PROOF SYSTEMS FOR SCALING BLOCKCHAINS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ben Fisch

August 2022

Abstract

The integrity of record-keeping is essential to a functioning society. Our dependence on digital records has become absolute, from our personal wealth, commerce, and identities to sources of knowledge and news. In recent years, blockchains have arisen as a new form of digital record-keeping that is operated by decentralized sets of participating computers. While traditional record-keeping places trust in a centralized service, blockchains remain secure so long as the number of corrupt computers does not exceed a large threshold, an assumption that can also be strengthened via economic incentives. However, this robustness comes at a cost. Blockchains struggle to process updates at the same rate as standard transactional systems. As a result, rising demand has led to congestion and soaring transaction fees. In this dissertation, we construct several cryptographic proof systems that help scale the throughput of blockchains. The first set of tools we construct are authenticated data-structures (ADSs) and succinct non-interactive arguments of knowledge (SNARKs) that leverage new techniques based on groups of unknown order (GUOs) to achieve significantly lower communication than prior methods, without relying on a so-called *trusted setup*. These tools can be used to reduce the required bandwidth, computation, and storage of nodes participating in the blockchain protocol. However, it is challenging to apply these tools on their own to scale the throughput of blockchains while maintaining the guarantee that critical public data recorded in the blockchain remains available to users. To address this challenge we construct a new type of proof system called *proofs of replication* (PoReps), which help to ensure data is redundantly stored and available.

Acknowledgments

First and foremost, I would like to acknowledge my advisor Dan Boneh for his immeasurable support throughput my PhD studies. Dan was not only a coauthor and collaborator on many of the papers I published over the years, including the results in Chapter 4 of this dissertation, but also mentored me in all facets of life, research, and my career path. Dan was an endless source of inspiration and creative ideas, and was always present to provide helpful feedback and suggestions on my work. Dan also generously helped me foster connections with industry, which led to many fruitful collaborations and even entrepreneurial involvement. Dan has been an incredible role model that I will continue to look up to.

Second, I would like to acknowledge my longtime research collaborator, business colleague, and close friend Benedikt Bünz, with whom I collaborated on many projects, including the results in Chapters 2-4 of this dissertation. Benedikt and I met in Dan's research group during the first year of my PhD. We experienced the PhD journey together and I couldn't ask for a better peer. I look forward to many years of collaboration to come. Benedikt and I met Alan Szepieniec at Eurocrypt 2019, which sparked our collaboration together on the results in Chapter 2. This collaboration spanned several years, as we had to work to resolve a critical bug in our original security analysis of the *DARK* protocol.

I had many other collaborators and mentors throughout grad school. At the end of my first year, Joe Bonneau introduced me to *randomness beacons* and their importance to blockchains. For decades, security researchers have sought to build the ideal *randomness beacon*: a system that regularly publishes random numbers that no party can predict or manipulate. Joe hypothesized the existence of better cryptographic tools related to computational time-stamping that would be helpful in building secure random beacons. Our collaboration resulted in a paper, together with Dan and Benedikt, on a new primitive we called *verifiable delay functions* (VDFs) [37], which was likely the most influential paper I wrote during grad school and also served as a launch pad into many of the later topics that I worked on. VDFs can also be used to construct a basic *proof of replication*, which we included as an application in the paper. Joe explored the concept of provable replication with me before I started working on the topic in more depth and initially suggested looking at using depth robust graphs. The ideas in Chapter 5 were very much inspired by these early discussions with Joe. Beyond research collaboration, Joe provided me with guidance and mentorship that I value greatly. Rafael Pass was also a mentor that I looked up to throughput grad school. Rafael reintroduced me to the research area of blockchains in 2015 and taught me all I know about algorithmic game theory and mechanism design. We collaborated together with Abhi Shelat on a better model and understanding of Bitcoin mining pools [82].

I began collaborating with Nicola Greco and Juan Benet on the *Filecoin* project and Protocol Labs in 2017, and both provided critical input and motivation for the work covered in Chapter 5 on *proofs of replication* (PoReps). Protocol Labs partially funded this research and provided many speaking opportunities, but most importantly, implemented PoReps integrated them into *Filecoin*, which was ultimately deployed in 2020. Today, *Filecon* runs in a decentralized storage network that houses more than 16 exabytes of data. Irene Giacomelli and Luca Nizzardo from Protocol Labs also reviewed the security proofs contained in this work and identified bugs in earlier versions of the report.

The list of people who collaborated with me throughput my research career leading up to this point is long and I cannot do justice to everyone on this list. I would like to thank these collaborators, as well as all the members of the applied cryptography group at Stanford, for helping to shape my experience. In addition to those already mentioned, this includes Justin Drake, Ariel Gabizon, Nirvan Tyagi, Stefano Tessaro, Ethan Cecchetti, Ian Miers, Ari Juels, Saba Eskandarian, Shashwat Silas, Dhinakaran Vinayagamurthy, Sergey Gorbunov, Krzysztof Pietrzak, Binyi Chen, Alex Xiong, Philippe Camacho, Alex Xiong, Zhenfei Zhang, Fernando Krell, Vlad Kolesnikov, Tal Malkin, Binh Vo, Abhishek Kumarasubramanian, Steven Bellovin, David Wu, Henry Corrigan-Gibbs, Valeria Nikolaenko, Yan Michalevsky, Sam Kim, Florian Tramer, Dima Kogan, Riad Wahby, Alex Ozdemir, Charles Lu, Daniel Freund, and Moni Naor. I would like to thank Moni Naor in particular, who introduced me to the field of cryptography and inspired me to pursue a PhD in the field.

Finally, I have my friends and family to thank for their relentless support over the years, and especially over the last several years as I was pursuing this PhD. While far from a comprehensive list, I would like to thank my parents Tobe and Nathaniel Fisch, my grandparents, my uncle Michael Mann, my aunt Renee Emunah, cousins Gavi and Melea, my brothers Mendy and Adam, Andrey Poletayev, AJ Kantor, Ben Philipson, Olaf Sakkers, and Laura Symul.

The work in this dissertation was funded in part by an NSF Graduate Research Fellowship and a Stanford Graduate Fellowship.

Bibliographic notes

This dissertation is based on the following (jointly authored) publications:

Chapter 2 "Transparent SNARKs from DARK compilers", by Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Published in *Eurocrypt*, 2020 [53].

Chapter 3 "Schwartz-Zippel for multilinear polynomials mod N", by Benedikt Bünz, and Ben Fisch. Manuscript in IACR Cryptology ePrint Archive, 2022 [52].

Chapter 4 "Batching techniques for accumulators with applications to IOPs and stateless blockchains", by Dan Boneh, Benedikt Buünz, and Ben Fisch. Published in *Crypto*, 2019 [41].

Chapter 5 "Tight proofs of space and replication", by Ben Fisch. Published in *Eurocrypt*, 2019 [81].

Contents

Chapter 1

Introduction

In recent years, there has been a proliferation of deployed systems, colloquially called *blockchains*, that aim to achieve the following abstract goals: (1) there is a *consensus* protocol through which participating servers agree upon and replicate an ordered log of transactions without reliance on a trusted coordinator, (2) the transaction history cannot be modified without detection by clients, and (3) all clients can verify that transactions in the log follow prespecified rules. The second property is called *immutability* and the third *public verifiability*. A fourth goal that is becoming more pervasive is *privacy*: clients can submit transactions that have hidden content and appear anonymous to all other system participants, including other clients and the servers involved in consensus.

There are a variety of security models for the consensus protocol.[1] All include a mechanism that determines the eligibility of a server to participate, ranging from a simple table of cryptographic public keys to *proof-of-work* mechanisms whereby servers are required to solve cryptographic puzzles in lieu of authentication. The participation mechanism implicitly assigns a weight $w > 0$ to all eligible servers. The consensus protocol is *byzantine fault tolerant* with respect to this weight assignment and a threshold parameter $t < 1/2$ if the system behaves correctly so long as the weighted sum of all servers behaving incorrectly

[1]For a broad overview of consensus protocols in the context of blockchains, see the book manuscript *Foundations of Distributed Consensus and Blockchains* by Elaine Shi, available at https://www.distributedconsensus.net.

does not exceed t. This correctness is formalized in two properties: *consistency*, the property that all correct nodes replicate the same finalized list of transactions, and *liveness*, the property that a transaction proposed by any node will eventually appear in the logs of all correct nodes.[2] The informal term *blockchain* originated from specific examples of these systems, such as Bitcoin and Ethereum, that utilize a chain of collision-resistant transaction block hashes as a critical datastructure within their consensus protocol. Today, the term is used to describe a much broader collection of systems that do not necessarily feature this datastructure. However, all blockchains use a combination of cryptographic tools (authenticated datastructures, digital signatures, etc), distributed consensus, and peer-to-peer communication protocols to achieve the aforementioned abstract goals. These core building blocks have been studied for decades, and while the renewed attention in recent years fueled by the blockchain industry has brought incremental advancement to these primitives, the novelty of modern blockchains stems mostly from their *scale* of deployment within highly heterogenous and geographically dispersed networks.

Applications Blockchains have been used for a wide range of applications, including digital commodities (i.e., cryptocurrencies) like Bitcoin; payments and record of ownership for more traditional assets such as digital fiat-currency[3], stocks, and property titles; and more elaborate financial protocols, such as algorithmically stabilized currencies, lending protocols, and automated market makers for the exchange of digital assets, collectively known as *decentralized finance* (DeFi). The total amount of user funds deposited into DeFi protocols, also known as *total locked value* (TVL), exceeded 100 billion in 2022.[4] The transaction volume of *non-fungible tokens* (NFTs), which have been used to represent digital collectibles from virtual multiplayer game objects to art, exceeded \$40 billion in 2021.[5] Cryptocurrencies have also notably been used as a vehicle for humanitarian aid. Between

[2]An alternative definition of liveness only requires valid transactions to appear in the log, where validity may depend on where the transaction is ordered in the log. Requiring every transaction to be included in the log is stronger, as invalid transactions can still be interpreted as having no effect on the state.

[3]https://www.circle.com/en/usdc

[4]https://defillama.com/

[5]https://blog.chainalysis.com/reports/chainalysis-web3-report-preview-nfts/

February and March 2022, approximately \$100M in cryptocurrency donations were sent to the government of Ukraine.[6] In 2021, the US Treasury authorized direct aid payments to over 60,000 healthcare workers in Venuezla in the form of cryptocurrency, as an exception to sanctions against the Venezuelan regime.[7]

Scalability challenges *Decentralized* blockchains, which are operated over heterogenous and geographically dispersed networks, face significant scalability challenges compared with more centrally managed transactional databases and computing platforms. The maximum throughput of blockchains like Bitcoin or Ethereum (measured in transactions per second) has remained at least three orders of magnitude lower than that of standard payment systems, such as Visa. Ethereum's throughput has not kept pace with rising user demand, leading to congestion and soaring transaction fees, in some cases over \$100 for relatively simple transactions.

There are a variety of resource constraints that affect the throughput of a blockchain, primarily bandwidth, computation, and storage. Either of these resource constraints may cause a bottleneck, depending on the protocol design and network characteristics.

Communication bottlenecks New blocks of transactions are typically broadcasted to all nodes participating in the consensus protocol before they are confirmed and finalized in the log (or state-machine). In this case, the blockchain throughput, in bytes of transaction data[8] finalized per second, is limited by the bandwidth of the broadcast channel. Moreover, as this channel is designed to be robust in a network where nodes may behave dishonestly, its bandwidth can be significantly lower than the worst-case bandwidth of all participating nodes [91]. In centralized systems, a single high-bandwidth server may deliver all messages, resulting in a broadcast channel with bandwidth equal to the worst-case node. Blockchains

[6]Michael Chobanian, https://www.banking.senate.gov/download/chobanian-testimony-3-17-22

[7]https://www-ft-com.stanford.idm.oclc.org/content/2a271032-35b4-4969-a4bf-488d4e9e3d18

[8]To be more precise, transaction data refers to the information replicated on all participating consensus nodes, as required by *consistency*. The size of this data is well-defined up to constant factors (e.g., ignoring formatting differences and compression). This excludes external data committed by transactions, such as the pre-image of a cryptographic hash included in a transaction. While commitments may ensure agreement on the external content, it does not ensure replication as required by the consistency property.

like Bitcoin or Ethereum implement broadcast with peer-to-peer gossip, where every node relays the message to $k \geq 1$ peers. A higher value of k decreases the latency of the broadcast channel, but lowers the throughput (by approximately a factor k) [159].[9]

Computation bottlenecks In most blockchain designs, all nodes participating in consensus must evaluate the validity of a transaction block before it can be finalized in the log. This can be expensive, depending on the complexity of the validity predicate. In Ethereum, verifying a transaction block involves executing a general purpose virtual machine (the EVM). In other blockchain designs, the consensus protocol exclusively orders transactions. Even so, as most blockchain applications require interpreting transactions as state machine transitions, nodes still need to run these computations in order to make use of the system. The effective throughput is thus still limited by the rate at which nodes can catch up to the current state. Finally, it is often desirable that nodes can securely read from the latest state without fully participating in consensus, known as *light client* verification. This is one reason to require validity checks, as it enables including a commitment to the correct state, such as a Merkle root, in each transaction block.

Storage bottlenecks By design, all nodes participating in consensus replicate the log and this requires an increasing amount of storage over time.[10] A higher throughput also means the storage requirement increases at a faster rate. It is important to distinguish between cold and hot storage requirements. While archival data can be stored cheaply on disk, which is not a limiting resource in practice, state data used in transaction verification typically needs to be stored in a faster-access storage medium (e.g., SSD or RAM). Otherwise, slow storage lookups can result in a computational bottleneck limiting throughput [138].

[9] A spanning tree communication graph can reduce the latency/throughput tradeoff in a gossip protocol, but even with periodic refreshing it is less robust to dynamic corruption than flooding or random peer selection.

[10] The current size of the Ethereum state stored by the default client (geth) is over 600 GB, and has grown at least linearly over time, https://etherscan.io/chartsync/chaindefault.

1.1 Scaling blockchains with authenticated datastructures and SNARKs

A *succinct non-interactive argument of knowledge* (SNARK) [35] is a proof, in the form of a succinct message, that demonstrates the correctness of a computation output in a way that can be verified much more efficiently than running the computation itself. A *zero knowledge* SNARK, or zk-SNARK, additionally hides information about the internal steps of the computation, or even secret inputs to the computation. More precisely, a SNARK provides for any predicate $\phi : \mathcal{X} \times \mathcal{W} \to \{0,1\}$ a prover algorithm $\mathcal{P}$, which on inputs $(x, w) \in \mathcal{X} \times \mathcal{W}$ outputs a proof π, and a verifier algorithm $\mathcal{V}$, which either accepts or rejects the pair (x, π). $\mathcal{V}$ always accepts (x, π) generated by $\mathcal{P}$ on valid inputs (x, w) such that $\phi(x, w) = 1$. As for security, it should be computationally infeasible to compute (x^*, π^*) that $\mathcal{V}$ accepts if there does not exists a w^* for which $\phi(x^*, w^*) = 1$. In a zk-SNARK, the proof π also reveals no information about w.

