:

- $Q_A^{(i,j)}$ (resp., $Q_B^{(i,j)}$ and $Q_E^{(i,j)}$): denotes the set of $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i,j).

- $P_A^{(i,j)}$ (resp., $P_B^{(i,j)}$ and $P_E^{(i,j)}$): denotes the set of query-response pairs corresponding to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i,j). More formally, for $\alpha \in \{A, B, E\}$, we have

$$P_\alpha^{(i,j)} = \left\{((s, x, y = \mathbf{M}(s,x))) : (s,x) \in Q_\alpha^{(i,j)}\right\}.$$

- $V_A^{(i,j)}$ (resp., $V_B^{(i,j)}$ and $V_E^{(i,j)}$): denotes the views of Alice (resp., Bob and Eve) by the end of sub-round (i,j). More formally, for $\alpha \in \{A, B\}$, we have

  $$V_\alpha^{(i,j)} = \left(\mathsf{in}_\alpha, \mathsf{r}_\alpha, \mathsf{m}^{(i,j)}, P_\alpha^{(i,j)}\right),$$

  where r_A (resp., r_B) denotes the internal randomness of Alice (resp., Bob). In addition, we have

  $$V_E^{(i,j)} = \left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

  In particular, the view of Eve does not have any randomness since Eve does not use any randomness.

We again adopt the notation $\mathcal{Q}(\cdot)$ from [BM09] to denote an operator that extracts the set of queries from any set of $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs or views; namely, for any set of query-response pairs P and any view $V = (r, \mathsf{m}, P)$, we have

$$\mathcal{Q}(P) = \mathcal{Q}(V = (r, \mathsf{m}, P)) = \{q = (s, x) : \exists y, (s, x, y) \in P\}.$$

Finally, we analogously use the notations $Q_A^{(i)}$ (resp,. $Q_B^{(i)}$ and $Q_E^{(i)}$), $P_A^{(i)}$ (resp,. $P_B^{(i)}$ and $P_E^{(i)}$) and $V_A^{(i)}$ (resp,. $V_B^{(i)}$ and $V_E^{(i)}$) to denote the set of queries asked by Alice (resp., Bob and Eve), the set of query-response pairs corresponding to the queries asked by Alice (resp., Bob and Eve), and the view of Alice (resp., Bob and Eve) at the end of all sub-rounds of round i in the 2-PC protocol.

Executions and Distributions. A (full) execution of Alice, Bob, and Eve can be described by a tuple $(\mathsf{r}_A, \mathsf{r}_B, \mathbf{M})$, where r_A denotes Alice's random tape, r_B denotes Bob's random tape, and **M** denotes the generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ (note that Eve is deterministic). We denote by $\mathcal{E}$ the distribution over (full) executions, obtained by running the algorithms for Alice, Bob and Eve with uniformly chosen random tapes r_A, r_B, and a uniformly sampled generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ **M**. We denote by $\Pr_{\mathcal{E}}[P_A^{(i,j)}]$ (resp., $\Pr_{\mathcal{E}}[P_B^{(i,j)}]$ and $\Pr_{\mathcal{E}}[P_E^{(i,j)}]$) the probability that $P_A^{(i,j)}$ (resp., $P_B^{(i,j)}$ and $P_E^{(i,j)}$) is the set of query-response pairs corresponding to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i,j) during the execution.

For any (i,j), for any sequence of exchanged messages $\mathsf{m}^{(i,j)}$, and for any set of $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs $P_E^{(i,j)}$, we denote by $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ the joint distribution over the views $\left(V_A^{(i,j)}, V_B^{(i,j)}\right)$ of Alice and Bob in their own (partial) executions up to just before the sub-round (i,j), conditioned on the event that:

1. the transcript of messages exchanged between Alice and Bob until this point being equal to $\mathsf{m}^{(i,j)}$, and
2. the set of all $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs corresponding to the queries issued by Eve until this point being equal to $P_E^{(i,j)}$.

We denote the probability of the aforementioned event by $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}]$. Similar to in [BM09], we use the distribution $\mathcal{V}(\mathsf{m}^{(i,j)})$ to essentially capture the conditional distribution of Alice's and Bob's views in the eyes of the attacker Eve who knows the public messages exchanged between Alice and Bob, and has learned all $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs described in $P_E^{(i,j)}$.

Intersection Queries and Equivalence Queries. We now formally define intersection and equivalence queries. Recall that for any (i,j), $Q_A^{(i,j)}$ (resp., $Q_B^{(i,j)}$) denotes the set of $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by Alice (resp., Bob and Eve) by the end of sub-round (i,j).

Intersection Queries. We define the set of intersection queries

$$Q_{A\cap B}^{(i,j)} = Q_A^{(i,j)} \cap Q_B^{(i,j)},$$

to be the set of *common* $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by *both* Alice *and* Bob until sub-round-(i,j).

Equivalence Queries. We now define the concept of *equivalent* queries with respect to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle queries issued by Alice and Bob.

Definition 4.10 (Equivalence Queries). Let $q_A = (s_A, x_A)$ and $q_B = (s_B, x_B)$ be two queries issued by Alice and Bob to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle. We say that q_A and q_B are *equivalent* queries if the following conditions hold simultaneously:

- $(s_A, x_A) \neq (s_B, x_B), \mathbf{M}(s_A, x_A) \neq \perp, \mathbf{M}(s_B, x_B) \neq \perp$.
- One of the following two cases must be true (x_0 being the $(k+1)$-base set element for the $(k+1)$-restricted SCMA$_{\text{2-PC}}$):
  - **Either** there exist $s'_A, s'_B \in \Sigma^*$ such that
  $$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = s_B \| s'_B.$$
  - **Or** there exist $a, b \in \Sigma$, and $s'_A, s'_B \in \Sigma^*$, such that
  $$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = (ab)^{k+1}, \quad s_B \| s'_B = (ba)^{k+1}.$$

Note that the first condition immediately implies that $\mathbf{M}(s_A, x_A) = \mathbf{M}(s_B, x_B)$. Additionally, the second condition also implies that

$$\begin{aligned} \mathbf{M}(s_A, x_A) &= \mathbf{M}(s_A \| s'_A, x) = \mathbf{M}((ab)^{k+1}, x) \\ &= \mathbf{M}((ba)^{k+1}, x) = \mathbf{M}(s_B \| s'_B, x) = \mathbf{M}(s_B, x_B). \end{aligned}$$

In other words, equivalence queries essentially depict two different sequences of queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle leading to the same (valid) output, and the two possibilities mentioned above depict the only scenarios that could lead to such a "collision" between two different sequence of queries with non-negligible probability (this follows immediately from statistical independence properties of the outputs of a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle on uncorrelated inputs).

Remark 4.11. We again remark here that, as in the case of our KE separation result, we could also have some additional classes of equivalence queries that are essentially combinations of the above two cases. However, we again avoid explicitly enumerating them since we do not need them for our eventual separation proof.

Next, we define the equivalence relation $\mathcal{R}_{A\equiv B}$ as follows:

$$\mathcal{R}_{A\equiv B} = \begin{cases} 1 & \text{if and only if } q_A \text{ and } q_B \text{ are equivalent,} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define the set of equivalence queries

$$Q_{A\equiv B}^{(i,j)} \qquad = \{(q_A, q_B \in Q_A^{(i,j)} \times Q_B^{(i,j)} : \mathcal{R}_{A\equiv B}(q_A, q_B) = 1\},$$

to be the set of equivalence query-pairs (where each pair consists of a query issued by Alice and a query issued by Bob) until sub-round-(i,j).

Good Events. For any (i,j), for any sequence of exchanged messages $\mathsf{m}^{(i,j)}$, and for any set of $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs $P_E^{(i,j)}$ (corresponding to queries issued by Eve) such that $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$, we define the following:

- The event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is defined over the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and is said to hold if and only if:
$$Q_{A\cap B}^{(i,j)} \subseteq \mathcal{Q}(P_E^{(i,j)}),$$
where $Q_{A\cap B}^{(i,j)}$ and $Q_{A\equiv B}^{(i,j)}$ are determined by $Q_A^{(i,j)}$ and $Q_B^{(i,j)}$, which are in turn determined by sampling the views of Alice and Bob as
$$\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$
- The event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ is defined over the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and is said to hold if and only if:
$$Q_{A\cap B}^{(i,j)} \subseteq \mathcal{Q}(P_E^{(i,j)}) \quad \text{and} \quad \forall (q_A, q_B) \in Q_{A\equiv B}^{(i,j)}, q_A \in \mathcal{Q}(P_E^{(i,j)}) \vee q_b \mathcal{Q}(P_E^{(i,j)}),$$
where $Q_{A\cap B}^{(i,j)}$ and $Q_{A\equiv B}^{(i,j)}$ are again determined by $Q_A^{(i,j)}$ and $Q_B^{(i,j)}$, which are in turn again determined by sampling the views of Alice and Bob as
$$\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right).$$

Intuitively, the event $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ indicates that Eve has issued all queries that have been issued by both both Alice and Bob, while the event $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ indicates that Eve has not only issued all queries that have been issued by both both Alice and Bob, but also at least one query from each pair of equivalence queries issued by Alice and Bob.

Finally, we denote by $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ the distributions obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the events $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, respectively.