In fact, SNARKs satisfy an even stronger property: any $\mathcal{P}^*$ which succeeds in producing a (x^*, π^*) that $\mathcal{V}$ accepts must *know* an input w^*, called a *witness*, for which $\phi(x^*, w^*) = 1$. This *knowledge* is defined as the existence of a *knowledge extractor*, an algorithm that can use $\mathcal{P}^*$ and its inputs to find w^* with high probability. To understand why this matters, consider an example where the input w represents the current state of a state-machine, $x = (\mathsf{tx}, h)$ includes a transaction and a cryptographic hash $h = H(w)$ of the current state w, while $\phi(x, w)$ checks that $h = H(w)$ and evaluates the correctness of tx in state w. Using SNARKs, a verifier can thus receive proofs of transaction validity while only needing to store a hash of the current state. If tx is invalid for w, there may still exist a collision $w' \neq w$ such that tx is valid for w' and $H(w') = H(w)$, and thus $\phi(x, w') = 1$. However, by the security property of the SNARK, computing a valid proof for tx invalid in state w implies the ability to extract the collision $w' \neq w$ with high probability, which should be computationally infeasible for a cryptographically secure hash function.

An *authenticated datastructure* (ADS) [125] provides a communication efficient way to commit to structured data and authenticate query/responses to its content. In more detail, an ADS combines a datastructure (e.g., set, list, tree, key/value map) with three algorithms:

- The Commit algorithm generates a succinct digest that is a binding commitment to the state of the ADS.
- The Authenticate algorithm produces a proof for a query/response to the state of the ADS (e.g., "x is a member of the set").
- The Verify algorithm verifies a proof for a query/response against a digest.

As a security property, it is computationally infeasible to generate valid proofs for conflicting responses to the same query that the Verify algorithm accepts with non-negligible probability. A dynamic ADS also provides a method to update the digest and authentication proofs upon updates to the datastructure content more efficiently than recomputing them from scratch. The simplest ADS is a collision-resistant hash function. The small output digest of the hash function is a binding commitment to its input data. However, this is an inefficient way to authenticate queries for membership in a set, as the Verify algorithm needs to receive the entire set. A popular ADS is a Merkle Tree, which is an authenticated list built from a tree of collision-resistant hashes where the root digest is the commitment. The authentication proof for the value at any position in the list consists of $\log n$ digests where n is the length of the list. It can be verified in time $O(\log n)$. An ADS can be viewed as a cryptographic commitment scheme combined with a specialized SNARK. General purpose SNARKs can be used to construct an ADS with asymptotically small and efficiently verifiable proofs for any datastructure, but specialized constructions can be much more efficient.

Authenticated data structures and SNARKs can be used in concert to increase the throughput of a blockchain by alleviating bottlenecks related to computation, storage, and communication.

1.1.1 Computation and storage

An ADS is used to commit to the state of the blockchain in a succinct digest. If all nodes participating in consensus agree on this digest, then a node proposing a new block of transactions may include a digest for the new state and a SNARK proof of block validity, i.e. that the transition is valid and new digest correct, which all other nodes can verify efficiently. In fact, verifying this small proof only requires reading the current state digest and proposed transaction block in addition to the proof, and thus only the succinct state digest needs to be stored in fast-access memory. Nodes may then asynchronously process transactions to update their local state without affecting the critical path of consensus progress. This achieves a similar optimization to blockchain designs that separate ordering of transactions (via consensus) from state computation, with the added advantage that it eliminates spam (i.e., invalid transactions from bloating the log). Furthermore, the asynchronous local computation is streamlined, for example, reduced to writing state changes to storage and checking their consistency with the ADS digest.[11] Finally, it is more compatible with light client verification: any node can read from and verify the current state after downloading it from another (untrusted) node that already computed it.

As a caveat, while verifying a SNARK is cheap, computing the SNARK involves significantly (several orders-of-magnitude) more computation than verifying the original block of transactions. Thus, utilizing SNARKs would only alleviate computational bottlenecks in a heterogenous network (e.g., where several powerful service providers produce transaction blocks and compute the SNARKs, while the majority of nodes participating in consensus are weaker and only verify). If desired, an ADS together with SNARKs can be used to completely separate the tasks of state replication and block production from the task of ordering valid transactions. These tasks can be handled by distinct systems, where one system may be operated by extremely lightweight nodes ordering/validating ADS digests via consensus

[11]Given that the SNARK already demonstrated validity, nodes do not need to run expensive computations to derive local state changes, as they can simply be provided (e.g., broadcasted as part of the block). For example, a computation that can be entirely eliminated is verification of cryptographic signatures. Even checking the consistency of state changes with the new digest can be streamlined with a SNARK, reduced to checking a hash of the state changes and the proof.

and receiving proofs of state transitions from the block production system [54, 108]. Even an ADS on its own (without SNARKs) can be used to enable nodes with very little storage to participate in the consensus protocol for ordering transactions and updating state digests without participating in full state replication [158, 145, 76].

1.1.2 Communication

The way in which an ADS and SNARKs can help address communication bottlenecks depends more subtly on the design, goals, and operational setting of the blockchain. As already described, a separation of transaction ordering (consensus) from state replication and block production greatly reduces the amount of data that needs to be broadcasted within the consensus protocol, consisting only of succinct ADS digests and SNARK proofs in the extreme case. The communication reduction can greatly increase the maximum sustainable consensus throughput, so long as the block replication/production subsystems are not the bottleneck, i.e. are able to produce and replicate blocks at high enough rate. For example, this may increase throughput in a setting where the consensus is highly decentralized and the block replication/production is more centralized.

As a caveat, separating ordering from block replication/production affects the security model of the blockchain, as the majority of nodes participating in consensus are not able to drive progress of the state machine. A compromised block replication/production subsystem may indefinitely stall liveness. This *data availability* problem is discussed more below. However, an ADS and SNARKs can still increase throughput in a setting where all consensus nodes participate in both replication and ordering by reducing communication along the critical path.

While the blockchain throughput is fundamentally limited by the rate at which data can be broadcasted to all nodes participating in replication, there can be multiple broadcast channels within a distributed network that achieve different latency/throughput tradeoffs. For example, in gossip protocols where each node relays messages simultaneously to k peers, increasing the degree parameter k decreases latency but also slows throughput [159]. By broadcasting only succinct ADS digests and SNARK validity proofs on the low-latency low-throughput channel, consensus nodes can finalize commitments to transaction blocks within a short confirmation time, and will eventually receive the full block data on the high-latency high-throughput channel. This way a node's replicated state may lag several blocks behind the latest confirmed block but will still keep pace with consensus progress over time.

In some protocol designs, the throughput of consensus is far below the broadcast throughput. For example, in Nakamoto consensus[12] there is a probabilistic delay between transaction block proposals (enforced by proof-of-work) which for security is, in expectation, a factor $c > 2$ larger than the worst-case broadcast latency of a block, resulting in a throughput that is at least c times lower than the broadcast throughput [131]. This is to minimize the advantage of a node that receives a block proposal earlier than others in getting a head start on winning the next leader election. In Bitcoin $c \approx 60$, which was shown to be necessary for reaching a byzantine fault tolerance threshold of 1/2 [131]. In this case, an ADS and SNARKs can be applied to increase consensus throughput up to the broadcast throughput without changing the security properties of the consensus protocol. Since a block proposal can be verified given only an ADS digest and SNARK proof while the rest of the data is still being transmitted, the block delay can be reduced to c times the broadcast latency of this smaller digest/proof. Thus, the consensus throughput can match the broadcast throughput if the digest/proof are at least c times smaller than the block.

1.1.3 Data availability

A serious risk of separating transaction ordering from state replication, while beneficial for scalability, is that data becomes lost or unavailable to the majority of nodes participating in consensus. As a consequence, users may be unable to read the state or build valid new transactions. In the context of cryptocurrencies or DeFi protocols, this indefinite loss of liveness can incur significant financial loss. For example, a single malicious block producer might broadcast an ADS digest and SNARK proof that is accepted by consensus, but withhold, or entirely discard, the block data. The risk comes not only from intentionally malicious

[12] This is the consensus protocol used in Bitcoin and many other blockchains based on proof-of-work.

actors, but also from a "tragedy of the commons": the nodes participating in consensus do not need this data to verify transactions and collect rewards for their participation, and thus, they may not have any individual incentive to replicate. In contrast, when ordering and replication are tightly coupled within the consensus protocol, new transactions and updates to the state are only accepted once an overwhelming majority of participating nodes have received the full transaction data and state updates, mitigating this risk.

There are several ways to address this problem without losing the scalability benefits of authenticated data structures and SNARKs.

Data sampling In a method known as *data availability sampling* [6, 126, 168], all nodes participating in consensus download a small segment of the newest transaction block. This can be done is such a way that the segments received by any sufficiently large subset of uncorrupted nodes can be used to recover the entire block. For example, using erasure codes, the block can be encoded such that any k out of m total segments can be used to recover the whole block.[13] If there are $n > m$ nodes participating, the total communication complexity of this scheme is a factor m smaller than broadcasting the entire block to everyone. On the other hand, a larger m also increases the complexity of block recovery and requires higher degree erasure codes to keep the fault tolerance threshold $1 - k/m$ constant. Furthermore, data availability sampling does not address the aforementioned incentive issue.

Proofs of replication An alternative approach is to create a robust data replication layer in which many nodes are incentivized to participate in persisting data and making it available. A marketplace naturally exists: users who need the data will pay service providers for access, providing an incentive to these service providers to persist the data. But relying on this marketplace alone runs the risk of monopolization by a dominant service provider. To encourage decentralization, the blockchain protocol may reward nodes for having access

[13] If the m segments are distributed evenly over all nodes participating in consensus, then any k/m fraction of the consensus nodes collectively hold at least k of the m segments and could cooperate to recover the whole block. In unauthenticated protocols (e.g., proof-of-work) where nodes do not have identities, a sufficiently even distribution can be achieved with high probability by having each node sample a number of random segments. This number depends on an estimate of the total participation.

to data, similar to how nodes are rewarded for participating in consensus (e.g., nodes are periodically given the opportunity, at random, to earn a portion of the transaction fees by proving they have access to the latest state). This requires nodes to prove they have access to the data. While sending the entire data is a trivial and inefficient proof of access, there are protocols called *proofs of retrievability* [102] that only require communication sublinear in the size of the data, or even constant size. However, there remains the risk that all nodes share access to a single physical copy of the data, or due to economies of scale, that a single node computes proofs of retrievability on behalf of other nodes for a small fee. A *proof of replication* (PoRep) [3, 79] addresses this issue. It is a new type of proof of retrievability in which proving access to data D under a unique identifier id requires storing a unique encoding of D. In particular, computing n distinct proofs of retrievability for D requires storing n unique encodings of D, thus lowering the economies of scale of centralized production.

The concept of a PoRep was introduced as part of the *Filecoin* whitepaper in 2017 [2, 3]. *Filecoin* is a decentralized storage network that subsidizes storage fees through currency minting, and provides an eco-friendly alternative to Bitcoin. *Filecoin* was deployed as a live system in 2020 using a PoRep implementation based on the construction developed in this dissertation. Today, *Filecoin* has a storage capacity exceeding 18 exabytes.[14]

1.2 Overview of contributions

1.2.1 DARK: succinct proofs without trusted setup

Three important performance characteristics of SNARKs are proof size, proving time, and verification time. All SNARKs are required to have asymptotically small proofs and verification time, sublinear in the size of the computation predicate. And while most constructions of SNARKs have a proving time that is quasilinear in the predicate size, all practical constructions to date still have runtimes that are orders of magnitude slower than computing the predicate directly. When it comes to scaling blockchain throughput, a smaller proof size

[14]https://messari.io/report/filecoin-has-it-an-ecosystem-overview

is better for addressing communication bottlenecks, a lower verification time is better for addressing computational bottlenecks, and a lower proving time reduces the burden placed on the block production layer. Currently, there are numerous SNARK constructions that achieve different tradeoffs between proof size, proof time, and verification time, but also under different trust models as well as cryptographic assumptions.

The SNARKs with the smallest proof size rely on a *trusted setup*. These are the pairing-based SNARKs beginning with the work of Gennaro *et. al.* [89, 149, 36, 26, 96], which have constant proof size and verification time. In particular, the construction known as Groth16 [96] has a proof size as small as 200 bytes for 128-bit security and a verification time on the order of milliseconds. In a trusted setup, a trusted party runs a preprocessing algorithm for a predicate ϕ to generate a proving key pk_ϕ (for the prover) and verification key vk_ϕ (for the verifier). If secret information leaks from this setup procedure the security (soundness) of the SNARK is compromised. The setup is called *universal* if the trusted party generates one set of parameters pp for each size parameter $n \in \mathbb{N}$ that anyone can use to derive the proving and verification keys for any predicate up to size n. Recent constructions of SNARKs with universal setup based on the Kate *et al.* [105] pairing-based commitment scheme for univariate polynomials have proof sizes around 1kb for $n \approx 2^{20}$, which scale logarithmically in the size of predicate [119, 63, 86]. Their verification times are also on the order of milliseconds and similarly scale logarithmically in the predicate size. A trusted setup can also be replaced with a *multi-party computation* (MPC) by a committee of parties, such that trust in only one of the parties is sufficient. This has been done on two occasions for the ZCash blockchain, involving elaborate "ceremonies" to engender public trust in the process [166]. Similar ceremonies have been repeated for other live blockchains.

A SNARK is called *transparent* if it does not involve any trusted setup. Prior to the work included in this dissertation, the best performing transparent SNARKs had proof sizes of at least 200 kilobytes for $n \approx 2^{20}$ that scale polylogarithmically in the predicate size, i.e. as $O(\log^2 n)$ [23, 28, 65].[15] These follow the construction paradigm of the earliest

[15]The verification times of these SNARK constructions also scale as $O(\log^2 n)$, but in practice are comparable with pairing-based SNARKs due to cheaper base operations (field arithmetic compared with elliptic curve group operations).

theoretical SNARKs by Micali [122], which were called *computationally sounds proofs* (CS proofs). The prover commits to a long *probabilistically checkable proof* (PCP) using a Merkle tree and then uses a random oracle to generate a few random query positions. The prover then verifiably opens the proof at the queried positions by providing Merkle inclusion paths. The more recent constructions of transparent SNARKs generalize PCPs to *interactive oracle proofs* (IOPs) and are specifically based on the Fast Reed-Solomon IOP of Proximity (FRI) [21] protocol, which proves the proximity of vector commitments to Reed-Solomon codewords [21]. While significantly less efficient (in proof size) compared with pairing-based SNARKs, these constructions have the added benefit of only relying on the security of hash functions and are thus believed to be post-quantum secure. Pairing-based SNARKs rely on various hardness assumptions in elliptic cure groups, including the hardness of discrete log, which are not post-quantum secure.

In this dissertation, we construct the first transparent SNARKs that have both $O(\log n)$ proof size and $O(\log n)$ verification time.[16] They are also the first SNARKs based on groups of unknown order (GUOs). More precisely, the construction's security is based on the difficulty of computing the order of a random element in a GUO. When the GUO is instantiated using class groups of imaginary quadratic order with 1600-bit discriminants, estimated to provide 128-bit security [33], the resulting SNARK proof size for $n \approx 2^{20}$ is around 10 kilobytes. At the core of our construction is a new polynomial commitment scheme for polynomials over finite fields, called DARK (*Diophantine Argument of Knowledge*), which utilizes integer representations of polynomials and integer commitments in GUOs.

For the security analysis of this protocol, we develop a new variant of the famous DeMillo-Lipton-Schwartz–Zippel[17] (DLSZ) lemma [74, 171, 148] that applies to multilinear polynomials modulo a composite integer N. The original DLSZ states that for any field

[16]Technically, the proofs have size $O(\lambda \log n)$. The proof consists of $O(\log n)$ elements in a group $\mathbb{G}$ and elements in a field $\mathbb{F}_p$. It is the first transparent group-based proof system to achieve this. Each group and field element is of size $O(\lambda)$, where λ is a security parameter. In SNARKs from IOPs, the proof consists of $O(\log^2 n)$ hash digests, which similarly each have size $O(\lambda)$. However, based on current instantiations of GUOs, the group elements in DARK are larger than hash digests.

[17]See this blog post for a detailed history of the lemma: `https://rjlipton.wpcomstaging.com/2009/11/30/the-curious-history-of-the-schwartz-zippel-lemma/`

$\mathbb{F}$, non-empty finite subset $S \subseteq \mathbb{F}$, and non-zero μ-variate polynomial f over $\mathbb{F}$ of total degree d, the probability that $f(\mathbf{x}) = 0$ for $\mathbf{x}$ sampled uniformly from S^n is bounded by $\frac{d}{|S|}$. For $\mu = 1$ this simply follows from the Fundamental Theorem of Algebra, but for multivariate polynomials, the number of zeros over the whole field could be unbounded. The classical lemma applies more broadly to integral domains, but not to arbitrary commutative rings. The lemma has been extended to commutative rings (such as $\mathbb{Z}_N$) by restricting the set S to "special" subsets in which the difference of any two elements is not a zero divisor [34]. For the purpose of analyzing the security of DARK, we need to bound the probability that $f(\mathbf{x}) = 0 \bmod N$ for any multilinear polynomial f co-prime to N and $\mathbf{x}$ sampled uniformly from a μ-dimensional box $[0, m)^\mu$. The set $S = [0, m)$ is not "special", but nonetheless, we are able to show that when f is multilinear and *co-prime* with N, then the probability is bounded by $\frac{\mu}{|S|} + 2^{-\lambda}$ for sufficiently large N, specifically $\log_2 N \geq 8\mu^2 + \lambda(\log_2 \mu + 1)$. We also computationally derive a tighter lower bound on the requisite size of N for given values of μ and λ using a knapsack approximation algorithm.

These new results on transparent SNARKs and the DARK polynomial commitment scheme are covered in Chapter 2. Our new variant of the Schwartz-Zippel lemma, which is used in the security proofs of Chapter 2, is presented separately in Chapter 3.

1.2.2 Authenticated datastructures with batched proofs and constant-size parameters

A second contribution of this dissertation is a new collection of authenticated data structures for sets, vectors, and key/value maps that have smaller size authentication proofs and amortization benefits for authenticating multiple items at once. Merkle trees [121] have $O(\log n)$ size authentication proofs for a vector of n items and offer little amortization unless authenticating a block of adjacent items. Authenticated sets, or *accumulators*, with constant size proofs have been constructed from the RSA assumption in groups of unknown order (such as an RSA group[18] or class group) [18, 57, 112, 117] and also from bilinear

[18]An RSA group is the multiplicative group of integers modulo N, where $N = pq$ is generated by a trusted party that samples and discards safe primes p and q.

maps [73, 56, 127]. Constructions based on bilinear maps, called *pairing-based accumulators*, or RSA groups, called *RSA accumulators*, both require a trusted setup, but the setup parameters in RSA groups are constant size whereas the pairing-based constructions have parameters of size linear in the set size. The construction in this dissertation augments RSA accumulators with new protocols for batching and aggregating authentication proofs for both membership and non-membership queries. In batching, a constant size proof of (non)-membership for multiple items at once is created, whereas aggregation takes multiple (non)-membership proofs for individual items and produces a single aggregate proof of constant size (independent of the number of items).

An authenticated data structure for a vector is also called a *vector commitment*, and authentication proofs for items at positions of the vector are called *openings*. Prior to the work included in this dissertation, there existed vector commitments with constant size openings, based on similar techniques to accumulators with constant size membership proofs [114, 60, 113]. However, these constructions similarly suffered from large setup parameters linear in the length of the vector, even when instantiated with groups of unknown order. In particular, large setup parameters linear in the length fo the vector prohibit commitments to *sparse* vectors with length exponential in the number of non-zero entries. Sparse vectors are useful because they are equivalent to key/value maps.

Using our RSA accumulator with constant-size batch membership and non-membership proofs, we construct the first vector commitments with constant size openings and constant size parameters, which also allow for opening multiple positions of the vector (i.e., a *sub-vector*) at once with a single constant size proof. Thus, this vector commitment is also compatible with sparse vectors, and is the first authenticated key/value map with constant size authentication proofs, independent of the size of the key space and independent of the number of key/value pairs authenticated in one batch (1.5kb with a 3072-bit RSA group). Prior constructions of authenticated key/value maps from sparse Merkle trees had authentication proofs of size logarithmic in the key space (8kb for a 256-bit key space) and did not feature batching for multiple key/value pairs.

A key new tool behind this result is a succinct proof of knowledge of an integer discrete log in a group of unknown order, i.e. given $g, h \in \mathbb{G}$ it proves knowledge of $x \in \mathbb{Z}$ such that $g^x = h$. Importantly, the proof size is constant, independent of the size of x. This protocol builds on Wesolowski's recent proof-of-exponentiation [163], which proves this same statement for a specified x, but was not a proof of knowledge for hidden x. Our protocol generalizes to a succinct proof of knowledge of a pre-image $\mathbf{x} \in \mathbb{Z}^n$ for any homomorphism $\phi : \mathbb{Z}^n \to \mathbb{G}$ and any group of unknown order $\mathbb{G}$, which has proven useful for follow up work [**?**]. Finally, in addition to direct applications to blockchain scalability, we also apply our new vector commitments to reduce the proof size of SNARKs constructed from PCPs or IOPs.

These results are covered in Chapter 4.

1.2.3 Proofs of replication

A proof-of-retrievability (PoR) [102, 151, 12, 47, 75] is an interactive proof system in which a client (verifier) sends a data array D to a server (prover), retains a succinct verification key, and later challenge the server to produce a succinct[19] proof that it is still able to retrieve D intact. In a *publicly verifiable* PoR, the verification key can be deterministically derived from D, and thus anyone can play the role of the verifier. For example, the verification key might be a cryptographic commitment to D.

Suppose that D is distributed to n servers and that each server engages in a PoR for D. For example, in the setting of a blockchain that incentivizes replication of its state, each server might collect a reward for producing a PoR of the state. However, this may create an economic incentive to store only one copy of D on one server that produces all n PoRs on behalf of the other $n-1$ proxies. Is it possible to verify that each of the n servers is indeed keeping an independent copy of D? Unfortunately, this is not possible, without making strong physical assumptions about the network, such as the geographic locations of servers and the ability to infer from network timing whether the n servers are computing PoRs independently or colluding in the background [162, 79]. However, it is possible to design a special type of PoR protocol where a prover engaging in n independent PoR instances for

[19]The succinctness requirement precludes the trivial solution of sending all of D as a proof.

D requires utilizing n times as much storage as in a single instance. While this does not guarantee replication, it reduces the incentive to cheat.

A *proof of replication* (PoRep) [3, 79] is a new type of publicly verifiable PoR that achieves this property. More precisely, given parameters m and t, the prover can initialize a PoRep on any data D of length m with unique identifier id by publishing a verification key vk_{id}, deterministically computed from D and id, and will respond to challenges at regular time intervals of length t. For correctness, for each id the protocol should allow the prover to store a unique copy of D utilizing space at most $O(m)$. For security, it should be computationally infeasible to succeed in this protocol as the prover for n distinct unique identifiers without persistently utilizing space $\Omega(n \cdot m)$ throughput the challenge/response period. If the honest prover uses $H(m)$ space in each instance and the adversarial lower bound is $L(m)$ per instance then we define the protocol *space gap* as $\frac{H(m)-L(m)}{H(m)}$. For example, if the honest prover uses exactly m space and the lower bound on the adversarial's prover space per instance is $(1-\epsilon)m$, then the space gap is ϵ. As $\epsilon \to 0$ the prover's ability to save space by deviating from the honest strategy vanishes.

A PoRep is closely related to a *proof of space* (PoS) [77, 92, 130]. A PoS is also an interactive protocol between a prover and verifier in which the prover must use a minimum amount of space in order to pass verification with non-negligible probability. A PoS is *persistent* if repeated audits force the prover to utilize this space over a period of time. More precisely, there is an offline phase in which the prover obtains challenges from a verifier, generates an incompressible string σ of length m that it stores, and outputs a compact verification tag τ to the verifier. The communication between prover and verifier during this phase should be succinct, i.e. ideally constant size and at the very least sublinear in m. This is followed by an online challenge-response protocol in which the verifier uses τ to generate challenges and the prover uses σ to efficiently compute responses to the verifier's challenges. The soundness of a persistent PoS says that, assuming audits are repeated with frequency t, any adversary that stores some $\sigma^* \neq \sigma$ of length at most $(1-\epsilon)m$ will fail to pass the protocol with overwhelming probability, where ϵ is the space gap security parameter. This soundness relies on a time bound on the online prover's response in addition to the frequency of audits. This time bound is necessary as otherwise the prover could store its compact transcript from the offline phase, and simulate the setup to re-derive σ whenever it needs to pass a challenge in the online phase. Thus, if the time to generate σ is $G(m)$, then we must set $t < G(m)$. Allowing for a longer audit interval is more practical. Thus, it is desirable to achieve $G(m) \in O(m)$ and $t \in \Omega(m)$.

There is a simple generic transformation that combines any PoS and PoR into a PoRep that only requires the prover to store one additional copy of D (i.e., $n+1$ encodings of D if running n instances under distinct identifiers, where one encoding is D itself). In a PoS, the string σ is deterministically derived from the transcript between the prover and verifier in the offline phase. This transcript has a compact representation as a string id, and the space of possible transcripts is exponentially large. This forms the space of unique identifiers to be used by the PoRep construction. To encode a file D of length m under an identifier id, the prover simulates the offline phase of the PoS to derive the string σ_{id} from id and stores $\sigma_{id} \oplus D$. Both parties also run the PoR setup for D to derive the verification key vk for D. The verifier stores vk and τ_{id} for each identifier as the PoRep verification key. For each audit, the PoRep verifier runs both the PoR challenge-response using vk and the PoS challenge-response using τ_{id} for each of the n identifiers. Since the prover is storing D, it can use this to pass the single PoR challenge and also to recover each σ_{id} and pass each PoS challenge. Even if this additional copy of D were lost, the prover could still extract D from any of the encoded replicas $\sigma_{id} \oplus D$ by deterministically deriving σ_{id} from id, although this is a slow procedure.

Our main contribution is a new practical PoS that can be configured to have an arbitrarily small space gap ϵ, where the size of the offline proof is $O(\lambda \log m/\epsilon)$ and online proof is $O(\lambda/\epsilon)$ targeting security level $e^{-\lambda}$ (i.e., the probability an adversary using less than a $1-\epsilon$ fraction of the honest prover's space succeeds). We call this a *tight* PoS because ϵ can be made as small as desired, while maintaining a practical proof size. Previously, the only other construction of PoS with arbitrarily small ϵ had offline proof size $\Omega(\log m/\epsilon^2)$ and also had enormous constant factors even for large ϵ [133]. Prior PoS constructions with more practical proof sizes achieved at best a 1/2 space gap [140]. At the core of our construction is a new type of directed acyclic graph built from bipartite expander graphs and depth robust graphs (DRGs), which satisfies certain *pebbling hardness* properties.

The concrete property of an m-node DRG that our construction relies on is the existence of a constant β such that every subgraph of size $0.80m$ contains a path of length at least βm. There are explicit constructions of DRGs with this property that have degree $O(\log m)$ [132, 118, 9]. In fact, the offline proof size in our construction depends on the degree d of this DRG and is more generally $O(\lambda d/\epsilon)$. There is a heuristic construction of Alwen et. al. [8] that has constant degree and has been empirically shown to achieve the property we require. Relying on their empirical analysis, we estimate that to achieve a 5% space gap (i.e, $\epsilon = 0.05$) targeting security level 2^{-10} the offline proof size in our construction is less that a megabyte, even as the data array scales to arbitrary lengths. In contrast, we estimate the offline proof sizes of prior constructions achieving a similar space gap and security level to be on the order of gigabytes. Finally, there is a simple transformation of our PoS into a PoRep that, unlike the generic construction outlined above, does not require keeping an additional copy of D.

These results are covered in Chapter 5.

Chapter 2

Transparent SNARKs from GUOs

Since the landmark discoveries of *interactive proofs* (IPs) [95] and *probabilistically checkable proofs* (PCPs) [14, 11] in the 90s, there has been tremendous development in the area of proof systems whereby a prover establishes the correct performance of an arbitrary computation in a way that can be verified much more efficiently than performing the computation itself. Such proof systems are *succinct* if they also have a low communication cost between the prover and the verifier, *i.e.*, the transcript of the protocol is much smaller than a witness to the computation. There are also *zero knowledge* variants of these efficient proof systems, beginning with ZK-IPs [20] and ZK-PCPs [107], in which the computation may involve secret information and the prover demonstrates correct performance without leaking the secrets. As a toy example, one could prove that a chess position is winning for white without actually revealing the winning moves themselves. General purpose zero-knowledge proofs [93] can be very expensive in terms of proof size and verification time even for computations that would be easy to perform given the secret inputs (*e.g.*, by proving that one decrypted a file properly without leaking the key or the plaintext). The same techniques that are used to build efficient proof systems for expensive computations are also useful for making zero-knowledge proofs more practical.

In addition to the applications of succinct proof systems to blockchain scalability discussed in Section 1.1, there are also applications of verifiable outsourced computation [161] in the realm of cloud computing. Additionally, blockchains use efficient zero-knowledge proofs as a solution for balancing privacy and publicly-verifiable integrity: examples include anonymous transactions in ZCash [24, 1, 100], and more generally, decentralized private computation [46]. In these applications, zero-knowledge proofs are posted to the blockchain ledger as a part of transactions and nodes must verify many proofs in the span of a short period of time. Therefore, succinctness and fast verification are necessary properties for the deployment of such proof systems.