The Main Separation Theorem for 2-PC. We prove the following main theorem:

Theorem 4.12 (Main Theorem for 2-PC Separation). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob such that:*

- *Alice and Bob have inputs in_A and in_B, respectively.*
- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle, and use random tapes r_A and r_B, respectively.*
- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

*Then for every $0 < \delta < \rho$, there exists **an attacker Eve** that corrupts Bob and makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle, corresponding to which, with probability at least $\rho - \delta$, **there exists no probabilistic simulator** $\mathcal{S}$ that makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle such that*

$$\mathcal{S}^{\mathbf{M}(\cdot,\cdot)}(\mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B)) \overset{c}{\approx} V_{\text{Eve}}^{\Pi},$$

where V_{Eve}^{Π} denotes the view of Eve (consisting of the messages exchanged by Alice and Bob, Eve's queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ *oracle, and Eve's own internal random coins, if any).*

Proof Strategy. Our proof strategy is analogous to that for our KE separation proof, and involves showing the existence of an attacker Eve that recovers more information about the honest party Alice's input in_A than is revealed by the knowledge of Bob's input in_B and the function output $f(\mathsf{in}_A, \mathsf{in}_B)$. Consequently, an ideal-world simulator $\mathcal{S}$ can never simulate Eve's view since it can never obtain this additional information about Alice's input in_A (except with non-negligible probability) given only $(\mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B))$. More formally, we prove the following auxiliary theorem, which in turn implies the main theorem above.

Theorem 4.13 (Auxiliary Theorem for 2-PC Separation). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob such that:*

- *Alice and Bob have inputs in_A and in_B, respectively.*
- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle, and use random tapes r_A and r_B, respectively.*
- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

Then for every $0 < \delta < \rho$, there exists an attacker Eve that corrupts Bob and makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ *oracle, such that Eve recovers, with probability at least $(\rho - \delta)$, **all** queries made by Alice to the $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle that are either identical to or are "equivalent" to the queries made by Bob to the $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle.*

Auxiliary Theorem 4.13 implies Main Theorem 4.12. For the sake of explanation, we briefly outline why this theorem still implies the existence of a valid attack on the 2-PC protocol Π. To begin with, observe that since the outputs of the generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle $\mathbf{M}$ are (by definition) uniformly random and uncorrelated except for the commutator relation that allows computing the function output in an honest execution of the protocol, Alice and Bob must issue certain intersection/equivalence queries to $\mathbf{M}$ in order to arrive at the final output with high enough probability, and these queries must contain information about the parts of the inputs in_A and in_B of Alice and Bob, respectively, that are relevant to the final function output $f(\mathsf{in}_A, \mathsf{in}_B)$. We emphasize that this follows from the definition of the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, which forces Alice and Bob to use the same input on every step for correctness to hold.

Also, note that Eve recovers (with high enough probability) **all** of the intersection and equivalence queries made by Alice and Bob to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle based on their respective inputs. As a result, Eve recovers more information about Alice's input beyond what is revealed trivially by the function output. In particular, since the SCMA$_{\text{2-PC}}$ oracle enforces Alice and Bob to use the same input on every step, Eve manages to recover the "exact" query Alice used in the computation that was used to get the final result. This in turn implies that Eve manages to extract a query from the SCMA$_{\text{2-PC}}$ oracle that allows her to simulate the computation on Alice's input for any of Bob's inputs she likes (thus breaking security of the 2-PC protocol Π immediately).

Finally, we also emphasize that, for perfect correctness to hold, Alice must use a query that (if it doesn't correspond to her correct input) must result in the exact same output for all possible inputs of Bob. Alice could, of course, use a query that corresponds to a different input than her "official" input in the protocol (as long as it gives the same results on all queries) in the process, but finding this again is clearly enough to break the security of the 2-PC protocol Π since, once again, Eve could simulate the computation on Alice's input for any of Bob's inputs.

Remark 4.14. In our proof, we construct an attacker Eve that recovers the part of Alice's input that is relevant to the output of the function (more concretely, the secret monoid element representing Alice's input that is used in Alice's queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle). Eve does not recover any parts of Alice's inputs that were not used by Alice to query the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle. In fact, it is impossible in general to recover any parts of Alice's input that are (potentially) irrelevant to the output, since Alice can (at least sometimes) start the interaction by first deleting the irrelevant part its input. We note, however, that recovering the part of Alice's input that is relevant to its output already constitutes an attack on the security of the 2-PC protocol since it allows Eve to learn potentially greater information than is leaked by the corrupt party Bob's output.

Remark 4.15. We remark that our attack strategy only allows Eve to recover the output of a single honest party, namely Alice. In particular, in the case where Bob is also honest and Eve only observes the messages exchanged by Alice and Bob, our attack strategy only allows it to recover the input of either Alice or Bob, but not necessarily the inputs of both parties. It is worth noting here that, in the case of 2-PC protocols, it is sometimes possible to only find one player's secret. Consider the following protocol: Alice sends her input to Bob in the clear, and then Bob computes the function(s) and outputs the result (or at least Alice's output) in the clear. This is technically a (insecure) 2-PC because both parties learn the final result (or at least, the output for Alice), and Alice's secret input, but clearly we cannot extract Bob's secret input (or even Bob's output if it is different from Alice's). Since our attack (or any generic attack on 2-PC protocols) should handle this situation, it seems hard to come up with a generic attack on any 2-PC protocol that recovers both parties' inputs and/or outputs.

Before describing Eve's attack algorithm, we introduce a special form of $2k$-round 2-PC (the existence of which is implied by any $2k$-round 2-PC protocol). The special form of $2k$-round 2-PC is introduced purely to make our attack analysis easier; our attack applies to any $2k$-round 2-PC protocol. We emphasize that we used a similar strategy in our key exchange proof.

4.2.2 2-PC with Equivalence Complete Query Pattern

We now introduce what we call an *equivalence complete* query pattern for Alice and Bob during an execution of a $2k$-round 2-PC protocol, which essentially depicts a sequence of queries issued by Alice and Bob to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, albeit subject to certain constraints as described subsequently.

Definition 4.16 (Query Length). Let $\mathbf{M}(\cdot,\cdot)$ be a generic $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, and let (s,x) be a query to $\mathbf{M}$. Let $s = s_1 \| \ldots \| s_\ell$ be a "decomposition" of s such that each $s_i \in \Sigma^*$ for $i \in [\ell]$. We say that the "length" of the query (for this decomposition) is ℓ. Observe that, by the associative properties of the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, we must have

$$\mathbf{M}(s,x) = \mathbf{M}(s_1, \mathbf{M}(s_2, \ldots, \mathbf{M}(s_\ell, x) \ldots)).$$

Remark 4.17. Note that the length of the query may vary depending on the decomposition of the string s, and may be different from $|s|$, which denotes the unique number of symbols from Σ in the string s.

Definition 4.18 (Equivalence Complete Query Pattern). Let Q be any set of queries to a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, such that each query $q \in Q$ is of the form $q = (s,x) \in \Sigma^* \times \{0,1\}^*$. We say that Q is equivalence complete if the following conditions are satisfied (x_0 being the $(k+1)$-base set element of the generic $(k+1)$-SCMA$_{\text{2-PC}}$ oracle):

- Informally, for any query $q \in Q$, the query set Q also contains all the "split" versions of this query. Formally, for each $q = (s,x) \in Q$ such that $x = \mathbf{M}(s', x_0)$ and such that $s \| s' = a_1 \ldots a_\ell$ for $\ell > 1$ (where for each $j \in [\ell]$, we have $a_j \in \Sigma$), there exists a subset of "single-element" queries $S \subset Q$ of the form
$$S = \{q_1 = (s_1, x_1), \ldots, q_\ell = (s_\ell, x_\ell)\},$$
such that for each $j \in [\ell]$, we
$$s_j = a_j, \quad x_j = \mathbf{M}(a_{j+1}, \mathbf{M}(a_{j+2}, \ldots, \mathbf{M}(a_\ell, x_0) \ldots)).$$
- Informally, for any query $q \in Q$ that is a substring of either $(ab)^{k+1}$ or $(ba)^{k+1}$, and which potentially "triggers" a build-up to an equivalence query of the form $\mathbf{M}\left((ab)^{k+1}, x_0\right) = \mathbf{M}\left((ba)^{k+1}, x_0\right)$, the query set Q also contains all the possible ways to compute this equivalence query. Formally, for any $q = (s,x) \in Q$ such that $x = \mathbf{M}(s', x_0)$ and such that there exist distinct elements $a, b \in \Sigma$ such that
$$|s \| s'| > 2, \quad s \| s' \in \mathsf{SUBSTRING}\left((ab)^{k+1}\right) \cup \mathsf{SUBSTRING}\left((ba)^{k+1}\right),$$
where $\mathsf{SUBSTRING}\left((ab)^{k+1}\right)$ and $\mathsf{SUBSTRING}\left((ba)^{k+1}\right)$ denote the sets of all possible substrings of $(ab)^{k+1}$ and $(ba)^{k+1}$, respectively, we must have
$$S_0 \subset Q \wedge S_1 \subset Q,$$
where the query subsets S_0 and S_1 are defined as:
$$S_0 = \left\{\widetilde{q} = (\widetilde{s}, x_0) : \widetilde{s} \in \mathsf{SUBSTRING}\left((ab)^{k+1}\right)\right\},$$
$$S_1 = \left\{\widetilde{q} = (\widetilde{s}, x_0) : \widetilde{s} \in \mathsf{SUBSTRING}\left((ba)^{k+1}\right)\right\}.$$

Definition 4.19 (2**-PC with Equivalence Complete Query Pattern).** Let Π be any 2-PC protocol as defined in Section 4.2.1. The protocol Π is said to have equivalence complete query pattern if for any round i, letting $Q_A^{(i)}$ and $Q_B^{(i)}$ denote the set of queried asked by Alice and Bob to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, we have that $Q_A^{(i)}$ and $Q_B^{(i)}$ are both equivalence complete query patterns as per Definition 4.18.

Equivalence Queries Follow Intersection Queries. We now state and prove that for any $2k$-round 2-PC protocol with equivalence complete query pattern where Alice and Bob make queries to a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, for each equivalence query, there exists a corresponding intersection query such that if Eve makes this intersection query, she makes a query that is either identical to or equivalent to the original equivalence query. It is this special property of a 2-PC protocol with equivalence complete query pattern that makes our subsequent attack analysis significantly simpler.