Following this pragmatic interest, there has also been a surge of research focused on obtaining proof systems with better concrete efficiency characteristics: *succinctness* (the proof size is sublinear in the original computation length T), *non-interactivity* (the proof is a single message), *prover-scalability* (proof generation time scales linearly or quasi-linearly in T), and *verifier-scalability* (verification time is sublinear in T). Some constructions achieve better efficiency by relying on a *preprocessing model* in which a one-time expensive setup procedure is performed in order to generate a compact verification key VK for a particular program, which is later used to verify proof instances efficiently. The proof systems with the smallest proof sizes and verification time require a *trusted* preprocessing. These are the pairing-based SNARKs extending from GGPR [89, 149, 36, 26, 96], which have been implemented in numerous libraries [26, 45], and even deployed in live systems such as the ZCash [1] cryptocurrency. A proof system is called *transparent* if it does not involve any trusted setup. Progress in recent years has yielded practical transparent *interactive* proof systems [23, 94, 28, 65], which are secure in the random oracle model and can be made non-interactive via the Fiat-Shamir heuristic [78, 59]. Nevertheless, their performance has remained significantly worse than SNARKs based on preprocessing, especially in terms of proof size. For computations expressed as an arithmetic circuit of 1-million gates, STARKs [23] report a proof size of 600 KB and $\mathsf{Fractal}$ [65] report 200 KB, whereas preprocessing SNARKs with trusted setup have achieved a proof size of only 200 bytes [96] independent of the circuit size.

Another thread of research has produced proof systems that remove trust from the circuit preprocessing step, and instead have a *universal* (trusted) setup: a one-time trusted setup that can be reused for *any* computation [120, 167, 87]. All of these systems build SNARKs by combining an underlying reduction of circuit satisfiability to probabilistic testing of polynomials (with degree at most linear in the circuit size) together with *polynomial commitment schemes*. In a polynomial commitment scheme (PCS), a prover commits to a μ-variate polynomial f over $\mathbb{F}$ of total degree at most d with a message that is much smaller than sending all the coefficients of f. The prover can later produce a non-interactive argument that $f(z) = y$ for arbitrary $z \in \mathbb{F}^\mu$ and $y \in \mathbb{F}$. The trusted portion of the universal SNARK is entirely confined to the PCS setup. These constructions use variants of the Kate *et al.* commitment scheme for univariate polynomials [105], which requires a trusted setup.

2.1 Summary of results

Following the observations of the recent universal SNARK constructions [87, 120, 167], SNARKs can be built from any PCS such that all the trust is confined to the setup of the PCS. The main technical contribution of our work is thus a new PCS without trusted setup (*i.e.*, a transparent PCS), which we can use to construct transparent SNARKs. The observation that transparent polynomial commitment schemes imply transparent SNARKs was also implicit in the recent works that build transparent SNARKs from multi-round classical PCPs, and specifically interactive oracle proofs of proximity (IOPPs) [21]. As a secondary contribution, we present a framework that unifies all existing approaches to constructing SNARKs from polynomial commitments using the language of *interactive oracle proofs* (IOPs) [139, 29]. Accordingly, we view a PCS as a compiler for *Polynomial IOPs*, and re-characterize the results of prior works as providing a variety of Polynomial IOPs for NP.

New polynomial commitment scheme We construct a new PCS for μ-multivariate polynomials of total degree d with optional zero-knowledge arguments of knowledge for correct evaluation that have $O(\mu \log d)$ size proofs and are verifiable in $O(\mu \log d)$ time.

The commitment scheme requires a group of unknown order: two candidate instantiations are RSA groups and class groups of an imaginary quadratic order. Using RSA groups, we can apply the scheme to obtain universal preprocessing SNARKs with *constant-size* setup parameters (though technically $O(\lambda)$, where λ is a security parameter), as opposed to the linear-size parameters from previous attempts. Using class groups, we can remove the trusted setup from trusted-setup SNARKs altogether, thereby making them *transparent.* Our polynomial commitment scheme is based on integer commitments and *Diophantine Arguments of Knowledge* [116]; accordingly, we nam this construction the *DARK* PCS.

Polynomial IOP formalism All SNARK constructions can be viewed as combining an underlying information-theoretic statistically-sound protocol with a *cryptographic compiler* that transforms the underlying protocol into a succinct argument at the cost of computational soundness. We define a *Polynomial IOP* as a refinement of algebraic linear IOPs [101, 36, 39], where in each round of interaction the prover provides the verifier with oracle access to a multivariate polynomial function of bounded degree. The verifier may then query this oracle to evaluate the polynomial on arbitrary points of its choice. The existing universal and transparent SNARK constructions provide a variety of statistically-sound Polynomial IOPs for circuit satisfiability which are then cryptographically compiled using some form of a polynomial commitment, typically using Merkle trees or pairing groups.

The linear PCPs underlying GGPR and its successors (*i.e.*, based on QAPs and R1CS) can also be transformed into Polynomial IOPs.[1] This transformation helps highlight the fundamental paradigm shift between constructions of non-transparent non-universal SNARKs that combine linear PCPs and *linear-only encodings* versus the more recent ones based on polynomial commitments: given the lack of efficient[2] *linear function* commitment schemes, the compilation of linear PCPs *necessarily* involves a trusted preprocessing step that preselects the verifier's linear PCP queries, and hides them inside a linear-only encoding. This

[1]This observation was also implicit in the paper by Ben-Sasson *et al.* introducing the system Aurora [28].

[2]Lai and Malavota [109] provide a n-dimensional *linear-map* commitment based on bilinear pairings, extending techniques in functional commitments [113], but verifying claimed evaluations of the committed function on query points takes $O(n)$.

linear-only encoding forces the prover to homomorphically output an (encoded) linear tranformation of the query, upon which the verifer performs several homomorphic checks (*e.g.*, using pairings). The shift towards Polynomial IOPs, which can be compiled more directly with efficient polynomial commitments, avoids the involvement of a trusted party to place hidden queries in the preprocessing. The only potential need for non-transparent setup is in the instantiation of polynomial commitment itself.

The precise definition of Polynomial IOPs as a central and standalone notion raises the question about its exact relation to other IOP notions. We present a univariate Polynomial IOP for extracting an indicated coefficient of a polynomial. Furthermore, we present a univariate Polynomial IOP for proving that the inner product between the coefficient vectors of two polynomials equals a given value. This proof system is of independent interest. Together with an offline pre-processing phase during which the correctness of a multivariate polynomial is ascertained, these two tools enable us to show that *any* algebraic linear IOP can be realized with a multivariate Polynomial IOP.

Polynomial IOP compiler We present a generic compilation of any public-coin Polynomial IOP into a doubly-efficient public-coin interactive argument of knowledge using an abstract polynomial commitment scheme. We prove that if the commitment scheme's evaluation protocol has witness-extended emulation, then the compiled interactive argument has this knowledge property as well. If the commitment scheme is hiding and the evaluation is honest-verifier zero knowledge (HVZK), then the compiled interactive argument is HVZK as well. Finally, public-coin interactive arguments may be cryptographically compiled into SNARKs using the Fiat-Shamir transform.

New SNARK without Trusted Setup The main practical outcome of this work is a new transparent proof system (Supersonic) for computations represented as arbitrary arithmetic circuits, obtained by cryptographically compiling the Polynomial IOPs underlying Sonic [120], PLONK [87], and Marlin[62] using the DARK polynomial commitment scheme.

Supersonic improves the proof size by an order of magnitude over STARKs without compromising on verification time. For one million gates, Supersonic's proofs are just 7.8KB and take around 75ms to verify. Using the notation $O_\lambda(\cdot)$ to hide multiplicative factors dependent on the security parameter λ, STARKs have size and verification complexity $O_\lambda(\log^2 T)$ whereas Supersonic has size and verification complexity $O_\lambda(\log T)$. (The additional multiplicative factors dependent on λ are actually better for Supersonic as well.) As a caveat, while the prover time in Supersonic is asymptotically on par with STARKs (*i.e.*, quasilinear in T), the concrete efficiency is much worse due to the use of heavy-weight "crypto operations" over 1200 bit class group elements in contrast to the light-weight FFTs and hash functions in STARKs. Furthermore, Supersonic is not quantum-secure due to its reliance on groups of unknown order, whereas STARKs are a candidate quantum-secure SNARK.

Security analysis Based on a standard *transcript forking* analysis (for special-sound multiround protocols), it is not hard to show the existence of an extractor that extracts a *rational* polynomial from a successful prover. As for intuition behind why this may be the case, while the protocol does ensure that the prover's last message is an integer (i.e., a constant degree integer polynomial), this isn't necessarily guaranteed in the prior rounds. The prover could possibly start by committing to a polynomial with rational coefficients, e.g. $f(X) = \frac{g(X)}{N} \in \mathbb{Q}[X]$ for $g(X) \in \mathbb{Z}[X]$ and $N \in \mathbb{Z}$. In each step of the protocol, the prover computes a random linear combination of two halves of the polynomial. It is possible that for some random challenge this random linear combination of two rational polynomials results in an integer polynomial.

Fortunately, the polynomial commitment is still binding to polynomials with bounded rational coefficients (see Lemma 10), i.e. both bounded numerators and denominators. A naive extraction analysis obtains extremely loose bounds on the size of the extracted rational coefficients, which would necessitate parameters that incur a super-quadratic prover and setup time. However, from the perspective of an adversary, it is challenging to pick a polynomial with a very large denominator and end up with an integer in the last round with high probability over the random challenges. Taking a closer look at the protocol, in a convincing proof the final prover message is an integer equivalent to $\tilde{f}(\alpha_1, \ldots, \alpha_\mu) = \frac{\tilde{g}(\alpha_1,\ldots,\alpha_\mu)}{N}$ where $\tilde{f}$ and $\tilde{g}$ are multi-linear polynomials with the same coefficients as f and g respectively and $\alpha_1, \ldots, \alpha_\mu$ are the verifier's random challenges. This implies that $\tilde{g}(\alpha_1, \ldots, \alpha_\mu) \equiv 0 \bmod N$. The probability of this event over random challenges can be bounded with an analysis akin to the famous Schwartz-Zippel lemma, but generalized for composite N. It turns out that for λ-bit challenges and sufficiently large $N \in 2^{\Omega(\lambda)}$, this probability decreases exponentially in λ. This in turn implies bounds on f. We present this variant of the Schwartz-Zippel lemma as an independent result in Chapter 3.

This analysis of a particular attack strategy inspires confidence in the existence of a tighter extraction analysis, but more work is yet required. When restricting the allowable extracted witnesses to rational polynomials with bounded coefficients, DARK is not *special-sound*, meaning there isn't an extractor that can compute a witness from any forking transcript tree. On the other hand, DARK has a special structure that we can still exploit. First, in DARK every message is a commitment. Given a transcript tree, if the messages below the $i+1$th level have all been opened to polynomials that have bounded norm, then it is possible to extract openings of the commitments at the ith level. The problem is that the norm grows, and the extracted opening at level i are no longer guaranteed to be small enough to continue to level $i-1$. We can call these two bounds: if all openings below level i satisfy bound A, then we can extract a polynomial at level i that satisfies bound B. However, there are two other key properties that ultimately allow us to get around this issue: (a) given openings to the last $\mu - i$ commitments in a DARK transcript to rational polynomials, if the ith commitment satisfies bound B then "rerunning the protocol" as the prover using the same round challenges starting from the ith opened commitment should either recover the same openings of the last $\mu - i$ commitments or break the commitment scheme by giving an opening of one of these commitments to a distinct message; and (b) if the ith opened message does not satisfy the bound A, then if we were to rerun the honest protocol on this message as above on uniformly random round challenges, the probability it gives a valid transcript is negligible. This probability analysis relies on our new variant of the Schwartz-Zippel lemma for multilinear polynomials modulo composite integers, presented in Chapter 3.

We generalize this to the notion of *Almost Special Sound*(ASS) protocols and replace the coefficient bounds with predicates. We prove (Theorem 6) that all protocols with this ASS structure are knowledge sound, where the knowledge error is dependent on the probability that a random completion of a transcript starting from a message that fails the first predicate results in a valid transcript. Intuitively, this captures the fact that once the adversary has a message that fails the desired extraction predicate it will fail with overwhelming probability over fresh challenges to complete the proof transcript successfully. We also show that if the commitment scheme is *computationally unique*, i.e. it is hard for a prover to produce two commitments to the same message, then the Fiat-Shamir transform of ASS protocols is secure (Theorem 7). The proof is in Section 2.8.

2.2 Additional related work

Arguments based on hidden order groups Fujisaki and Okamoto [85] proposed homomorphic integer commitment schemes based on the RSA group and interactive proofs for showing that a list of committed integers satisfy modular polynomial equations. Damgård and Fujisaki [71] were the first to suggest using class groups of an imaginary quadratic order as a candidate group of unknown order. Lipmaa drew the link between zero-knowledge proofs constructed from integer commitment schemes and Diophantine complexity [116], coining the term *Diophantine Arguments of Knowledge*. Couteau *et al.* [68] weaken the security assumptions needed for integer commitments in the RSA group and also present proofs for showing a committed integer lies within a range.

Pietrzak [134] developed an efficient proof of repeated squaring, *i.e.*, proving that $x^{2^T} = y$ with $O(\log T)$ proof size and verification time in order to build a conceptually simple verifiable delay function [37] based on the RSW time-lock puzzle [142]. Wesolowski [164] improved on this result by proposing a single-round protocol to prove correct repeated squaring in groups of unknown order with a constant size proof. In Chater 4 we observe that this protocol generalizes to arbitrary exponents (PoE) and develop a proof of knowledge of an integer exponent (PoKE), as well as a zero-knowledge variant.[3]

Transparent polynomial commitments Whaby *et al.* constructed a transparent polynomial commitment scheme [160] for multilinear polynomials by combining a matrix commitment of Bootle *et al.* [43] with the inner-product argument of Bünz *et al.* [51]. For polynomials of degree d it has commitments of size $O(\sqrt{d})$ and evaluation arguments with $O(\sqrt{d})$ communication. Zhang *et al.* [169] and Kattis *et al.* [106] recently and independently showed how to build a polynomial commitment from FRI (Fast Reed Solomon IOPP) [21, 30] The commitment is transparent, has $O(\lambda)$ size commitments and evaluation arguments with $O(\log^2 d)$ communication.

Polynomial IOP formalism Chiesa *et al.* [62] introduce an information theoretic framework called *algebraic holographic proofs (AHP).* They also show that with a polynomial commitment scheme an AHP can be compiled to a preprocessing SNARK. The AHP framework is essentially equivalent to our Polynomial IOP framework. Chiesa, Ojha, and Spooner [65] explore connections between AHPs and recursive proof composition. In the same work, the authors develop an AHP-based transparent SNARK called Fractal.

2.3 Technical overview

This technical overview provides an informal description of our key technical contribution: a polynomial commitment scheme with logarithmic evaluation proofs and verification time. The commitment scheme relies on four separate tools.

1. Integer encoding of polynomials Given a univariate polynomial $f(X) \in \mathbb{Z}_p[X]$ the prover first encodes the polynomial as an integer. Interpreting the coefficients of $f(X)$ as integers in $[0, p)$, define $\hat{f}(X)$ to be the *integer* polynomial with these coefficients. The prover computes $\hat{f}(q) \in \mathbb{Z}$ for some large integer $q \geq p$. This is an injective map from

[3]In this paper we will use additive notation so technically integer exponentiation refers to a scalar multiplied with a group element. Despite this, we will continue to use the term PoE to refer to Wesolowski's protocol.

polynomials with bounded coefficients to integers and is also decodable: the coefficients of $f(q)$ can be recovered from the base-q expansion of $\hat{f}(q)$. Note that this encoding is also additively homomorphic, assuming that q is sufficiently large. The more homomorphic operations we want to permit, the larger q needs to be. The encoding additionally permits multiplication by polynomials ($\hat{f}(q) \cdot q^k$ is equal to the encoding of $f(X) \cdot X^k$).

2. Succint integer commitments The integer $x \leftarrow \hat{f}(q) \in \mathbb{Z}$ encoding a degree d polynomial $f(X)$ lies between q^d and q^{d+1}; in other words, its size is $(d+1)\log_2 q$ bits. The prover commits to x using a *succinct* integer commitment scheme that is additively homomorphic. Specifically, we use scalar multiplication in an additive group $(\mathbb{G}, +)$ of unknown order: the commitment is the single group element $x \cdot \mathsf{G}$ for a base element $\mathsf{G} \in \mathbb{G}$ specified in the setup. (Note that if the order n of $\mathbb{G}$ is known then this is not an integer commitment; $x \cdot \mathsf{G}$ could be opened to any integer $x' \equiv x \bmod n$.)

3. Evaluation protocol The evaluation protocol is an interactive argument to convince a verifier that C is an integer commitment to $\hat{f}(q)$ such that $f(z) = y$ at a provided point $z \in \mathbb{Z}_p$. The protocol must be *evaluation binding*: it should be infeasible for the prover to succeed in arguing that $f(z) = y$ and $f(z) = y'$ for $y \neq y'$. The protocol should also be an *argument of knowledge*, which informally means that any prover who succeeds at any point x must "know" the coefficients of the committed f.

As a warmup, we first describe how a prover can efficiently convince a verifier that C is a commitment to an integer polynomial of degree at most d with bounded coefficients. Assume for now that $d = 2^k - 1$. The protocol uses a recursive divide-and-combine strategy. In each step we split $f(X)$ into two degree $d' = \lfloor \frac{d}{2} \rfloor$ polynomials $f_L(X)$ and $f_R(X)$. The left half $f_L(X)$ contains the first $d' + 1$ coefficients of $f(X)$ and the right half $f_R(X)$ the second, such that $f(X) = f_L(X) + X^{d'+1} f_R(X)$. The prover now commits to f_L and f_R by computing $\mathsf{C}_L \leftarrow \hat{f}_L(q) \cdot \mathsf{G}$ and $\mathsf{C}_R \leftarrow \hat{f}_R(q) \cdot \mathsf{G}$.

The verifier checks the consistency of these commitments by testing $\mathsf{C}_L + q^{d'+1} \cdot \mathsf{C}_R = \mathsf{C}$. The verifier then samples random $\alpha \in \mathbb{Z}_p$ and computes $\mathsf{C}' \leftarrow \mathsf{C}_L + \alpha \cdot \mathsf{C}_R$, which is an integer commitment to $\hat{f}_L(q) + \alpha \cdot \hat{f}_R(q)$. The prover and verifier recurse on the statement that C' is a commitment to a polynomial of degree at most d', thus halving the "size" of the statement. After $\log_2(d+1)$ rounds, the commitment C' exchanged between prover and verifier is a commitment to a polynomial of degree 0, *i.e.*, to a scalar $c \in \mathbb{Z}_p$. So C' is of the form $\hat{c} \cdot \mathsf{G}$ where $\hat{c}$ is some integer congruent to c modulo p. The prover sends $\hat{c}$ to the verifier directly. The verifier checks that $\mathsf{G}^{\hat{c}} = \mathsf{C}'$ and also that $\hat{c} < q$.[4]

To also show that $f(z) = y$ at a provided point z, the prover additionally sends $y_L = f_L(z) \bmod p$ and $y_R = f_R(z) \bmod p$ in each round. The verifier checks consistency with the claim, *i.e.*, that $y_L + z^{d'+1} y_R = y$, and also computes $y' \leftarrow y_L + \alpha \cdot y_R \bmod p$ to proceed to the next round. (The recursive claim is that C' commits to f' such that $f'(z) = y' \bmod p$.) In the final round of recursion, the value of the constant polynomial in z is the constant itself. So in addition to testing $\mathsf{C} = \hat{c} \cdot \mathsf{G}$ and $\hat{c} < q$, the verifier also checks that $\hat{c} \equiv y \bmod p$.

4. Outsourcing large scalar multiplications for efficiency The evaluation protocol requires communicating only 2 group elements and 2 field elements per round. However, the verifier needs to check that $\mathsf{C}_L + q^{d'+1} \cdot \mathsf{C}_R = \mathsf{C}$, and naïvely performing the scalar multiplication requires $\Omega(d \cdot \log q)$ work. To reduce this workload, we employ a recent technique for proofs of exponentiation (PoE) in groups of unknown order due to Wesolowski [164] in which the prover computes this scalar multiplication (also referred to as exponentiation when using a multiplicative group) and the verifier verifies it in essentially constant time. This outsourcing reduces the total verifier time (*i.e.*, of the entire protocol) to a quantity that is logarithmic in d.

2.4 Preliminaries

We present the preliminaries on the computational assumptions in groups of unknown orders and definitions of polynomial commitment schemes. The preliminaries on proof systems in Section 2.8 along with the security analysis of our proposed scheme.

[4]In the full scheme, the verifier actually checks that $\hat{c} < B$ for a bound $B < q$ that depends on the field size p and the polynomial's maximum degree d

2.4.1 Assumptions

The cryptographic compilers make extensive use of groups of unknown order, *i.e.*, groups for which the order cannot be computed efficiently. Concretely, we require groups for which two specific hardness assumptions hold. The binding property of the polynomial commitment and the evaluation protocol, rely on the most basic assumption in groups of unknown order. The assumption states that it is hard to compute the order of random group elements. This assumption is implied by the famous RSA Assumption [143] which states that it is hard to take *random* roots (technically scalar divisions) of *random* elements. Secondly, our proofs of exponentiation which are used to make the verifier efficient, rely on the much newer Adaptive Root Assumption [164] which is the dual of the Strong RSA Assumption and states that it is hard to take *random* roots of *arbitrary* group elements. The assumption, is also used to show that the commitment scheme is *computationally unique*, that is given a message an adversary can only output a single valid commitment to the message. Both of these assumptions hold in generic groups of unknown order [72, 41].

Assumption 1 (Random Order Assumption)**.** *The random order assumption states that an efficient adversary cannot compute a multiple of the order of a given random group element. Specifically, it holds for GGen if for any probabilistic polynomial time adversary $\mathcal{A}$:*

$$\Pr\left[a \cdot \mathsf{G} = 0 : \begin{array}{l} \mathbb{G}, N \leftarrow GGen(\lambda) \\ \mathsf{G}, \xleftarrow{\$} \mathbb{G} \\ a \in \mathbb{Z} \leftarrow \mathcal{A}(\mathbb{G}, N, \mathsf{G}) \end{array} \right] \leq \mathsf{negl}(\lambda) \quad .$$

Assumption 2 (RSA assumption, [143, 68])**.** *The RSA assumption states that an efficient adversary cannot compute a random root (co-prime with the order of the group) for a given random group element. Specifically, it holds for GGen if for any probabilistic polynomial* *time adversary $\mathcal{A}$:*

$$\Pr\left[\ell \cdot \mathsf{U} = \mathsf{G} \; : \; \begin{array}{c} \mathbb{G}, N \leftarrow GGen(\lambda) \\ \mathsf{G} \xleftarrow{\$} \mathbb{G}, \ell \xleftarrow{\$} [N] \\ \mathsf{U} \in \mathbb{G} \leftarrow \mathcal{A}(\mathbb{G}, \mathsf{G}) \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

Assumption 3 (Adaptive Root Assumption)**.** *The* Adaptive Root Assumption *holds for GGen if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $\mathsf{W} \in \mathbb{G}$ and some st. Then, a random prime ℓ in $\mathsf{Primes}(\lambda)$ is chosen and $\mathcal{A}_1(\ell, \mathsf{st})$ outputs $\mathsf{W}^{1/\ell} \in \mathbb{G}$. For all efficient $(\mathcal{A}_0, \mathcal{A}_1)$:*

$$\Pr\left[\ell \cdot \mathsf{U} = \mathsf{W} \neq 1 \; : \; \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (\mathsf{W}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(\mathbb{G}) \\ \ell \xleftarrow{\$} \mathsf{Primes}(\lambda) \\ \mathsf{U} \leftarrow \mathcal{A}_1(\ell, w, \mathsf{st}) \end{array}\right] \leq \mathsf{negl}(\lambda) \, .$$

Lemma 1. *The RSA Assumption for GGen implies the Random Order Assumption*

Proof. Given an efficient adversary $\mathcal{A}_{\mathsf{Order}}$ for the random order assumption that succeeds with non-negligible probability ϵ we will construct an efficient adversary $\mathcal{A}_{\mathsf{RSA}}$ for the RSA assumption. On input $\mathbb{G}, \mathsf{G}, \ell$ to $\mathcal{A}_{\mathsf{RSA}}$ we will forward $\mathbb{G}, \mathsf{G}$ to $\mathcal{A}_{\mathsf{Order}}$. $\mathcal{A}_{\mathsf{Order}}$ outputs a such that $a \cdot \mathsf{G} = 0$ with non-negligible probability ϵ. $\mathcal{A}_{\mathsf{RSA}}$ computes $a' \leftarrow \frac{a}{\gcd(a, \ell^k)}$ for $k = \lceil \log_\ell(a) \rceil$. The probability that ℓ is not co-prime to the order of $\mathbb{G}$ is bounded by $\frac{\log_2 |\mathbb{G}|}{|\mathsf{Primes}(\lambda)|}$ which is negligible in λ. Otherwise ℓ is co-prime with the order of G and a is a multiple of the order of G we have that a' is still a multiple of the order of G. Now $\mathcal{A}_{\mathsf{RSA}}$ computes $w \leftarrow \ell^{-1} \bmod a'$ and outputs $\mathsf{U} \leftarrow w \cdot \mathsf{G}$. Now we have $\ell \cdot \mathsf{U} = \mathsf{G}$ so $\mathcal{A}_{\mathsf{RSA}}$ succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. □

Lemma 2. *The Adaptive Root Assumption for GGen implies the Random Order Assumption*

Proof. Given an efficient adversary $\mathcal{A}_{\mathsf{Order}}$ for the random order assumption we will construct an efficient adversary $\mathcal{A}_{\mathsf{AR}} = (\mathcal{A}_0, \mathcal{A}_1)$ for the Adaptive Root assumption. On input $\mathbb{G}$ to

$\mathcal{A}_0$ we will sample a random group element G from $\mathbb{G}$ and forward it to $\mathcal{A}_{\mathsf{Order}}$. $\mathcal{A}_{\mathsf{Order}}$ outputs a such that $a \cdot \mathsf{G} = 0$ with non-negligible probability ϵ. And we set the output of $\mathcal{A}_0$ to be (G, a). The adaptive root game then samples a random prime ℓ. $\mathcal{A}_1$ on input (a, g, ℓ) computes $a' \leftarrow \frac{a}{\gcd(a,\ell^k)}$ for $k = \lceil \log_\ell(a) \rceil$. Note that since ℓ is co-prime to the order of $\mathbb{G}$ and thus also the order of G and a is a multiple of the order of G we have that a' is still a multiple of the order of G. If we don't abort then we compute $\ell^{-1} \bmod a$ and $\mathsf{U} = \ell^{-1} \cdot \mathsf{G}$. Finally A_1 outputs U, which by construction is such that $\ell \cdot \mathsf{U} = \mathsf{G}$. $\square$

2.4.2 Group of unknown order (GUO)

We consider two candidate groups of unknown order. Both have their own upsides and downsides.

RSA Group. In the multiplicative group $\mathbb{Z}_n^*$ of integers modulo a product $n = p \cdot q$ of large primes p and q, computing the order of the group is as hard as factoring n. The Adaptive Root Assumption does not hold for $\mathbb{Z}_n^*$ because $-1 \in \mathbb{Z}_n^*$ can be easily computed and has order two. This can be resolved though by working instead in the quotient group $\mathbb{Z}_n^*/\{x \mid x^2 = 1\} \cong \mathrm{QR}_n$. The downside of using an RSA group, or more precisely, the group of quadratic residues modulo an RSA modulus, is that this modulus cannot be generated in a publicly verifiable way without exposing the order, and thus requires a trusted setup.

Class Group. The class group of an imaginary quadratic order is defined as the quotient group of fractional ideals by principal ideals of an order of a number field $\mathbb{Q}(\sqrt{\Delta})$, with ideal multiplication. A class group $\mathcal{C}\ell(\Delta)$ is fully defined by its discriminant Δ, which needs to satisfy only public constraints such as $\Delta \equiv 1 \bmod 4$ and $-\Delta$ must be prime. As a result, Δ can be generated from public coins, thus obviating the need for a trusted setup. A group element can be represented by two integers strictly smaller (in absolute value) than $-\Delta$, which in turn is on the same order of magnitude as RSA group elements for a similar security level. We refer the reader to Buchmann and Hamdy's survey [48] and Straka's accessible blog post [155] for more details.

Working in $\mathcal{C}\ell(\Delta)$ does present an important difficulty: there is an efficient algorithm due to Gauss to compute square roots of arbitrary elements [44], and by repetition, arbitrary power of two roots. Despite this, the random order and the adaptive root assumption still hold in class groups. Computing power of two roots does not directly enable one to compute the order of a random element or compute random (large prime) roots of a chosen element. The new security proof only relies on the random order assumption for extraction and the adaptive root assumption for the PoEs. It, therefore, holds even for adversaries that can compute square (or other small) roots of elements.

2.4.3 Commitment Schemes

In defining the syntax of the various protocols, we use the following convention with respect to public values (known to both the prover and the verifier) and secret ones (known only to the prover). In any list of arguments or returned tuple $(a, b, c; d, e)$ those variables listed before the semicolon are public, and those variables listed after it are secret. When there is no secret information, the semicolon is omitted.

Definition 1 (Commitment scheme)**.** *A commitment scheme* Γ *is a tuple* $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *of PPT algorithms where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *generates public parameters* pp*;*
- $\mathsf{Commit}(\mathsf{pp}; x) \to (C; r)$ *takes a secret message* x *and outputs a public commitment* C *and (optionally) a secret opening hint* r *(which might or might not be the randomness used in the computation).*
- $\mathsf{Open}(\mathsf{pp}, C, x, r) \to b \in \{0, 1\}$ *verifies the opening of commitment* C *to the message* x *provided with the opening hint* r*.*

A commitment scheme Γ *is* ***binding*** *if for all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\left[b_0 = b_1 \neq 0 \wedge x_0 \neq x_1 \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C, x_0, x_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, C, x_0, r_0) \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, x_1, r_1) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

We now extend the syntax to polynomial commitment schemes. The following definition generalizes that of Kate *et. al.* [105] to allow interactive evaluation proofs. It also stipulates that the polynomial's degree be an argument to the protocol, contrary to Kate *et al.* where the degree is known and fixed.