We note here that this step again constitutes a core novelty of our attack analysis, and is necessitated by the additional algebraic structure that is inherent to a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle over and above a plain random oracle. In particular, the proofs of [IR89, BM09] do not require this additional analysis since any equivalence query is, by definition, an intersection query by default for a plain random oracle. However, since this is not the case for a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, we additionally need to establish that Eve can "cover" all equivalence queries by identifying only the intersection queries. We formally prove this via Lemmas 4.20 and 4.21, that we state and prove below.

Lemma 4.20 (Equivalence Queries Follow Intersection Queries-1). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob till round i of a $2k$-round 2-PC protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) = ((s_A, x_A), (s_B, x_B)) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist $s'_A, s'_B \in \Sigma^*$ such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = s_B \| s'_B.$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the 2-PC protocol. Then there exists a set intersection queries

$$S = \{q_1, \ldots, q_\ell\} \subset Q_A^{(i)} \cap Q_B^{(i)},$$

such that if Eve asks each query in S, she asks a query that is equivalent to both the queries q_A and q_B.

Proof. Since Alice and Bob are only given the initial set-element x_0, they must have each issued a sequence of queries building up to the queries (s'_A, x_0) and (s'_B, x_0), respectively. By the definition of equivalence complete query pattern, they also issue all possible singleton queries leading up to these queries. In addition, they also issued all possible singleton queries building up to the queries (s_A, x_A) and (s_B, x_B), respectively. Suppose

$$s_A \| s'_A = s_B \| s'_B = a_1 a_2 \ldots a_\ell,$$

where for each $j \in [\ell]$, we have $a_j \in \Sigma$. Then, by definition of equivalence complete query pattern, there exists a set of queries of the form

$$S = \{q_1 = (s_1, x_1), \ldots, q_\ell = (s_\ell, x_\ell)\},$$

such that for each $j \in [\ell]$, we

$$s_j = a_j, \quad x_j = \mathbf{M}(a_{j+1}, \mathbf{M}(a_{j+2}, \ldots, \mathbf{M}(a_\ell, x_0) \ldots)),$$

such that $S \subset Q_A^{(i)} \cap Q_B^{(i)}$, and such that q_1 is equivalent to both q_A and q_B. This completes the proof of Lemma 4.20. □

Lemma 4.21 (Equivalence Queries Follow Intersection Queries-2). *Let $Q_A^{(i)}$ and $Q_B^{(i)}$ be the set of queries issued by Alice and Bob till round i of a $2k$-round 2-PC protocol with an equivalence complete query pattern. Suppose that there is an equivalence query pair $(q_A, q_B) \in Q_A^{(i)} \times Q_B^{(i)}$ such that there exist $a, b \in \Sigma$, and $s'_A, s'_B \in \Sigma^*$, such that*

$$x_A = \mathbf{M}(s'_A, x_0), \quad x_B = \mathbf{M}(s'_B, x_0), \quad s_A \| s'_A = (ab)^{k+1}, \quad s_B \| s'_B = (ba)^{k+1},$$

and that Alice and Bob are only given the base set element x_0 at the beginning of the 2-PC protocol. Then we must have

$$q_A \in Q_A^{(i)} \cap Q_B^{(i)} \quad \text{or} \quad q_B \in Q_A^{(i)} \cap Q_B^{(i)}.$$

Proof. We will show that if Alice and Bob compute an equivalence query of the aforementioned form in at most $2k$ rounds, then either Alice or Bob must have computed a query that triggered the equivalence complete query pattern. Therefore, (at least) one of Alice and Bob will have computed the equivalence query in all possible ways, implying the existence of a corresponding intersection query by definition.

Based on the definition of equivalence query as outlined in Definition 4.10, in this scenario, Alice and Bob effectively compute an equivalence query of the form

$$(ab)^{k+1} \star x_0 = (ba)^{k+1} \star x_0,$$

given only the base set element x_0. To do this, they each must make queries of the form $\mathbf{M}\left(t_1, t_2 \star x\right)$ where $t_1 || t_2$ is a right substring of either $(ab)^{k+1}$ or $(ba)^{k+1}$ and send these back and forth between one another, constantly building t_2. Suppose we assume that if either Alice or Bob makes multiple queries of the above form in the same round that build upon one another, we replace them with a single query. Note that this will not change the final equivalence query or whether or not we have triggered an equivalence complete query pattern.

With this assumption, we may assume that Alice and Bob make no more than $2k$ queries of the form $q_i = \mathbf{M}\left(s_i, q_{i-1}\right)$ for $i \in [2k]$ such that

$$s_1 \| \ldots \| s_{2k} = (ab)^{k+1} \quad \text{or} \quad s_1 \| \ldots \| s_{2k} = (ba)^{k+1}.$$

If less than $2k$ queries are used by either Alice or Bob (or both), we simply assume that the extra s_i strings are empty strings.

By the pigeonhole principle, at least one of the s_i strings must contain a string concatenation of both a and b. Therefore, by the definition of equivalence complete query pattern (Definition 4.18) ,at least one of Alice and Bob must have computed all possible ways to compute that particular equivalence query, and hence made the corresponding queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle. This completes the proof of Lemma 4.21. □

From any 2-PC to 2-PC with Equivalence Complete Query Pattern. Next, we show that any $2k$-round 2-PC protocol (for polynomially large k) implies the existence of a $2k$-round 2-PC protocol while incurring only a polynomial blow-up in the number of queries issued to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle by Alice and Bob (assuming that Alice and Bob make at most polynomially many queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle in the original $2k$-round 2-PC protoco). More formally, we state and prove the following lemma.

Lemma 4.22. *Assuming the existence of any secure $2k$-round 2-PC protocol (for polynomially large k) between Alice and Bob with correctness probability ρ such that Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle such that n_A and n_B are at most polynomially large, there exists a secure $2k$-round 2-PC protocol between Alice and Bob with correctness probability ρ such that the query pattern for Alice and Bob is equivalence complete, and such that Alice and Bob make at most* $\text{poly}(k, n_A)$ *and* $\text{poly}(k, n_B)$ *queries to a generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle.*

Proof. Given any $2k$-round 2-PC, we can immediately construct a $2k$-round 2-PC with equivalence complete query pattern as follows: we allow Alice and Bob to behave exactly as in the original $2k$-round 2-PC except that they additionally ask the extra queries entailed by the definition of equivalence complete query pattern, and ignore the corresponding responses of the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle to these additional queries. Since both Alice and Bob are PPT algorithms, the lengths of their queries are also poly-bounded. Hence, the blow-ups in the number of queries issued by Alice and Bob are at most $\text{poly}(k, n_A)$ and $\text{poly}(k, n_B)$, respectively. Note that neither changes the transcript of messages exchanged by Alice and Bob, nor does it change the view of Eve. This immediately implies that the following must hold:

- If the original $2k$-round 2-PC is correct with probability ρ, then the new $2k$-round 2-PC protocol with equivalence complete query pattern is also correct with the same probability ρ.
- If the original $2k$-round 2-PC is secure against any PPT adversary Eve, then the new $2k$-round 2-PC protocol with equivalence complete query pattern is also secure against any PPT adversary Eve.

This completes the proof of Lemma 4.22. □

4.2.3 Attacking 2-PC with Equivalence Complete Query Pattern

At this point, we shift focus from the main theorem to the following auxiliary theorem.

Theorem 4.23 (Theorem for 2-PC with Equivalence Complete Query Pattern). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob such that:*

- *Alice and Bob have inputs in_A and in_B, respectively.*
- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle, and use random tapes r_A and r_B, respectively.*
- *Π has an equivalence complete query pattern per Definition 4.18.*
- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

Then for every $0 < \delta < \rho$, there exists ***an attacker Eve*** *that corrupts Bob and makes at most $O(\text{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle, corresponding to which, with probability at least $\rho - \delta$,* ***there exists no probabilistic simulator*** *$\mathcal{S}$ that makes at most $O(\text{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted* SCMA$_{\text{2-PC}}$ *oracle such that*

$$\mathcal{S}^{\mathbf{M}(\cdot,\cdot)}\left(\mathsf{in}_B, f(\mathsf{in}_A, \mathsf{in}_B)\right) \overset{c}{\approx} V_{\text{Eve}}^{\Pi},$$

where V_{Eve}^{Π} denotes the view of Eve (consisting of the messages exchanged by Alice and Bob, Eve's queries to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ *oracle, and Eve's own internal random coins, if any).*

We note that Theorem 4.23, together with Lemma 4.22, immediately implies Theorem 4.12, which is the main theorem that we originally set out to prove[1]. Hence, in the rest of the paper, we focus purely on proving Theorem 4.23 in the context of a $2k$-round 2-PC with equivalence complete query pattern.

Finally, we prove Theorem 4.23 by proving the following auxiliary theorem.

Theorem 4.24 (Auxiliary Theorem for 2-PC with Equivalence Complete Query Pattern). *Let Π be a $2k$-round 2-PC protocol between Alice and Bob such that:*

- *Alice and Bob have inputs in_A and in_B, respectively.*
- *Alice and Bob make at most n_A and n_B queries, respectively, to a generic $(k+1)$-restricted* $\mathrm{SCMA}_{\text{2-PC}}$ *oracle, and use random tapes r_A and r_B, respectively.*
- Π *has an equivalence complete query pattern per Definition 4.18.*
- *Alice and Bob output s_A and s_B, respectively, such that $\Pr[s_A = s_B = f(\mathsf{in}_A, \mathsf{in}_B)] > \rho$, where the probability is taken over the choice of $(r_A, r_B, \mathbf{M})$ describing the execution of the protocol.*

Then for every $0 < \delta < \rho$, there exists an attacker Eve that corrupts Bob and makes at most $O(\mathrm{poly}(n_A, n_B, k)/\delta^2)$ queries to the generic $(k+1)$-restricted $\mathrm{SCMA}_{\text{2-PC}}$ *oracle, such that Eve recovers, with probability at least $(\rho - \delta)$,* ***all*** *queries made by Alice to the $(k+1)$-restricted* $\mathrm{SCMA}_{\text{2-PC}}$ *oracle that are either identical to or are "equivalent" to the queries made by Bob to the $(k+1)$-restricted* $\mathrm{SCMA}_{\text{2-PC}}$ *oracle.*

We note that Theorem 4.24 implies Theorem 4.23 in the same way as Theorem 4.13 implies Theorem 4.12. Indeed the only change from the previous set of theorems is that we now require the 2-PC protocol to additionally satisfy the requirement of equivalence complete query pattern, which does not affect any of the arguments for why the existence of a (semi-honest) 2-PC attacker per Theorem 4.24 implies the existence of a (semi-honest) 2-PC attacker per Theorem 4.23 whose view cannot be simulated (except with negligible probability) by any probabilistic simulator.