Definition 2. *(Polynomial commitment) A polynomial commitment scheme is a tuple of protocols* $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *where* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is a binding commitment scheme for a message space* $R[X]$ *of polynomials over some ring* R*, and*

- $\mathsf{Eval}(\mathsf{pp}, C, z, y, d, \mu; f) \to b \in \{0,1\}$ *is an interactive public-coin protocol between a PPT prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$*. Both* $\mathcal{P}$ *and* $\mathcal{V}$ *have as input a commitment* C*, points* $z, y \in R$*, and a degree* d*. The prover additionally knows the opening of* C *to a secret polynomial* $f(X) \in R[X]$ *with* $\deg(f(X)) \leq d$*. The protocol convinces the verifier that* $f(z) = y$*. In a multivariate extension of polynomial commitments, the input* $\mu > 1$ *indicates the number of variables in the committed polynomial and* $z \in R^{\mu}$*.*

A polynomial commitment scheme is ***correct*** *if an honest committer can successfully convince the verifier of any evaluation. Specifically, if the prover is honest, then for all polynomials* $f(X) \in R[X]$ *and all points* $z \in R$*,*

$$\Pr\left[b = 1 \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda}) \\ (C; r) \leftarrow \mathsf{Commit}(\mathsf{pp}, f(X)) \\ y \leftarrow f(z) \\ d \leftarrow \deg(f(X)) \\ b \leftarrow \mathsf{Eval}(\mathsf{pp}, c, z, y, d; f(X), r) \end{array} \right] = 1 \ .$$

A polynomial commitment scheme is ***evaluation binding*** *if no efficient adversary can convince the verifier that the committed polynomial* $f(X)$ *evaluates to different values* $y_0 \neq y_1 \in R$ *in the same point* $z \in R$*. However, our applications require a stronger property called* knowledge soundness.

Knowledge soundness Any successful prover in the Eval protocol must *know* a polynomial $f(X)$ such that $f(z) = y$ and C is a commitment to $f(X)$. More formally, since Eval

is a public-coin interactive argument we define this knowledge property as a special case of witness-extended emulation (Definition 7).

Define the following NP relation given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$:

$$\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) = \left\{ \langle (C, z, y, d), (f(X), r) \rangle : \begin{array}{l} f \in R[X] \text{ and } \deg(f(X)) \leq d \text{ and } f(z) = y \\ \text{and } \mathsf{Open}(\mathsf{pp}, C, f(X), r) = 1 \end{array} \right\}$$

The correctness definition above implies that if $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ is *correct* then Eval is a correct interactive argument for $\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp})$, with overwhelming probability over the randomness of Setup. We say that Γ has **witness-extended emulation** if Eval has witness-extended emulation as an interactive argument for $\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp})$.

It is easy to see that witness-extended emulation implies evaluation binding when the $\mathsf{Setup}, \mathsf{Commit}$, and Open part of Γ form a binding commitment scheme. If the adversary succeeds in Eval on both (C, z, y_0, d_0) and (C, z, y_1, d_1) for $y_0 \neq y_1$ or $d_0 \neq d_1$ then the emulator obtains two distinct witnesses $f(X) \neq f'(X)$ such that C is a valid commitment to both. This would contradict the binding property of the commitment scheme.

2.4.4 Proofs of Exponentiation

Wesolowski [164] introduced a simple yet powerful proof of correct exponentiation ("PoE") in groups of unknown order. A prover can efficiently convince a verifier that a large scalar multiplication in such a group was done correctly. For instance, the prover wishes to convince the verifier that $\mathsf{W} = \mathsf{U}^x$ for known group elements $\mathsf{U}, \mathsf{W} \in \mathbb{G}$ and exponent $x \in \mathbb{Z}$, and the verifier wants to verify this with much less work than performing the scalar multiplication. To do this, the verifier samples a large enough prime ℓ at random and the prover provides him with $\mathsf{Q} \leftarrow \mathsf{U}^q$ where $q = \lfloor \frac{x}{\ell} \rfloor$. The verifier then simply computes the remainder $r \leftarrow (x \mod \ell)$ and checks that $\mathsf{Q}^\ell \mathsf{U}^r = \mathsf{W}$. The protocol is an argument for the relation $\mathcal{R}_{\mathsf{PoE}} = \{\langle (\mathsf{U}, \mathsf{W}, x), \varnothing \rangle \ : \ \mathsf{U}^x = \mathsf{W}\}$. The proof verification uses just $O(\lambda)$ group operations. When x is $x = q^d$ the verifier can compute $r \leftarrow x \bmod \ell$ using just $\log(d)$ ℓ-bit multiplications.

```
PoE(U, W, $x$) :
    1. $\mathcal{V}$ samples $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ and sends $\ell$ to $\mathcal{P}$
    2. $\mathcal{P}$ computes quotient $q$ and remainder $r$ such that $x = q\ell + r$ and
       $r \in \{0, \ldots, \ell - 1\}$
    3. $\mathcal{P}$ computes $\mathsf{Q} \leftarrow q \cdot \mathsf{U}$ and sends it to $\mathcal{V}$
    4. $\mathcal{V}$ computes $r \leftarrow (x \mod \ell)$ and checks that $\ell \cdot \mathsf{Q} + r \cdot \mathsf{U} = \mathsf{W}$
    5. if check passes then return 1 else return 0
```

Lemma 3 (PoE soundness [164]). *PoE is an argument system for relation $\mathcal{R}_{\mathsf{PoE}}$ with negligible soundness error, assuming the Adaptive Root Assumption (Assumption 3) holds for GGen.*

Lemma 4 (PoE random oracle soundness [164]). *The Fiat-Shamir transform of PoE, replacing the verifier message ℓ with $\ell \leftarrow \mathsf{H}(u, w, x)$ and H is an argument system for relation $\mathcal{R}_{\mathsf{PoE}}$ with negligible soundness error, assuming that H is modeled as a random oracle and that the Adaptive Root Assumption (Assumption 3) holds for GGen.*

2.5 Polynomial Commitments from Groups of Unknown Order

2.5.1 Information-Theoretic Abstraction

Before we present our concrete polynomial commitment scheme based on groups of unknown order, we present the underlying information theoretic protocol that abstracts the concrete cryptographic instantiations. The purpose of this abstraction is two-fold: first, it provides an intuitive stepping stone from which presenting and studying the concrete cryptographic protocol is easier; and second, it opens the door to alternative cryptographic instantiations that provide the same interface but based on alternative hardness assumptions.

Let $[\![*]\!] : \mathbb{Z}_p[X] \to \mathbb{S}$ be a homomorphic commitment function that sends polynomials over a prime field to elements of some set $\mathbb{S}$. Moreover, let $\mathbb{S}$ be equipped with operations $* + * : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ and $* \cdot * : \mathbb{Z}_p[X] \times \mathbb{S} \to \mathbb{S}$ that accommodate two homomorphisms for $[\![*]\!]$:

- a *linear homomorphism*: $a \cdot [\![f(X)]\!] + b \cdot [\![g(X)]\!] = [\![af(X) + bg(X)]\!]$
- a *monomial homomorphism*: $X^d \cdot [\![f(X)]\!] = [\![X^d f(X)]\!]$.

For now, assume both prover and verifier have oracle access to the function $[\![*]\!]$ and to the operations $* \cdot *$ and $* + *$. (Later on, we will instantiate this commitment function using groups of unknown order and an encoding of polynomials as integers.)

The core idea of the evaluation protocol is to reduce the statement that is being proved from one about a polynomial $f(X)$ of degree d and its evaluation $y = f(z)$, to one about a polynomial $f'(X)$ of degree $d' = \lfloor \frac{d}{2} \rfloor$ and its evaluation $y' = f'(z)$. For simplicity, assume that $d+1$ is a power of 2. The prover splits $f(X)$ into $f_L(X)$ and $f_R(X)$ such that $f(X) = f_L(X) + X^{d'+1} f_R(X)$ and such that both halves have degree at most d'. The prover obtains a random challenge $\alpha \in \mathbb{Z}_p$ from the verifier and proceeds to prove that $f'(X) = f_L(X) + \alpha \cdot f_R(X)$ has degree d' and that $f'(z) = y' = y_L + \alpha y_R$ with $y_L = f_L(z)$ and $y_R = f_R(z)$.

The proof repeats this reduction by using $f'(X), z, y'$ and d' as the input to the next recursion step. In the final step, $f(X) = f$ is a constant and the verifier checks that $f = y$.

The commitment function binds the prover to one particular polynomial for every commitment held by the verifier. In particular, at the start of every recursion step, the verifier is in possession of a commitment $[\![f(X)]\!]$ to $f(X)$. The prover provides commitments $[\![f_L(X)]\!]$ and $[\![f_R(X)]\!]$, and the verifier checks their soundness homomorphically by testing $[\![f(X)]\!] = [\![f_L(X)]\!] + X^{d'+1} \cdot [\![f_R(X)]\!]$. From these commitments, the verifier can also compute the commitment to $f'(X)$ homomorphically, via $[\![f'(X)]\!] = [\![f_L(X)]\!] + \alpha \cdot [\![f_R(X)]\!]$. In the last step, the verifier checks that the constant polynomial f matches the commitment by computing $[\![f]\!]$ directly.

2.5.2 Integer Polynomial Encoding

We propose using integer commitments in a group of unknown order as a concrete instantiation of the homomorphic commitment scheme required for the abstract protocol presented in Section 2.5.1. At the heart of our protocol is thus an encoding of integer polynomials with bounded coefficients as integers, which also has homomorphic properties. Any commitment scheme which is homomorphic over integer polynomials is automatically homomorphic over $\mathbb{Z}_p[X]$ polynomials as well (by reducing integer polynomials modulo p). Polynomials over $\mathbb{Z}_p[X]$ can be lifted to integer polynomials in a canonical way by choosing representatives in $[0, p)$. Therefore, from here on we will focus on building a homomorphic integer encoding of integer polynomials, and how to combine this with a homomorphic integer commitment scheme.

Strawman encoding In order to encode integer polynomials over an odd prime field $\mathbb{F}_p$, we first lift them to the ring of polynomials over the integers by choosing representatives in $[0, p)$. In the technical overview (Section 2.3) we noted that a polynomial $f \in \mathbb{Z}[X]$ with positive coefficients bounded by q can be encoded as the integer $f(q)$. The coefficients of f can be recovered via the base q decomposition of $f(q)$. This encoding is an injective mapping from polynomials in $\mathbb{Z}[X]$ of degree at most d with positive coefficients less than q to the set $[0, q^{d+1})$. The encoding is also *partially* homomorphic. If f is encoded as $f(q)$ and g is encoded as $g(q)$ where coefficients of both g, f are less than $q/2$, then the base-q decomposition of $f(q) + g(q)$ gives back the polynomial $f + g$. By choosing a sufficiently large $q \gg p$ it is possible to perform several levels of homomorphic operations on encodings.

What goes wrong? Unfortunately, this simple encoding scheme does not quite work yet for the protocol outlined in Section 2.3. The homomorphic consistency checks ensure that if $[\![f_L(X)]\!]$ is a homomorphic integer commitment to the encoding of $f_L \in \mathbb{Z}[X]$, $[\![f_R(X)]\!]$ is a homomorphic integer commitment to the encoding of $f_R \in \mathbb{Z}[X]$, and both f_L, f_R are polynomials with $q/2$-bounded integer coefficients, then $[\![f(X)]\!]$ is an integer commitment to the encoding of $f_L + X^{d'} f_R$. (Moreover, if $f_L(z) = y_L \bmod p$ and $f_R(z) = y_R \bmod p$ then $f(z) = y_L + z^{d'} y_R \bmod p$).

However, the validity of $[\![f_L(X)]\!]$ and $[\![f_R(X)]\!]$ are never checked directly. The verifier only sees the opening of the commitment at the bottom level of recursion. If the intermediate encodings use integer polynomials with coefficients larger than $q/2$, or even rational coefficients the homomorphism is not necessarially preserved. Furthermore, even if $[\![f(X)]\!]$

is a commitment to $f^*(q)$ with positive q-bounded coefficients, an adversarial prover could find an integer polynomial g^* that does not have positive q-bounded coefficients such that $g^*(q) = f^*(q)$ and $g^* \not\equiv f^* \bmod p$ (*i.e*, g^* with coefficients greater than q or negative coefficients). The prover could then commit to $g^*_L(q)$ and $g^*_R(q)$, and recurse on $g^*_L(q) + \alpha g^*_R(q)$ instead of $f^*_L(q) + \alpha f^*_R(q)$. This would be non-binding. (For example $f^*(X) = q - 1$ and $g^*(X) = X - 1$, or $f^*(X) = q + 1$ and $g^*(X) = X + 1$).

Inferring coefficient bounds So what can the verifier infer from the opened commitment $\llbracket f' \rrbracket$ at the bottom level of recursion? The opened commitment is an integer $f' = f_L + \alpha f_R$. From f', the verifier can infer a bounds coefficients of the polynomial $f(X) = f_L + X f_R$, given that f_L and f_R were already committed in the second to last round. The bound holds with overwhelming probability over the randomness of $\alpha \in [0, 2^\lambda)$. This is reasoned as follows: if $f'_0 \leftarrow f_L + \alpha_0 f_R$ and $f'_1 \leftarrow f_L + \alpha_1 f_R$ but f_L and f_R are not bounded rational polynomials, then there is a negligibly small probability that f' would have passed the bound check.

What about negative coefficients? As shown above, the verifier can infer a bound on the absolute values of f_L and f_R, but still cannot infer that f_L and f_R are both *positive* integers. Moreover, if $f_R > 0$ and $f_L < 0$, then it is still possible that $f_L + q f_R > 0$, and thus that there is a distinct $g \neq f$ with q-bounded positive coefficients such that $g(q) = f(q)$. For example, say $f_R = q/2$ and $f_L = -1$ then $f_L + q f_R = q^2/2 - 1$, and $f_L + \alpha f_R = q/2 - \alpha > 0$ for every $\alpha \in [0, 2^\lambda)$. Yet, also $q^2/2 - 1 = g(q)$ for $g(X) = (q/2 - 1)X + q - 1$.

It turns out that we also can't ensure that f_L and f_R but only that they are bounded rational polynomials. We show that the same encoding works even for rational polynomials in Section 2.8.2.

Ensuring injectivity How can we ensure the encoding scheme is injective over polynomials with either positive/negative coefficients bounded in absolute value? Fortunately, it is a fact that if $|f_L| < q/2$ and $|f_R| < q/2$ then at least one coefficient of g must be larger than $q/2$. In other words, if the prover had committed instead to f^*_L and f^*_R such that

$g(X) = f_L^* + X f_R^*$ then the verifier could reject the opening of $\hat{f}_L^* + \alpha \hat{f}_R^*$ with overwhelming probability based on its size.