The Attack Algorithm. We now describe the algorithm that the attacker Eve uses to break any $2k$-round 2-PC protocol with equivalence complete query pattern. We follow essentially the same attack strategy as used in our KE separation result; the main difference lies in actually analyzing the attack algorithm in our setting, as presented subsequently. However, we summarize the attack strategy here for the sake of completeness.

The attack algorithm is parameterized by some constant $\epsilon > 0$, which we assume is smaller than $1/10$. Let (i, j) denote some sub-round of the 2-PC protocol, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of messages between Alice and Bob until sub-round (i, j), and let $P_E^{(i,j)}$ denote the set of $(k+1)$-restricted $\mathrm{SCMA}_{\text{2-PC}}$ oracle query-answer pairs until sub-round (i, j) asked by Eve. At this point, Eve proceeds as follows during sub-round (i, j):

- If $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] = 0$, Eve aborts.

[1] Note that the number of queries made by Eve when attacking the 2-PC protocol with equivalence complete query pattern is actually independent of k; the factor of $\mathrm{poly}(k)$ blowup in the number of queries over and above any 2-PC protocol (as in the statement of Theorem 4.12) is already implicit in the number of queries n_A and n_B in the statement of Theorem 4.23.

- Otherwise, as long as there is a query $q = (s, x)$ for $s \in \Sigma^{k+1}$ and x such that $\mathsf{Level}(x) \neq -1$ such that
$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)} [q \in \mathcal{Q}(V_A^{(i,j)})] > \frac{\epsilon}{n_B},$$
or
$$\Pr_{\left(V_A^{(i,j)}, V_B^{(i,j)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)} [q \in \mathcal{Q}(V_B^{(i,j)})] > \frac{\epsilon}{n_A},$$
Eve issues the lexicographically first such query q to the $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle and adds the query-response pair $(q, \mathbf{M}(q))$ to $P_E^{(i,j)}$.
- Eve continues in this way until there remains no additional query that Eve can ask, at which point she stops and waits for the next sub-round to commence.

Eventually, at the end of all sub-rounds of the final round $2k$ (when Eve is also done with asking her oracle queries), Eve samples
$$\left(V_A^{(2k)}, V_B^{(2k)}\right) \leftarrow \mathcal{V}\left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right),$$
computes Alice's input in_A determined by $V_A^{(2k)}$, and outputs in_A as its own output.

Note that Eve's algorithm above may ask much more than $n_A n_B$ queries. However, we will show that the probability that Eve needs to ask more than $O(n_A n_B/\epsilon^2)$ queries is bounded by $O(\epsilon)$, and hence we can stop Eve after asking these many queries without changing significantly her success probability.

Remark 4.25. As in the case of the attack algorithm of [BM09] and also our attack algorithm for our KE separation result, our attacking algorithm above is not computationally efficient, as in general computing the probability distribution $\mathcal{V}\left(\mathsf{m}^k, P_E^{(k)}\right)$ could be a hard problem since it involves "inverting" the algorithms of Alice and Bob to a certain extent. But because computing these probabilities is in #**P** we can use known techniques to approximate them with arbitrarily good precision using an NP-oracle. In particular this means that our attacker (as was the case in previous works) is computationally efficient in a relativized world in which **P** = **NP**, and hence our result also rules out relativizing reductions from any $(k+1)$-restricted SCMA$_{\text{2-PC}}$ to $2k$-round 2-PC (and hence, relativizing reductions from $(2k+1)$-round 2-PC to $2k$-round 2-PC).

Analyzing Events. Our target is to prove Theorem 4.23. To do so, we first analyze some events for any $2k$-round 2-PC protocol with equivalence complete query pattern. Recall that the event Good_0 holds if Eve has found all of the intersection queries, while event Good_1 holds if Eve has found all of the intersection *and* equivalence queries. We now state and prove the following lemma.

Lemma 4.26 ($\mathsf{Good}_0 \implies \mathsf{Good}_1$ **(Informal)).** *For any 2-PC protocol with equivalence complete query pattern as described above, the event* Good_0 *holds if and only if the event* Good_1 *holds. In other words, if Eve finds all of the intersection queries during an execution of the 2-PC protocol, it also finds all of the equivalence queries during the same execution of the 2-PC protocol.*

More formally, we state and prove the following.

Lemma 4.27 ($\mathsf{Good}_0 \implies \mathsf{Good}_1$ **(Formal)).** *Given any* 2*-PC protocol with equivalence complete query pattern as described above, let* (i, j) *denote some sub-round, let* $\mathsf{m}^{(i,j)}$ *denote the corresponding set of exchanged messages until sub-round* (i, j)*, and let* $P_E^{(i,j)}$ *denote some sequence of* $(k + 1)$*-restricted* SCMA$_{\text{2-PC}}$ *oracle query-answer pairs until sub-round* (i, j)*, such that we have* $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$*. Then, we have*

$$\Pr_{\mathcal{E}}[\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right) |\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)] = 1.$$

Let $\mathcal{V}(\mathsf{m}^{(i,j)})$ denote the conditional distribution of Alice's and Bob's views in the eyes of the attacker Eve who knows the public messages exchanged between Alice and Bob, and has learned all $(k + 1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs described in $P_E^{(i,j)}$. Finally, let $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ denote the distributions obtained by conditioning the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ on the events $\mathsf{Good}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ and $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$, respectively. Then, assuming Lemma 4.27, we also immediately obtain the following corollary.

Corollary 4.28. $\mathcal{GV}_0\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ *and* $\mathcal{GV}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ *are identical.*

Proof. Lemma 4.27 follows immediately from Lemmas 4.20 and 4.21. □

We define two additional events, which we call *fail* event and *long* event.

Fail Event. Given any $2k$-round 2-PC protocol with equivalence complete query pattern, let (i, j) denote some sub-round, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of exchanged messages until sub-round (i, j), and let $P_E^{(i,j)}$ denote the sequence of $(k + 1)$-restricted SCMA$_{\text{2-PC}}$ oracle query-answer pairs made by Eve until sub-round (i, j), such that we have $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$. We define the event $\mathsf{Fail}^{(i,j)}$ to be the event that:

- **EITHER** the query (made by Alice or Bob) to the $(k + 1)$-restricted SCMA$_{\text{2-PC}}$ oracle after this sub-round is an intersection query but is not contained in $P_E^{(i,j)}$.
- **OR** the query (made by Alice or Bob) to the $(k + 1)$-restricted SCMA$_{\text{2-PC}}$ oracle after this sub-round is an equivalence query w.r.t. some query issued earlier by the other party, but $P_E^{(i,j)}$ does not contain a query that is either identical or equivalent to this query,

and this is the first instance of Eve missing either an intersection query or an equivalence query. Let the event $\mathsf{Fail} = \bigvee_{(i,j)} \mathsf{Fail}^{(i,j)}$ be the event that at some point during the $2k$-round 2-PC protocol with equivalence query pattern, an intersection query is missed by Eve.

Long Event. We also denote by Long the event that Eve makes more than $O(n_A n_B/\epsilon^2)$ queries when attacking any $2k$-round 2-PC protocol with equivalence complete query pattern.

Theorem 4.23 immediately follows from the following lemmas.

Lemma 4.29 (Attack is successful). *For any sub-round* (i, j) *of the* 2*-PC protocol with equivalence complete query pattern, we have*

$$\Pr_{\mathcal{E}}[\mathsf{Fail}^{(i,j)}] = O\left(\frac{\epsilon}{(n_A + n_B)}\right).$$

Hence, by union bound, we have $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$*.*

Lemma 4.30 (Attack is efficient). *We have* $\Pr_{\mathcal{E}}[\mathsf{Long}] = O(\epsilon)$.

We prove Lemma 4.29 by proving the following stronger result.

Lemma 4.31. *For any sub-round* (i, j) *of the* 2*-PC protocol with equivalence complete query pattern, let let* $\mathsf{m}^{(i,j)}$ *denote the corresponding set of exchanged messages until sub-round* (i, j)*, and let* $P_E^{(i,j)}$ *denote the sequence of* $(k+1)$*-restricted* $\mathrm{SCMA}_{\text{2-PC}}$ *oracle query-answer pairs made by Eve until sub-round* (i, j)*, such that we have* $\Pr_{\mathcal{E}}[\mathsf{m}^{(i,j)}, P_E^{(i,j)}] > 0$*. Then we have*

$$\Pr_{\mathcal{E}}\left[\mathsf{Fail}^{(i,j)} | \mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] = O\left(\frac{\epsilon}{(n_A + n_B)}\right).$$

To see why Lemma 4.31 implies Lemma 4.29, observe that $\mathsf{Fail}^{(i,j)}$ is the event that Eve fails to query an intersection query or an equivalence query for the first time in sub-round (i, j), and hence, Eve found all intersection queries and equivalence queries during the execution up until sub-round (i, j), meaning that $\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ holds. Hence, we must have

$$\Pr_{\mathcal{E}}[\mathsf{Fail}^{(i,j)}] \leq \Pr_{\mathcal{E}}\left[\mathsf{Fail}^{(i,j)} | \mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] = O\left(\frac{\epsilon}{(n_A + n_B)}\right),$$

which is precisely the statement of Lemma 4.29. We prove Lemma 4.31 by using a product characterization of the distribution $\mathcal{GV}_1$. The proof is very similar to the proof of Lemma 3.69 that use for our KE separation result, and is hence not detailed here.