More generally, for every integer z in the range $B = (-\frac{q^{d+1}}{2}, \frac{q^{d+1}}{2})$ there is a unique degree (at most) d integer polynomial $h(X)$ with coefficients whose absolute values are bounded by $q/2$ such that $h(q) = z$. *We prove this elementary fact below and show how the coefficients of h can be recovered efficiently from z (Fact 1).* If the prover is committed to $h(q)$ at level i of the protocol, there is a unique pair of integers polynomial h_L and h_R with coefficients of absolute value bounded by $q/2$ such that $h_L(q) + q^{\frac{d+1}{2}} h_R(q) = h(q)$, and if the prover recurses on any other h_L^* and h_R^* with larger coefficients then the verifier's bound check at the bottom level of recursion will fail with overwhelming probability.

Final encoding scheme Let $\mathbb{Z}(b) := \{x \in \mathbb{Z} : |x| \leq b\}$ denote the set of integers with absolute value less than or equal to b. Define $\mathbb{Z}(b)[X] := \{f \in \mathbb{Z}[X] : ||f||_\infty \leq b\}$, the set of integer polynomials with coefficients from $\mathbb{Z}(b)$. (For a polynomial $g \in \mathbb{Z}[X]$ the norm $||g||_\infty$ is the maximum over the absolute values of all individual coefficients of g.)

- **Encoding.** For any integer q, the function $\mathsf{Enc} : \mathbb{Z}(b)[X] \to \mathbb{Z}$ maps $h(X) \mapsto h(q)$. A polynomial $f(X) \in \mathbb{Z}_p[X]$ is first mapped to $\mathbb{Z}(p-1)[X]$ by replacing each coefficient of f with its unique integer representative from $[0, p)$ of the same equivalence class modulo p.

- **Decoding.** Decoding works as follows. Define the partial sum $S_k := \sum_{i=0}^{k} f_i q^i$ with $S_{-1} := 0$. Assuming $|f_i| < q/2$ for all i, observe that for any partial sum S_k we have $|S_k| < \frac{q^{k+1}}{2}$. Therefore, when $S_k < 0$ then $S_k \bmod q^{k+1} > q^{k+1}/2$ and when $S_k \geq 0$ then $S_k \bmod q^{k+1} < q^{k+1}/2$. This leads to a decoding strategy for recovering S_k from $y \in \mathbb{Z}$. The decode algorithm sets S_k to $y \bmod q^{k+1}$ if this value is less than $q^{k+1}/2$ and to $q^{k+1} - (y \bmod q^{k+1})$ otherwise. Two consecutive partial sums yield a coefficient of $f(X)$: $f_k = \frac{S_k - S_{k-1}}{q^k} \in \mathbb{Z}(b)$. These operations give rise to the following algorithm.

```
$\mathsf{Dec}(y \in \mathbb{Z})$ :
  1. for each $k$ in $[0, \lfloor \log_q(|y|) \rfloor]$ do:
  2.     $S_{k-1} \leftarrow (y \bmod q^k)$
  3.     if $S_{k-1} > q^k/2$ then $S_{k-1} \leftarrow q^k - S_{k-1}$ end if
  4.     $S_k \leftarrow (y \bmod q^{k+1})$
  5.     if $S_k > q^{k+1}/2$ then $S_k \leftarrow q^{k+1} - S_k$ end if
  6.     $f_k \leftarrow (S_k - S_{k-1})/q^k$
  7. return $f(X) = \sum_{k=0}^{\lfloor \log_q(|y|) \rfloor} f_k X^k$
```

Fact 1. *Let q be an odd integer. For any z in the range $B = (-\frac{q^{d+1}}{2}, \frac{q^{d+1}}{2})$ there is a unique degree (at most) d integer polynomial $h(X)$ in $\mathbb{Z}(\frac{q-1}{2})[X]$ such that $h(q) = z$.*

Proof. Given any degree (at most) d integer polynomial $f \in \mathbb{Z}(\frac{q-1}{2})$, by construction we see that $\mathsf{Dec}(\mathsf{Enc}(f)) = f$. Therefore, Enc is an injective map from degree (at most) d polynomials in $\mathbb{Z}(\frac{q-1}{2})[X]$ to B. Furthermore, the cardinality of both the domain and range of this map is q^{d+1}. This shows that the map is surjective. In conclusion, the map is bijective. □

2.5.3 Concrete Polynomial Commitment Scheme

We now instantiate the abstract homomorphic commitment function $[\![*]\!]$. To this end we sample a group of unknown order $\mathbb{G}$, and sample a random element G from this group. Lift the field polynomial $f(X) \in \mathbb{Z}_p[X]$ to an integer polynomial with bounded coefficients, *i.e.*, $\hat{f}(X) \in \mathbb{Z}(p-1)[X]$ such that $\hat{f}(X) \bmod p = f(x)$. We encode $\hat{f}(X)$ as an integer by evaluating it at a "large enough" integer q. Finally, we use scalar multiplication in $\mathbb{G}$ to commit to the integer. Therefore, $[\![f(X)]\!]$, corresponds to $\hat{f}(q) \cdot \mathsf{G}$. This commitment function inherits the homomorphic properties of the integer encoding for a limited number of additions and multiplications-by-constant. The monomial homomorphism for X^d is achieved by raising the group element to the power q^d. To maintain consistency between the prover's witness polynomials and the verifier's commitments, the prover operates on polynomials with integer coefficients $\hat{f}(X), \hat{g}(X)$, *etc.*, without ever reducing them modulo

p.

The $\mathsf{Setup}, \mathsf{Commit}$ and Open functionalities are presented formally below. Note that the scheme is parameterized by p and q.

- $\mathsf{Setup}(1^\lambda)$: Sample $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$ and $\mathsf{G} \overset{\$}{\leftarrow} \mathbb{G}$. Return $\mathsf{pp} = (\lambda, \mathbb{G}, \mathsf{G}, q)$.
- $\mathsf{Commit}(\mathsf{pp}; f(X) \in \mathbb{Z}_p[X])$: Compute $\mathsf{C} \leftarrow \hat{f}(q) \cdot \mathsf{G}$ and return $(\mathsf{C}; f(X), \hat{f}(X))$.
- $\mathsf{Open}(\mathsf{pp}, \mathsf{C}, f(X), \hat{f}(X))$: Check that $\hat{f}(X) \in \mathbb{Z}(q/2)[X]$ and $\hat{f}(q) \cdot \mathsf{G} = \mathsf{C}$ and $f(X) = \hat{f}(X) \bmod p$.

Evaluation protocol Using the cryptographic compilation of the information theoretic protocol we get an Eval protocol with logarithmic communication. In every round, however, the verifier needs to check consistency between $[\![f_L(X)]\!], [\![f_R(X)]\!]$ and $[\![f(X)]\!]$. This is done by checking that $\mathsf{C}_L + q^{d'+1} \cdot \mathsf{C}_R = \mathsf{C}$. This naive check is highly inefficient as the exponent $q^{d'+1}$ has $O(d)$ bits. To resolve this inefficiency, we utilize a proof of exponentiation (PoE) [134, 164] to outsource the computation to the prover. The PoE protocol is an argument that a large scalar multiplication in a group of unknown order was performed correctly. Wesolowski's PoE [164] is public coin, has constant communication and verification time, and is thus particularly well-suited here.

We now specify subtleties that were previously glossed over. Instead of presenting a protocol for univariate degree d polynomials we present one for μ-linear polynomials. This is more general as for any univariate polynomial $f(X)$ there exists a $\mu = \lceil \log_2(d+1) \rceil$-linear polynomial $\hat{f}$ with the same coefficients such that $\hat{f}(X, X^2, \ldots, X^{2^{(\mu-1)}}) = f(X)$. By the same argument, we can use a multi-linear polynomial commitment and evaluation scheme for arbitrary multivariate polynomials where the degree in each variable is a power of 2. For non-power-of-2 multivariate polynomials, it is possible to round up to the next power of 2 and prove correctness using a PoE proof.

The coefficients of $f(X_1, \ldots, x_\mu)$ grow by a factor of 2^λ in every recursion step, but eventually, the transmitted constant f has to be tested against some bound because if it is *too large* it should be rejected. However, the function interface provides no option to specify the allowable size of coefficients. We therefore define and use a subroutine EvalB, which takes an additional argument b and which proves, in addition to what Eval proves, that all coefficients f_i of $f(X_1, \ldots, X_\mu)$ satisfy $|f_i| \leq b$. Importantly, b grows by a factor for 2^λ, the challenge space, in every recursion step. This subroutine is also useful if commitments were homomorphically combined prior to the execution of EvalB. The growth of these coefficients determines a lower bound on q: q needs to be *significantly* larger than b for security. Exactly which factor constitutes "significantly" is determined by the knowledge-soundness proof.

In the final round we check that the constant f satisfies $|f| \leq b$ and the protocol's correctness is guaranteed if $b = (p-1) \cdot 2^{\lambda\mu}$, where $\mu = \log(d+1)$ are the number of rounds. However, q needs to be even larger than this value in order for extraction to work (and hence, for the proof of witness-extended emulation to go through). The precise value of q depends on the number of rounds μ, and is defined in Theorem 1 and is $2^{O(\mu\lambda)}$.

We now present the full, formal Eval protocol below.

```
$\mathsf{Eval}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, \vec{z} \in \mathbb{Z}_p^\mu, y \in \mathbb{Z}_p; f(X_1, \ldots, X_\mu) \in \mathbb{Z}_p[X_1, \ldots, X_\mu],):$ // $f$ is $\mu$-linear
  1. $\mathcal{P}$ computes $\hat{f}(X_1, \ldots, X_\mu) \in \mathbb{Z}(p)[X_1, \ldots, X_\mu]$ such that $\hat{f} \bmod p \equiv f$
  2. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalB}(\mathsf{pp}, \mathsf{C}, \vec{z}, y, \mu, p-1; f, \hat{f})$

$\mathsf{EvalB}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, \vec{z} \in \mathbb{Z}_p^\mu, y \in \mathbb{Z}_p, \mu \in \mathbb{N}, b \in \mathbb{Z}; f \in \mathbb{Z}_p[X_1, \ldots, X_\mu], \hat{f} \in \mathbb{Z}(b)[X_1, \ldots, X_\mu])$
  1. if $\mu = 0$:
  2.     $\mathcal{P}$ sends $\hat{f} \in \mathbb{Z}$ to the verifier. // $\hat{f}$ is a constant
  3.     $\mathcal{V}$ checks that $|\hat{f}| \le b$
  4.     $\mathcal{V}$ checks that $\hat{f} \equiv y \bmod p$
  5.     $\mathcal{V}$ checks that $\hat{f} \cdot \mathsf{G} = \mathsf{C}$
  6.     $\mathcal{V}$ outputs 1 if all checks pass, 0 otherwise.
  7. else :
  8.     $\mathcal{P}$ and $\mathcal{V}$ compute $\mu' \leftarrow \mu - 1$
  9.     $\mathcal{P}$ computes $\hat{f}_L(X_1, \ldots, X_\mu')$ and $\hat{f}_R(X_1, \ldots, X_\mu')$ such that $\hat{f} = \hat{f}_L + X_\mu \hat{f}_R$
  10.    $\mathcal{P}$ computes $f_L, f_R$ analogously for $f$
  11.    $\mathcal{P}$ computes $y_L \leftarrow f_L(z_1, \ldots, z_\mu') \bmod p$ and $y_R \leftarrow f_R(z_1, \ldots, z_\mu') \bmod p$
  12.    $\mathcal{P}$ computes $\mathsf{C}_L \leftarrow \hat{f}_L(q, q^2, \ldots, q^{(2^{\mu'-1})}) \cdot \mathsf{G}$ and $\mathsf{C}_R \leftarrow \hat{f}_R(q, q^2, \ldots, q^{(2^{\mu'-1})}) \cdot \mathsf{G}$
  13.    $\mathcal{P}$ sends $y_L, y_R, \mathsf{C}_L, \mathsf{C}_R$ to $\mathcal{V}$. // See Section 2.5.5 for an optimization
  14.    $\mathcal{V}$ checks that $y = y_L + z_\mu \cdot y_R \bmod p$, outputs 0 if check fails.
  15.    $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(\mathsf{C}_R, \mathsf{C} - \mathsf{C}_L, q^{(2^{\mu'})})$// Showing that $\mathsf{C}_L + q^{(2^{\mu'})} \cdot \mathsf{C}_R = \mathsf{C}$
  16.    $\mathcal{V}$ samples $\alpha \xleftarrow{\$} [0, 2^\lambda)$ and sends it to $\mathcal{P}$
  17.    $\mathcal{P}$ and $\mathcal{V}$ compute $y' \leftarrow y_L + \alpha \cdot y_R \bmod p$, $\mathsf{C}' \leftarrow \mathsf{C}_L + \alpha \cdot \mathsf{C}_R$, $b' \leftarrow b \cdot 2^\lambda$.
  18.    $\mathcal{P}$ computes $f' \leftarrow f_L + \alpha \cdot f_R \in \mathbb{Z}_p[X_1, \ldots, X_{\mu'}]$
  19.    $\mathcal{P}$ computes $\hat{f}' \leftarrow \hat{f}_L + \alpha \cdot \hat{f}_R \in \mathbb{Z}[X_1, \ldots, X_{\mu'}]$ // $\hat{f}'$ and $f'$ are $\mu'$-linear
  20.    $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalB}(\mathsf{pp}, \mathsf{C}', \vec{z}' = (z_1, \ldots, z_\mu'), y', \mu', b'; f', \hat{f}')$
```

2.5.4 Security Analysis

The new security analysis is in a self-contained document in the appendix. We, briefly, restate the main lemmas and theorems but refer the reader to the appendix for more details.

We show in Lemma 10 that the DARK polynomial commitment scheme is binding.

Lemma 5. *The polynomial commitment scheme is correct for μ-linear polynomials in $\mathbb{Z}_p[X]$.*

The proof of this lemma is in Section 2.8. Next is the main security theorem, which states that the evaluation protocol has witness-extended emulation.

Theorem 1. *Let $\mathsf{CSZ}_{\mu,\lambda} = 8\mu^2 + \log_2(2\mu)\lambda$. Let $\mathsf{EBL}_{\mu,\lambda} = \lambda \cdot \mu$ and $\mathsf{CB}_{p,\mu,\lambda} = \lambda \cdot \mu + \log_2 p$. Let com be the DARK commitment scheme as described in Lemma 10. There exists a pair of predicates ϕ such that the μ-round DARK polynomial commitment evaluation protocol Eval' with λ-bit challenges, a group of unknown order GGen, and $\log q \geq 4(\lambda+1+\mathsf{CSZ}_{\mu,\lambda}) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} + 1$ is $(2^{(\mu)}, \frac{3\mu}{2^\lambda}, \mathsf{com}, \phi)$-almost-special-sound .*[5]

As a corollary, under the adaptive root assumption for GGen, the DARK polynomial commitment scheme with the same parameters has witness-extended-emulation (Definition 7).