Proof of Lemma 4.30: The Attack is Efficient. We follow a strategy similar to our separation result for 2-party NIKE to prove that the attack is efficient by crucially relying on the fact that the attack is successful. Recall that in her algorithm, Eve follows the following strategy: at any given sub-round of the protocol, Eve keeps making the lexicographically first query q that has "significant" probability of appearing in either Alice's query set or Bob's query set, until all such queries are exhausted. Also recall that this probability is based on the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ (where $\mathsf{m}^{(i,j)}$ denotes the set of messages exchanged between Alice and Bob until sub-round (i, j), and $P_E^{(i,j)}$ denotes the set of $(k+1)$-restricted $\mathrm{SCMA}_{\text{2-PC}}$ oracle query-answer pairs until sub-round (i, j) asked by Eve), conditioned on the event that Eve has not missed any intersection or equivalence queries up until this point (i.e. the event Good_1). Now, since we have proven that the event Good_1 happens with high probability (Lemma 4.27), this implies that queries with a significant probability of occurrence according the distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$ conditioned on Good_1 also have a significant probability of occurrence under the real distribution $\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)$. Intuitively, we use this to bound the number of queries that Eve has to make by arguing that each query that Eve makes decreases the (nonzero) expected number of unknown queries.

A Bad Event. For the formal proof, we begin by defining an additional event, which we refer to as a "bad" event. Let (i, j) denote some sub-round of the 2-PC protocol, let $\mathsf{m}^{(i,j)}$ denote the corresponding set of messages between Alice and Bob until sub-round (i, j), and let $P_E^{(i,j)}$ denote some sequence of $(k+1)$-restricted $\mathrm{SCMA}_{\text{2-PC}}$ oracle query-answer pairs until sub-round (i, j) learned by Eve. We use $\mathsf{Bad}^{(i,j)}$ to denote the event that

$$\Pr_{\mathcal{V}\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)}\left[\neg\mathsf{Good}_1\left(\mathsf{m}^{(i,j)}, P_E^{(i,j)}\right)\right] > \frac{1}{2}.$$

We also define the probability space $\widehat{\mathcal{E}}$ to denote the same execution probability space as $\mathcal{E}$ with the difference that for any sub-round (i, j), Eve stops asking more queries at sub-round (i, j) if the event $\mathsf{Bad}^{(i,j)}$ occurs (the behavior of Alice and Bob remains unchanged). Note that $\mathcal{E}$ and $\widehat{\mathcal{E}}$ are identical as long as $\mathsf{Bad}^{(i,j)}$ does not happen, and so we have

$$\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}].$$

More generally speaking, for any event D whose definition depends on the behavior of Eve, we have

$$\Pr_{\mathcal{E}}[\mathsf{Bad} \vee \mathsf{D}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad} \vee \mathsf{D}].$$

The proof of Lemma 4.30 follows from the following steps:

- **Step-1:** We first show the following:

$$\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon) \implies \Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon).$$

  Since our analysis of the success probability of the attack already established that $\Pr_{\mathcal{E}}[\mathsf{Fail}] = O(\epsilon)$ (Lemma 4.29), we have

$$\Pr_{\mathcal{E}}[\mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}] = O(\epsilon).$$

- **Step-2:** We then show the following: $\Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}] = O(\epsilon)$.

Observe that

$$\Pr_{\mathcal{E}}[\mathsf{Long}] \le \Pr_{\mathcal{E}}[\mathsf{Long} \vee \mathsf{Bad}] = \Pr_{\widehat{\mathcal{E}}}[\mathsf{Long} \vee \mathsf{Bad}] \le \Pr_{\widehat{\mathcal{E}}}[\mathsf{Long}] + \Pr_{\widehat{\mathcal{E}}}[\mathsf{Bad}].$$

Hence, we have $\Pr_{\mathcal{E}}[\mathsf{Long}] = O(\epsilon)$, which is precisely the statement of Lemma 4.30. The detailed proof is very similar to the proof of Lemma 3.68 that we use for our KE separation result, and is hence not detailed.

Finishing the Attack: Eve finds Alice's Input. Finally, we formally prove that Eve actually finds the Alice's private input. For any triple of the form (V_A, V_B, V_E), we say that:

- the event $\mathsf{Good}_0\left(V_A, V_B, V_E\right)$ holds if $\mathcal{Q}\left(V_A\right)$ and $\mathcal{Q}\left(V_B\right)$ have no intersection query that does not also appear in V_E, and
- the event $\mathsf{Good}_1\left(V_A, V_B, V_E\right)$ holds if $\mathcal{Q}\left(V_A\right)$ and $\mathcal{Q}\left(V_B\right)$ have no intersection query that does not appear in V_E *and* no equivalence query-pair such that V_E does not have a corresponding query equivalent to this pair.

The proof of the fact that Eve finds Alice's input now follows from the following claims.

Claim 4.32. *We claim that* $\Pr[\neg\mathsf{Good}_1\left(V_A^{(2k)}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon)$.

Proof. The proof of this claim follows immediately from the proof of Theorem 4.23. □

Claim 4.33. *We claim that* $\Pr[\neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon)$.

Proof. We argue this claim as follows. It follows from Lemmas 4.20 and 4.21 that for any $2k$-round 2-PC protocol with equivalence complete query pattern,

$$\Pr\left[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right) \middle| \neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right] = 1,$$

and hence

$$\Pr\left[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right] = \Pr\left[\neg\mathsf{Good}_1\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)\right].$$

Now suppose we fix $V_E^{(2k)} = \left(\mathsf{m}^{(2k)}, P_E^{(2k)}\right)$ and sample $\widehat{V}$ as above. Then $\widehat{V}$ is independent of $V_B^{(2k)}$, and hence, any query q such that $q \in \mathcal{Q}\left(V_B^{(2k)}\right)$ and $q \notin \mathcal{Q}\left(V_E^{(2k)}\right)$ has probability at most ϵ/n_B of appearing in $\mathcal{Q}\left(\widehat{V}\right)$ (this follows from Eve's strategy of choosing queries in the attack). Hence, we must have

$$\Pr[\neg\mathsf{Good}_0\left(\widehat{V}, V_B^{(2k)}, V_E^{(2k)}\right)] = O(\epsilon),$$

which completes the proof of this claim. □

Finally, it follows from the above claims that, during the 2-PC protocol with equivalence complete query pattern, if Alice issued an intersection/equivalence query of the form $\left((ab)^{k+1}, x\right)$, then Eve must have issued either the same query or an equivalent query, which allows it to recover a, and hence, the input in_A for Alice. This completes the proof of successful input recovery by Eve.

Remark 4.34. Since our definition of the SCMA$_{\text{2-PC}}$ oracle currently models 2-PC protocols for symmetric functionalities, our proof as described below works for 2-PC protocols supporting symmetric functionalities. In Section 4.4, we discuss how to generalize the definition of SCMA$_{\text{2-PC}}$ oracle to additionally model 2-PC protocols for asymmetric functionalities. The rest of our proof strategy generalizes in a straightforward manner.

4.3 Separating $(2k-1)$-round 2-PC from $2k$-round Maliciously Secure 2-PC

In this section, we argue that we can also black-box separate any $(2k-1)$-round 2-PC protocol from any $2k$-round maliciously secure 2-PC protocol. The argument is almost identical to the separation of $2k$-round 2-PC from $(2k+1)$-round maliciously secure 2-PC, with the exception of some minor tweaks to the $(k+1)$-commutator property of a $(k+1)$-restricted SCMA$_{\text{2-PC}}$ oracle, and our core argument that for any 2-PC protocol with equivalence complete query pattern, each equivalence query is also essentially an intersection query. The rest of the proof structure as well as the arguments surrounding attack success, attack efficiency, and the probability that Eve finds Alice's input, remain essentially unchanged.

Changing the k-Commutator Property Slightly. For $k \geq 1$, suppose that we tweak the k-commutator property of a $(k+1)$-commutator SCMA$_{\text{2-PC}}$ oracle $\mathbf{M}(\cdot,\cdot)$ slightly as follows: instead of requiring that $\mathbf{M}((ab)^{k+1}, x_0) = \mathbf{M}((ba)^{k+1}, x_0)$ (x_0 being the base set element), we now require that

$$\mathbf{M}(b\|(ab)^k, x_0) = \mathbf{M}(a\|(ba)^k, x_0)$$

It is easy to see that in this case, a $(k+1)$-commutator oracle implies a $2k$-round 2-PC protocol as follows:

- Given a base element x_0, Alice would sample some $a \in M$ and obtain $\mathbf{M}(a, x_0)$, while Bob would sample some $b \in M$ and obtain $\mathbf{M}(b, x_0)$. Alice and Bob would then exchange their first-round messages, where Alice sends $\mathbf{M}(a, x_0)$ to Bob and Bob sends $\mathbf{M}(b, x_0)$ to Alice.
- In the next round, Alice would obtain $\mathbf{M}(ab, x_0) = \mathbf{M}\left(a, \mathbf{M}\left(b, x_0\right)\right)$, and Bob would obtain $\mathbf{M}(ba, x_0) = \mathbf{M}\left(b, \mathbf{M}\left(a, x_0\right)\right)$. Alice and Bob would then exchange their second-round messages, where Alice sends $\mathbf{M}(ab, x_0)$ to Bob and Bob sends $\mathbf{M}(ba, x_0)$ to Alice.

Observe that by repeating this process for $2k$ rounds and asking a final query to the $(k+1)$-SCMA oracle, Alice and Bob would have obtained $\mathbf{M}(a\| \left(ba\right)^k, x_0) = \mathbf{M}(b\| \left(ab\right)^k, x_0)$, which they can use as the output. Note that this computation requires the full $2k$ rounds[1].

Arguing Impossibility of $(2k-1)$**-round** 2**-PC.** Now let's look at what happens if Alice and Bob try to exploit the "commutative" property of the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle in less than $2k$ rounds. Again, they must generate some equivalence query-pair of the form $\mathbf{M}(a\|(ba)^k, x_0) = \mathbf{M}(b\|(ab)^k, x_0)$ with less than $2k$ rounds of communication. Once again, note that when "building up" to such an equivalence query that gives Alice and Bob the same final set element via two different query sequences in less than $2k$ rounds, Alice and Bob cannot only issue queries to the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle, where the monoid element is either a or b like in the $2k$-round 2-PC protocol outlined above. In particular, by the pigeonhole principle, at least one of Alice or Bob must compute a query involving both the elements a and b.

At this point, we can the same core argument as in the separation of $2k$-round 2-PC from $(2k+1)$-round 2-PC to establish that even in this case, as long as the $(2k-1)$-round 2-PC protocol is in a special form that "forces" Alice and Bob to make all "split" versions of their queries and at least one of Alice or Bob to compute all possible ways of computing an equivalence query as soon as there is a "trigger" query where the monoid element is a substring of either $(ab)^k$ or $(ba)^k$, any equivalence query w.r.t. the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle that can be computed within $(2k-1)$ rounds is also an intersection query.

This again effectively reduces all equivalence queries that rely on the (modified) commutative property of the $(k+1)$-SCMA$_{\text{2-PC}}$ oracle to the "traditional" notion of intersection queries, and we can again handle such queries using the [BM09] framework. More concretely, the rest of the proof structure as well as the arguments surrounding attack success, attack efficiency, and the probability that Eve finds Alice's input, remain essentially unchanged.

4.4 Generalization to 2-PC Protocols for Asymmetric Functionalities

Our impossibility results so far have assumed 2-PC protocols for symmetric functionalities where both parties Alice and Bob receive the same output, computed by evaluating a single function f on their inputs in_A and in_B. In this section, we discuss how to generalize our separation result to 2-PC protocols for *asymmetric functionalities* where Alice and Bob receive different outputs $f_A(\mathsf{in}_A, \mathsf{in}_B)$ and $f_B(\mathsf{in}_A, \mathsf{in}_B)$ (including protocols where one of the participants may not receive any output).

"Asymmetric" DCMA$_{\text{2-PC}}$**.** Informally speaking, the generalization essentially uses a slight structural tweak to our definition of the commutative monoid action for 2-PC (ℓ-DCMA$_{\text{2-PC}}$, Definition 4.1) wherein we encode the functions for Alice and Bob (which are different) as part of the same monoid element, except that

[1] We again note that if M is a countably infinite set, then a uniform distribution over M is not well-defined; in this case, we restrict to those distributions for which the set of all strings consisting of more than $2k$ elements has negligible density in the sample space.

the order in which they are encoded differs for Alice and Bob. In particular, Alice encodes the functions into a monoid element as tuple of the form (f_A, f_B), while Bob encodes it as (f_B, f_A). The modified DCMA$_{\text{2-PC}}$ operates as follows: on input a monoid element that is a tuple of the above form and a set element, it now produces as output a tuple of set elements, where the order of the elements in the tuple depends on the order in which the functions are encoded. Finally, we assume that in the output of the DCMA$_{\text{2-PC}}$, only the first element in each output tuple is received as the output of the query, while the second element remains private. This effectively models the asymmetric nature of the functionality being computed, such that even if Alice and Bob issue a sequence of queries that yields the same tuple of function outputs for both of them (albeit in different order), each party only learns the output of its own functionality. We now formalize this asymmetric ℓ-DCMA$_{\text{2-PC}}$ in Definition 4.35 below.

Definition 4.35 (Asymmetric ℓ-DCMA$_{\text{2-PC}}$). A monoid action $(M, X, \star)$ is an **asymmetric** ℓ-DCMA$_{\text{2-PC}}$ if it satisfies following additional structural properties:

- The monoid $(M, \oplus)$ is a string concatenation monoid structured as $M = M_A \cup M_B$ where

$$M_A = I \times F \times R_A, \quad M_B = I \times F \times R_B,$$

  such that both of the sub-monoids M_A and M_B are individually string concatenation monoids themselves, and F consists of tuples of all possible functions of the form (f_A, f_B).

- The set X is structured as

$$X = PP \times \left(\bigcup_{i \in [\ell]} S_{i,A} \cup \bigcup_{i \in [\ell]} S_{i,B} \cup \{\bot\} \right) \times (Y \cup \{\bot\}) \times (Y \cup \{\bot\}) .$$

- For any public parameters $\mathsf{pp} \in PP$, any pair of inputs $\mathsf{in}_A, \mathsf{in}_B \in I$, any tuple of functions $(f_A, f_B) \in F$, and any pair of randomnesses $(r_A, r_B) \in R_A \times R_B$, letting

$$\begin{aligned} g &= (\mathsf{in}_A, (f_A, f_B), r_A) \in M_A, \quad h = (\mathsf{in}_B, (f_B, f_A), r_B) \in M_B, \\ x &= (\mathsf{pp}, \bot, \bot, \bot) \in X, \quad y_A = f_A(\mathsf{in}_A, \mathsf{in}_B), \quad y_B = f_B(\mathsf{in}_A, \mathsf{in}_B). \end{aligned}$$

  we have

$$(g \oplus h)^\ell \star x = (\mathsf{pp}, \bot, (y_A, y_B)), \quad (h \oplus g)^\ell \star x = (\mathsf{pp}, \bot, (y_B, y_A)).$$

Generic k-restricted Asymmetric SCMA$_{\text{2-PC}}$ **Oracle.** We similarly tweak the definition of the commutator property of a generic k-restricted SCMA$_{\text{2-PC}}$ oracle to incorporate such an asymmetric functionality as follows.

Definition 4.36 (k'-Asymmetric Commutator k-restricted SCMA$_{\text{2-PC}}$ **Oracle).** A generic k-restricted SCMA$_{\text{2-PC}}$ oracle over an alphabet $\Sigma = \Sigma_A \cup \Sigma_B$ with initial element x_0 is said to be a k'-asymmetric commutator (for $k' \in [1, k]$) if for any $a \in \Sigma_A, b \in \Sigma_B$, there exists $y_a, y_b \in \{0,1\}^*$ such that we have

$$\mathbf{M}\left((ab)^{k'}, x_0\right) = y_a \| y_b, \quad \mathbf{M}\left((ba)^{k'}, x_0\right) = y_b \| y_a.$$

Lemma 4.37. *There exists a construction of $(2k-1)$-round 2-PC protocol for asymmetric functionalities satisfying malicious security with abort from any k-restricted SCMA oracle $\mathbf{M}(\cdot, \cdot)$ over a sufficiently large alphabet Σ.*

Proof. The proof of this lemma is very similar to the proof of Lemma 4.37, with some minor modifications to incorporate the asymmetric nature of the protocol. At a high level, each party encodes (the string representations of) the functions f_A and f_B into a single monoid element, except that party A encodes it as $f_A \| f_B$, while party B encodes it as $f_B \| f_A$. For each query, the oracle checks that the query from each party encodes the same set of functions (arranged in a fixed order as stipulated above depending on which party issues the query), and then, in response to the final query, provides each party with either y_A or y_B depending on whether the response is to the final query from party A or party B (we assume that both parties are not allowed to query the oracle any further once they have issued their final queries). Correctness is immediate, while security also follows since each player is committed to the set of functions (f_A, f_B) in each query, and are hence implicitly in agreement on the output throughout. Finally, the extraction argument works exactly as in the case of Lemma 4.37.

Separating Asymmetric 2-PC by Rounds. Given the above generic k-restricted asymmetric SCMA$_{\text{2-PC}}$ oracle, it is immediate to extend our proof strategy for 2-PC supporting symmetric functionalities to the case of 2-PC supporting asymmetric functionalities. In particular, Eve uses essentially the same attack strategy, and our arguments for it recovering the intersection and equivalence queries remain unchanged. We avoid detailing the whole attack strategy and proof for brevity.

5 On Black-Box Separating Multiparty NIKE

It is natural to ask if our approach to black-box separations using structural characterization extends to other similar cryptographic primitives, such as multiparty noninteractive key exchange (NIKE). More concretely, we ask if there exists a structural characterization of k-party NIKE that would allow us to extend our black-box separation techniques for 2-party KE by rounds (Section 3) and 2-PC by rounds (Section 4) for showing a black-box separation between $(k+1)$-party NIKE and k-party NIKE (for $k \geq 2$). We give evidence that such a characterization is likely to require very different techniques (at least generally for all $k \geq 2$).

Multiparty NIKE. We begin by recalling the definition of a plain multiparty NIKE protocol.

Definition 5.1 (k-party NIKE). An ℓ-noisy k-party NIKE is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Combine})$ defined as follows:

- $\mathsf{Setup}\left(1^\lambda, 1^k\right)$: Takes as input a security parameter λ, and the number of parties k, and outputs a public parameter pp.
- $\mathsf{Gen}(\mathsf{pp}, i \in [k])$: Takes as input a public parameter pp and an index $i \in [k]$, and outputs a (message, state) pair $(\mathsf{m}_i, \mathsf{st}_i)$.
- $\mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j\in[k], j\neq i}\right)$: Takes as input a public parameter pp, an index $i \in [k]$, and a sequence of messages $\{\mathsf{m}_j\}_{j\in[k], j\neq i}$, and outputs a key $\mathsf{k}_i \in \{0,1\}^\lambda$.

We require the following correctness and security properties to be satisfied.

- **Correctness:** For any $\lambda \in \mathbb{N}$, letting

$$\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^k, 1^\ell\right), \quad \{(\mathsf{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)\}_{i\in[k]},$$

and for each $i \in [k]$,letting

$$\mathsf{k}_i = \mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j \in [k], j \neq i}\right),$$

we must have the following: there exists some $\mathsf{k}^* \in \{0,1\}^\lambda$ such that

$$\mathsf{k}_1 = \ldots = \mathsf{k}_k = \mathsf{k}^*.$$

- **Security:** For any $\lambda \in \mathbb{N}$, letting

  $$\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^k, 1^\ell\right), \quad \{(\mathsf{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)\}_{i \in [k]},$$

  and for each $i \in [k]$, letting

  $$\mathsf{k}_i = \mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j \in [k], j \neq i}\right),$$

  such that $\mathsf{k}_1 = \ldots = \mathsf{k}_k = \mathsf{k}^*$, we must have the following: for any passive eavesdropping PPT adversary $\mathcal{A}$, we have

  $$\left|\Pr[\mathcal{A}(\mathsf{m}_1, \ldots, \mathsf{m}_k, \mathsf{k}^*) = 1] - \Pr[\mathcal{A}(\mathsf{m}_1, \ldots, \mathsf{m}_k, \mathsf{k}') = 1]\right| \leq \mathrm{negl}(\lambda),$$

  where $\mathsf{k}' \leftarrow \{0,1\}^\lambda$, and where the probability is taken over the internal random coins of Setup and Gen.

"Noisy" Multiparty NIKE. In particular, we show that (for large enough k), a k-party NIKE protocol black-box implies a slightly weaker "noisy" variant of a $(k+1)$-party NIKE protocol, which we call "2-noisy" NIKE protocol. We formally describe this notion of multiparty NIKE below.

Definition 5.2 (ℓ-noisy k-party NIKE). An ℓ-noisy k-party NIKE is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Combine})$ defined as follows:

- $\mathsf{Setup}\left(1^\lambda, 1^k, 1^\ell\right)$: Takes as input a security parameter λ, the number of parties k, and the "noise" parameter ℓ, and outputs a public parameter pp.

- $\mathsf{Gen}(\mathsf{pp}, i \in [k])$: Takes as input a public parameter pp and an index $i \in [k]$, and outputs a (message, state) pair $(\mathsf{m}_i, \mathsf{st}_i)$.

- $\mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j \in [k], j \neq i}\right)$: Takes as input a public parameter pp, an index $i \in [k]$, and a sequence of messages $\{\mathsf{m}_j\}_{j \in [k], j \neq i}$, and outputs a list of ℓ keys $(\mathsf{k}_{i,1}, \ldots, \mathsf{k}_{i,\ell}) \in \left(\{0,1\}^\lambda\right)^\ell$.

We require the following correctness and security properties to be satisfied.

- **ℓ-"noisy" correctness:** Informally, an ℓ-noisy k-party NIKE is said to satisfy ℓ-"noisy" correctness if at least one of the ℓ keys received by each party is guaranteed to be shared by all parties, and hence can be treated as the shared secret key. Formally, for any $\lambda \in \mathbb{N}$, letting

  $$\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^k, 1^\ell\right), \quad \{(\mathsf{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)\}_{i \in [k]},$$

  and for each $i \in [k]$,letting

  $$(\mathsf{k}_{i,1}, \ldots, \mathsf{k}_{i,\ell}) = \mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j \in [k], j \neq i}\right),$$

  we must have the following: there exists a key $\mathsf{k}^* \in \{0,1\}^\lambda$ and there exist indices $j_1, \ldots, j_\ell \in [\ell]$ s.t.

  $$\mathsf{k}_{i,j_1} = \mathsf{k}_{2,j_2} = \ldots = \mathsf{k}_{i,j_1} = \mathsf{k}^*.$$

- **Security:** Informally, an ℓ-noisy k-party NIKE is said to be secure if a passive eavesdropping (computationally bounded) adversary cannot predict (with non-negligible property) *any* of the ℓ candidate keys received by each party. Formally, for any $\lambda \in \mathbb{N}$, letting

$$\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^k, 1^\ell\right), \quad \{(\mathsf{m}_i, \mathsf{st}_i) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)\}_{i \in [k]},$$

  and for each $i \in [k]$, letting

$$(\mathsf{k}_{i,1}, \ldots, \mathsf{k}_{i,\ell}) = \mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j \in [k], j \neq i}\right),$$

  we must have the following: for each $i \in [k]$, each $j \in [\ell]$, and any passive eavesdropping PPT adversary $\mathcal{A}$, we have

$$\Pr[\mathcal{A}(\mathsf{m}_1, \ldots, \mathsf{m}_k) = \mathsf{k}_{i,j}] \leq \mathrm{negl}(\lambda),$$

  where the probability is taken over the internal random coins of Setup and Gen.

Remark 5.3. For many practical applications (such as encryption), an ℓ-NIKE protocol in conjunction with a random oracle offers the same functionality as a regular NIKE protocol, albeit inefficiently. We illustrate this using the example of encryption below.

Application of "noisy" multiparty NIKE for Encryption. We illustrate how a k-party ℓ-noisy NIKE protocol can be used to enable (symmetric-key) encryption. Party-i proceeds as follows upon receiving the set of keys $(\mathsf{k}_{i,1}, \ldots, \mathsf{k}_{i,\ell})$ from the NIKE protocol:

- Party-i passes $(\mathsf{k}_{i,1}, \ldots, \mathsf{k}_{i,\ell})$ through a random oracle H to derive ℓ uncorrelated encryption keys as

$$\mathsf{k}'_{i,1} = H(\mathsf{k}_{i,1}), \ldots, \mathsf{k}'_{i,\ell} = H(\mathsf{k}_{i,\ell}).$$

- Party-i then uses these derived keys $(\mathsf{k}'_{i,1}, \ldots, \mathsf{k}'_{i,\ell})$ to encrypt a message m as a tuple of ciphertexts $(\mathsf{ct}_{i,1}, \ldots, \mathsf{ct}_{i,\ell})$, where

$$\mathsf{ct}_1 = \mathsf{Enc}(\mathsf{k}'_{i,1}, \mathsf{m}), \ldots, \mathsf{ct}_{i,\ell} = \mathsf{Enc}(\mathsf{k}'_{i,\ell}, \mathsf{m}).$$

Note that one of these derived keys is guaranteed to be shared by all parties. Hence, correctness of decryption follows (albeit inefficiently since each party must also decrypt under each derived key) from the noisy correctness guarantee of the ℓ-noisy NIKE protocol, while semantic security follows from the unpredictability guarantee of the ℓ-noisy NIKE protocol and the properties of the random oracle H (concretely, as long as the keys output by the NIKE protocol are sufficiently unpredictable, the corresponding derived keys are sufficiently random under the assumption that H is a random oracle).

Constructing $(k+1)$-party 2-noisy NIKE from k-party NIKE. We now show that, for large enough k, given a k-party (regular) NIKE protocol, there exists a construction of a $(k+1)$-party 2-noisy NIKE satisfying the aforementioned requirements, such that the construction uses the underlying k-party NIKE in a fully black-box manner. Concretely, we state and prove the following theorem:

Theorem 5.4. *For $k = \omega(\log \lambda)$ (λ being the security parameter), a k-party (regular) NIKE protocol satisfying Definition 5.1 implies (in a black-box manner) a $(k+1)$-party 2-noisy NIKE protocol satisfying Definition 5.2.*

Proof. The construction uses, in addition to the k-party (regular) NIKE protocol $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Gen}, \Pi.\mathsf{Combine})$, a randomness extractor $\mathsf{Ext} : \{0,1\}^\lambda \to \{0,1\}$. The construction is as follows:

- $\mathsf{Setup}_{(\ell=2)}\left(1^\lambda, 1^{(k+1)}\right)$: For each $i \in [k+1]$, sample $\mathsf{pp}_i \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$ and output the public parameter
$$\mathsf{pp} = (\mathsf{pp}_1, \ldots, \mathsf{pp}_{k+1}) .$$
- $\mathsf{Gen}(\mathsf{pp}, i \in [k+1])$: For each $j \in [k+1]$ such that $j \neq i$, do the following:
  - If $i < j$, sample $(\mathsf{m}_{i,j}, \mathsf{st}_{i,j}) \leftarrow \Pi.\mathsf{Gen}(\mathsf{pp}_j, i)$.
  - If $i > j$, sample $(\mathsf{m}_{i,j}, \mathsf{st}_{i,j}) \leftarrow \Pi.\mathsf{Gen}(\mathsf{pp}_j, i-1)$.

  Output a (message, state) pair $(\mathsf{m}_i, \mathsf{st}_i)$, where
$$\mathsf{m}_i = (\mathsf{m}_{i,j})_{j\in[k+1], j\neq i}, \quad \mathsf{st}_i = (\mathsf{st}_{i,j})_{j\in[k+1], j\neq i} .$$
- $\mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_i, \{\mathsf{m}_j\}_{j\in[k], j\neq i}\right)$: For each $j \in [k+1]$ such that $j \neq i$, parse
$$\mathsf{m}_j = (\mathsf{m}_{j,j'})_{j'\in[k+1], j'\neq j} .$$
  Now, for each $j' \in [k+1]$ such that $j' \neq i$, recover
$$\mathsf{k}'_{i,j'} = \mathsf{Combine}\left(\mathsf{pp}, \mathsf{st}_{i,j'}, \{\mathsf{m}_{j,j'}\}_{j\in[k+1], j\neq i, j\neq j'}\right),$$
  and set $b_{i,j'} = \mathsf{Ext}\left(\mathsf{k}'_{i,j'}\right)$. Finally, output the key-pair $(\mathsf{k}_{i,0}, \mathsf{k}_{i,1})$, where:
$$\mathsf{k}_{i,0} = (b_{i,1}\| \ldots \| b_{i,i-1} \| 0 \| b_{i,i+1} \| b_{i,k+1}), \quad \mathsf{k}_{i,1} = (b_{i,1}\| \ldots \| b_{i,i-1} \| 1 \| b_{i,i+1} \| b_{i,k+1}) .$$
  Finally, party-i outputs the pair of keys $(\mathsf{k}_{i,0}, \mathsf{k}_{i,1})$.

Correctness and Security. Correctness follows immediately from the correctness of the underlying (regular) k-party NIKE protocol Π. To argue security, we observe that for each $b \in \{0,1\}$, $\mathsf{k}_{i,b}$ is sufficiently unpredictable since: (a) each $\mathsf{k}'_{i,j'}$ is pseudorandom (this follows from the security guarantees of the underlying k-party NIKE protocol Π), and (b) each extracted bit $b_{i,j'}$ is pseudorandom (this follows from (a) and the security guarantees of the random extractor Ext), which in turn implies that for $k = \omega(\log\lambda)$, no PPT adversary can predict either of the final keys $\mathsf{k}_{i,b}$ for $b \in \{0,1\}$ with probability greater than $1/2^k \leq \mathrm{negl}(\lambda)$.

This completes the proof of Theorem 5.4. □

Discussion: Separating Multiparty NIKE by Number of Parties. Note that 2-noisy NIKE does not exactly meet the definition of regular NIKE and thus, our construction above does not necessarily rule out *any* black-box separation of $(k+1)$-party NIKE from k-party NIKE. However, it does offer strong evidence that such a separation will have to rely on very different techniques as compared to the techniques used in our black-box separation proofs, as well as the proof frameworks from [IR89, Rud92, BM09].

We begin by observing that our result indicates that *any* black-box separation of $(k+1)$-party NIKE and k-party NIKE (for large enough k) will have to rely on the distinction between "noise-free" and "noisy" NIKE. Our approach of using structural characterization of primitives for black-box separations was the following: we identified a structured primitive that is equivalent to the "base" cryptoprimitive of interest for the separation, and then argued that a (generic, statistically secure version of) this algebraic structure is not sufficient to realize the "target" cryptoprimitive. Unfortunately, it seems impossible to capture the distinction between "noise-free" and "noisy" NIKE using such a structural characterization (such as one based on "hard" monoid actions). In other words, if there exists a general black-box separation between $(k+1)$-party NIKE and k-party NIKE (and hence between $(k+1)$-party regular NIKE and $(k+1)$-party 2-noisy NIKE), we do not believe that the separation can be explained in terms of the algebraic structure inherent to these primitives.

More generally, it seems difficult to use the black-box separation frameworks from the prior works that we build upon [IR89, Rud92, BM09] to separate $(k+1)$-party NIKE and k-party NIKE. Note that all of these frameworks rely on the fact that an eavesdropping adversary Eve can make all of the queries to the oracle that the honest participants can make. Unfortunately, given a k-party NIKE oracle, any subset of k parties can issue a query to this oracle that Eve provably cannot make (in fact, our construction above crucially exploits this feature). Hence, we believe that a black-box separation of $(k+1)$-party NIKE and k-party NIKE would require entirely new techniques.

References

[ABG$^+$20] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In *ITCS 2020*, pages 71:1–71:18. LIPIcs, January 2020.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020, Part II*, LNCS, pages 411–439. Springer, Heidelberg, December 2020.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

[AJJ12] Gorjan Alagic, Stacey Jeffery, and Stephen P Jordan. Partial-indistinguishability obfuscation using braids. *arXiv preprint arXiv:1212.6458*, 2012.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[AMP19] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Symmetric primitives with structured secrets. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 650–679. Springer, Heidelberg, August 2019.

[AMPR19] Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Arnab Roy. Minicrypt primitives with algebraic structure and applications. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 55–82. Springer, Heidelberg, May 2019.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015.

[AS16] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 512–541. Springer, Heidelberg, January 2016.

[Bar17] Boaz Barak. The complexity of public-key cryptography. In *Tutorials on the Foundations of Cryptography*, pages 45–77. 2017.

[BDV17] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 696–723. Springer, Heidelberg, August 2017.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BHMO18] Amos Beimel, Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. Tighter bounds on multi-party coin flipping via augmented weak martingales and differentially private sampling. In Mikkel Thorup, editor, *59th FOCS*, pages 838–849. IEEE Computer Society Press, October 2018.

[BI87] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes (extended abstract). In *28th FOCS*, pages 118–126. IEEE Computer Society Press, October 1987.

[BKLS24] Elette Boyle, Lisa Kohl, Zhe Li, and Peter Scholl. Direct fss constructions for branching programs and more from prgs with encoded-output homomorphism. Cryptology ePrint Archive, Paper 2024/192, 2024. https://eprint.iacr.org/2024/192.

[BM07] Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th FOCS*, pages 680–688. IEEE Computer Society Press, October 2007.

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 374–390. Springer, Heidelberg, August 2009.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 299–315. Springer, Heidelberg, August 2003.

[BPR$^+$08] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *49th FOCS*, pages 283–292. IEEE Computer Society Press, October 2008.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2020.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[BY91] Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Heidelberg, August 1991.

[CD22] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

[CFM21] Geoffroy Couteau, Pooya Farshim, and Mohammad Mahmoody. Black-box uselessness: Composing separations in cryptography. pages 47:1–47:20. LIPIcs, 2021.

[CFW11] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Adaptive pseudo-free groups and applications. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 207–223. Springer, Heidelberg, May 2011.

[CI14] Andrew M Childs and Gábor Ivanyos. Quantum computation of discrete logarithms in semigroups. *Journal of Mathematical Cryptology*, 8(4):405–416, 2014.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.

[CLM$^+$18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

[Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. https://ia.cr/2006/291.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Heidelberg, November 2017.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[DJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

[DLMM11] Dana Dachman-Soled, Yehuda Lindell, Mohammad Mahmoody, and Tal Malkin. On the black-box complexity of optimally-fair coin tossing. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 450–467. Springer, Heidelberg, March 2011.

[FHKP13] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, February / March 2013.

[Fis00] Marc Fischlin. A note on security proofs in the generic model. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 458–469. Springer, Heidelberg, December 2000.

[Fis12] Marc Fischlin. Black-box reductions and separations in cryptography (invited talk). In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 12*, volume 7374 of *LNCS*, pages 413–422. Springer, Heidelberg, July 2012.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Heidelberg, May / June 2010.

[Gar08] David Garber. Braid group cryptography, 2008.

[GHMM18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ameer Mohammed. Limits on the power of garbling techniques for public-key encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 335–364. Springer, Heidelberg, August 2018.

[GKLM12] Vipul Goyal, Virendra Kumar, Satyanarayana V. Lokam, and Mohammad Mahmoody. On black-box reductions between predicate encryption schemes. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 440–457. Springer, Heidelberg, March 2012.

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000.

[GMM17a] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Heidelberg, August 2017.

[GMM17b] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 82–115. Springer, Heidelberg, November 2017.

[GMMM18] Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 545–574. Springer, Heidelberg, August 2018.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd FOCS*, pages 126–135. IEEE Computer Society Press, October 2001.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008.

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *48th FOCS*, pages 669–679. IEEE Computer Society Press, October 2007.

[HK05] Omer Horvitz and Jonathan Katz. Bounds on the efficiency of "black-box" commitment schemes. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 128–139. Springer, Heidelberg, July 2005.

[HK17] Mohammad Hajiabadi and Bruce M. Kapron. Toward fine-grained blackbox separations between semantic and circular-security notions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 561–591. Springer, Heidelberg, April / May 2017.

[HMO18] Iftach Haitner, Nikolaos Makriyannis, and Eran Omri. On the complexity of fair coin flipping. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 539–562. Springer, Heidelberg, November 2018.

[Hoh03] Susan Rae Hohenberger. *The cryptographic impact of groups with infeasible inversion*. PhD thesis, Massachusetts Institute of Technology, 2003.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 99–108. ACM Press, May 2006.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

[IKSS22] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box secure computation from two-round malicious ot. In *Theory of Cryptography Conference*, pages 441–469, 2022.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, June 1995.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989.

[JQSY19] Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun. General linear group action on tensors: A candidate for post-quantum cryptography. In *TCC 2019, Part I*, LNCS, pages 251–281. Springer, Heidelberg, March 2019.

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 603–648. Springer, Heidelberg, April / May 2018.

[KSY11] Jonathan Katz, Dominique Schröder, and Arkady Yerukhimovich. Impossibility of blind signatures from one-way permutations. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 615–629. Springer, Heidelberg, March 2011.

[Mer78] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[MM11] Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 597–614. Springer, Heidelberg, March 2011.

[MM16] Mohammad Mahmoody and Ameer Mohammed. On the power of hierarchical identity-based encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 243–272. Springer, Heidelberg, May 2016.

[MMN$^+$16] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and abhi shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 49–66. Springer, Heidelberg, January 2016.

[MP12] Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 701–718. Springer, Heidelberg, August 2012.

[MP23] Hart Montgomery and Sikhar Patranabis. A computational category-theoretic approach to cryptography and average-case complexity. *Mathematical Cryptology*, 3(2):24–52, 2023.

[MW20] Hemanta K. Maji and Mingyuan Wang. Black-box use of one-way functions is useless for optimal fair coin-tossing. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2020, Part II*, LNCS, pages 593–617. Springer, Heidelberg, August 2020.

[MW21] Hemanta K. Maji and Mingyuan Wang. Computational hardness of optimal fair computation: Beyond minicrypt. LNCS, pages 33–63. Springer, Heidelberg, 2021.

[oST22] National Institute of Standards and Technology. Call for additional digital signature schemes for the post-quantum cryptography standardization process, 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[PR23] Aurel Page and Damien Robert. Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Paper 2023/1766, 2023. https://eprint.iacr.org/2023/1766.

[PRV12] Periklis A Papakonstantinou, Charles W Rackoff, and Yevgeniy Vahlis. How powerful are the ddh hard groups? *Cryptology ePrint Archive*, 2012.

[Riv04] Ronald L. Rivest. On the notion of pseudo-free groups. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 505–521. Springer, Heidelberg, February 2004.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RSS17] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 747–776. Springer, Heidelberg, November 2017.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2004.

[Rud92] Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 242–251. Springer, Heidelberg, August 1992.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345. Springer, Heidelberg, May / June 1998.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.