2.5.5 Optimizations

We present several ideas for optimizing the performance of the Eval protocol.

Precomputation. The prover has to compute powers of G as large as $q^{2^\mu-1}$. While this can be done in quasi-linear time, this expense can be shifted to a preprocessing phase in which all elements $\mathsf{G}^{q^i}, i \in \{1, \ldots, 2^\mu - 1\}$ are computed. Since for coefficient $|f_i| \leq -\frac{p-1}{2}$ this allows the computation of $\mathsf{G}^{f(q)}$ in $O(\lambda 2^\mu)$ group operations as opposed to $O(\lambda 2^\mu \mu)$. In addition to reducing the prover's workload, this optimization enables parallelizing it. The computation of the PoE proofs can similarly be parallelized. The prover can express each Q

[5]The $\mathsf{CSZ}_{\mu,\lambda}$ value can be replaced with values from the table in Lemma 8

as a power of G which enables pre-computation of powers of G and parallelism as described by Boneh *et al.* [41].

The pre-computation also enables the use of multi-scalar multiplication techniques [135]. Boneh *et al.* [41] and Wesolowski [164] showed how to use these techniques to reduce the complexity of the PoE prover. The largest PoE exponent $q^{2^{\mu-1}}$ has $O(\lambda 2^{\mu}\mu)$ bits. Multi-scalar multiplication can therefore reduce the prover work to $O(\lambda 2^{\mu})$ instead of $O(\lambda 2^{\mu}\mu)$. For univariate polynomials of degree d this translates to prover work that is $O(\lambda d)$.

Two group elements per round. In each round the verifier has a value C and receives C_L and C_R such that $\mathsf{C}_L + q^{2^{\mu'-1}} \cdot \mathsf{C}_R = \mathsf{C}$. This is redundant. It suffices that the verifier sends C_R. The verifier could now compute $\mathsf{C}_L \leftarrow \mathsf{C} - q^{2^{\mu'-1}}\mathsf{C}_R$, but this is expensive as it involves an scalar multiplication by q^d. Instead, the verifier infers $q^{2^{\mu'-1}} \cdot \mathsf{C}_R$ from the PoE: the prover's message is Q and the verifier can directly compute $q^{2^{\mu'-1}} \cdot \mathsf{C}_R \leftarrow \ell \cdot \mathsf{Q} + r \cdot \mathsf{C}_R$ for a challenge ℓ and $r \leftarrow q^{2^{\mu'-1}} \bmod \ell$. From this the verifier infers $\mathsf{C}_L \leftarrow \mathsf{C} - q^{2^{\mu'-1}} \cdot \mathsf{C}_R$. The security of PoE does not require that $q^{d'+1} \cdot \mathsf{C}_R$ be sent before the challenge ℓ as it is uniquely defined by C_R and $q^{2^{\mu'-1}}$. The same optimization can be applied to the non-interactive variant of the protocol.

Similarly the verifier can infer y_L as $y_L \leftarrow y - z^{2^{\mu'-1}} y_R$. This reduces the communication to two group elements per round and 1 field element. Additionally the prover sends f which has roughly the size of μ field elements, which increases the total communication to roughly 2μ elements in $\mathbb{G}$ and 2μ elements in $\mathbb{Z}_p$.

Evaluation at multiple points The protocol and the security proof extend naturally to the evaluation in a vector of points $\boldsymbol{z}$ resulting in a vector of values $\boldsymbol{y}$, where both are members of $\mathbb{Z}_p^k$. The prover still sends $\mathsf{C}_L \in \mathbb{G}$ and $\mathsf{C}_R \in \mathbb{G}$ in each round and additionally $\boldsymbol{y}_L, \boldsymbol{y}_R \in \mathbb{Z}_p^k$. In the final round the prover only sends a single integer f such that $\mathsf{G}^f = \mathsf{C}$ and $f \bmod p = y$.

This is significantly more efficient than independent executions of the protocol as the encoding of group elements is usually much larger than the encoding of elements in $\mathbb{Z}_p$.

Using the optimization above, the marginal cost with respect to k of the protocol is a single element in $\mathbb{Z}_p$. If $\lambda = \lceil \log_2(p) \rceil$ is 120, then this means evaluating the polynomial at an additional point increases the proof size by only 15μ bytes.

Joining Evals. In many applications such as compiling polynomial IOPs to SNARKs (see Section 2.6) multiple polynomial commitments need to be evaluated at the same point z. This can be done efficiently by taking a random linear combination of the polynomials and evaluating that combination at z. The prover simply sends the evaluations of the individual polynomials and then a single evaluation proof for the combined polynomials. The communication cost for evaluating m polynomials at 1 point is still linear in m but only because the evaluation of each polynomial at the point is being sent. The size of the eval proof, however, is independent of m. Taking a random linear combination does increase the bound on q slightly, as shown in Theorem 2 which is presented below.

$$\mathcal{R}_{\mathsf{JE}}(\mathsf{pp}) = \left\{ \langle (\mathsf{C}_1, \mathsf{C}_2, z, y_1, y_2, d), (\hat{f}_1, \hat{f}_2) \rangle : \begin{array}{l} \mathsf{C}_1, \mathsf{C}_2 \in \mathbb{G} \\ z, y_1, y_2 \in \mathbb{Z}_p \\ f_1, f_2 \in \mathbb{Z}(b)[\vec{X} = (X_1, \ldots, X_\mu)] \\ (\mathsf{C}_1, \vec{z}, y_1, \mu) \in \mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) \\ (\mathsf{C}_2, \vec{z}, y_2, \mu) \in \mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) \end{array} \right\}$$

```
$\mathsf{JoinedEval}(\mathsf{pp}, \mathsf{C}_1, \mathsf{C}_2, \vec{z} \in \mathbb{Z}_p^\mu, y_1 \in \mathbb{Z}_p, y_2 \in \mathbb{Z}_p, \mu; \hat{f}_1 \in \mathbb{Z}(b)[\vec{X}], \hat{f}_2 \in \mathbb{Z}(b)[\vec{X}])$ :
Statement: $(\mathsf{pp}, \mathsf{C}_1, \mathsf{C}_2, z, y_1, y_2, b, d) \in \mathcal{R}_{\mathsf{JE}}$
  1. $\mathcal{V}$ samples $\alpha \xleftarrow{\$} [0, 2^\lambda)$ and sends it to $\mathcal{P}$
  2. $\mathcal{P}$ and $\mathcal{V}$ compute $y' \leftarrow y_1 + \alpha \cdot y_2 \bmod p$
  3. $\mathcal{P}$ computes $\hat{f}'(\vec{X}, Y) \leftarrow \hat{f}_1(\vec{X}) + Y \hat{f}_2(\vec{X})$ and $f' = \hat{f}' \bmod p$
  4. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalB}(\mathsf{pp}, \mathsf{C}_1 + q^{2^\mu} C_2, (\vec{z}, \alpha), y', \mu + 1; f', \hat{f}')$[a]
```

[a] The prover and verifier don't actually need to compute $\mathsf{C}_1 + q^{2^\mu} C_2$ as the next prover message can be computed directly from C_1 and C_2.

Theorem 2. *Let* $\mathsf{CSZ}_{\mu,\lambda}, \mathsf{EBL}_{\mu,\lambda}$ *and* $\mathsf{CB}_{p,\mu,\lambda}$ *be defined as in theorem 1 . Let* $\log q \geq 4(\lambda + 1 + \mathsf{CSZ}_{\mu+1,\lambda}) + \mathsf{EBL}_{\mu+1,\lambda} + \mathsf{CB}_{p,\mu+1,\lambda} + 1$. *Under the adaptive root assumption for GGen, the* $\mathsf{JoinedEval}$ *protocol has witness-extended-emulation (Definition 7) for the relation* $\mathcal{R}_{\mathsf{JE}}$.

Proof. Security directly follows from Theorem 1 as C_1, C_2 is a binding virtual commitment to the $\mu + 1$-linear polynomial $f_1 + Y \cdot f_2$. That is, $C = C_1 + q^{2^\mu} \cdot C_2$ can be computed from C_1, C_2 thus if C is a binding commitment then so is (C_1, C_2). We will show that the protocol has 2-special soundness using the extractor on the $\mu + 1$-linear evaluation protocol. Using two executions with challenges α and α' we call the DARK extractor. This returns a witness polynomials $f, f' \in \mathbb{Z}_p$ such that $f(\vec{z}, \alpha) = y_L + \alpha y_R$ and $f(\vec{z}, \alpha') = y_L + \alpha' y_R$. If $f \neq f'$ the we have a break of the binding property of the commitment scheme as $C_L + q^{2^\mu} C_R$ is both a commitment to f and to f'. Otherwise we get that $(\alpha - \alpha')^{-1}(f(\vec{z}, \alpha) - f(\vec{z}, \alpha')) = f_R(\vec{z}) = y_R$ and $(\alpha' - \alpha)^{-1}(\alpha' f(\vec{z}, \alpha) - \alpha f(\vec{z}, \alpha')) = f_L(\vec{z}) = y_L$. □

We can additionally combine this optimization with the previous optimization of evaluating a single polynomial at different points. This allows us to evaluate m polynomials at k points with very little overhead. The prover groups the polynomials by evaluation points and first takes linear combinations of the polynomials with the same evaluation point and computes y_1 to y_k using the same linear combinations. Then it takes another combination of the joined polynomials. In each round of the Eval protocol the prover sends $y_{L,1}$ through $y_{L,k}$, i.e. one field element per evaluation point and computes $y_{R,1}$ through $y_{R,k}$. In the final step the prover sends f and the verifier can check whether the final y values are all equal to $f \bmod p$. This enables an Eval proof of m, μ-linear polynomials at k points using only 2μ group elements and $(1 + k)\mu$ field elements.

Evaluating the polynomial over multiple fields The polynomial commitment scheme is highly flexible. For example, it does not specify a prime field $\mathbb{Z}_p$. It instead commits to an integer polynomial with bounded coefficients. That integer polynomial can be evaluated modulo arbitrary primes which are exponential in the security parameter λ as the soundness error is proportional to its inverse. Note that q also needs large enough such that the scheme is secure for the given prime p and linearity μ (see Theorem 1). The second condition, however, can be relaxed. A careful analysis shows that as long as p is exponential in λ and q is sufficiently large, the scheme is secure. So as long as $\log q \geq 4(\lambda + 1 + \mathsf{CSZ}_{\mu,\lambda}) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{b,\mu,\lambda} + 1$ one can evaluate μ-linear polynomial with coefficients bounded by b over any exponential prime field.

Additionally, the proof elements C_L, $\mathsf{C}_R \in \mathbb{G}$ are independent of the field over which the polynomial is evaluated. This means that it is possible to evaluate a committed polynomial $f(X) \in \mathbb{Z}(b)$ over two separate fields $\mathbb{Z}_p$ and $\mathbb{Z}_{p'}$ in parallel using only 2μ group elements and sending the evaluations y modulo $p \cdot p'$. The verifier performs all operations on y modulo $p \cdot p'$ as well.

This property can be used to efficiently evaluate the polynomial modulo a large integer m by choosing multiple λ bit primes $p_1, \ldots p_k$ such that $\prod_{i=1}^{k} p_i \geq m$ and using the Chinese Remainder Theorem to simulate the evaluation modulo m.

2.5.6 Performance

The polynomial commitment scheme has logarithmic proof size and verifier time in the degree d of the committed polynomial. It has highly batchable proofs and it is possible to evaluate n degree d polynomials at k points using only $2\log_2(d+1)$ group elements and $(k+1)\log_2(d+1)$ field elements (see Section 2.5.5). Note that this means the proof size is independent of n and linear in k but with a small constant ($15\log(d)$ bytes). We describe the performance of our scheme for different settings for uni- and multivariate instantiations in Table 2.1.

2.5.7 Comparison to Other Polynomial Commitment Schemes

Based on Pairings

The polynomial commitment by Kate *et al.* [105] has evaluation proofs that consist of only a single element in a bilinear group and verifying an evaluation requires only a single pairing

Operation	$\vert\mathsf{pp}\vert$	Prover	Verifier	Communication
$\mathsf{Commit}(f(X))$	$1\ \mathbb{G}$	$O(\lambda d \log(d))\mathbb{G}$	-	$1\mathbb{G}$
$\mathsf{Commit}(f(X))$	$d\ \mathbb{G}$	$O(\frac{\lambda d}{\log(d)})\mathbb{G}$	-	$1\mathbb{G}$
$f(z) = y \in \mathbb{Z}_p$	$1\ \mathbb{G}$	$O(\lambda \log(d) d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G} + 2\log(d)\mathbb{Z}_p$
$f(z) = y \in \mathbb{Z}_p$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G} + 2\log(d)\mathbb{Z}_p$
$f(\boldsymbol{z}) = \boldsymbol{y} \in \mathbb{Z}_p^k$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G} + (k+1)\log(d)\mathbb{Z}_p$
$f(z) = y, g(z) = y' \in \mathbb{Z}_p$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G} + 2\log(d)\mathbb{Z}_p$

Table 2.1: $\mathbb{G}$ denotes the size of a group element for communication and a single group operation for computation. $\mathbb{Z}_p$ denotes the size of a field element, *i.e.*, λ bits. $|\mathsf{pp}|$ is the size of the public parameters (which is greater than one $\mathbb{G}$ when preprocessing is used), and d the degree of the polynomial. Rows 3-6 are for Eval proofs of different statements.

computation. However, this asymptotically optimal performance comes at the cost of a trusted setup procedure that outputs a structured reference string whose size is linear in the degree of the polynomial. Our DARK polynomial commitment scheme requires no trusted setup but pays for this reduced trust requirement with a proof size and verification work that scale logarithmically in the degree of the polynomial.

In the multivariate setting, our scheme is logarithmic in the total number of coefficients: $\mu \log(d)$ for a μ-variate polynomial of degree d in each variable. The multivariate extension of Kate *et al.*'s commitment scheme [170] evaluation proofs consist of μ group elements.

Based on Discrete Logarithms

Bulletproofs [43, 51] is a proof system based on prime order groups in which the discrete logarithm is hard. As a core component it relies on an inner product argument which can be used as a polynomial commitment (see [160]). The polynomial commitment has logarithmic evaluation proofs with great constants. Unfortunately, the verifier time is linear in the size of the polynomial, i.e. $(d+1)^{\mu}$ for a μ- variate degree d polynomial. The more general version of the commitment [43] can also give evaluation proofs with square root verifier time and square root proof size.

Based on Merkle Trees of Reed-Solomon Codewords

The FRI protocol [21] is an efficient interactive oracle proof (IOP) that a committed oracle is close to a Reed-Solomon codeword, meaning that the prover commits to large sequences of field elements and the verifier queries only a few specific elements rather than reading the entire sequence. The abstract functionality is cryptographically compiled with a Merkle tree, which results in constant-size commitments and element queries that are logarithmic in the length of the codeword, *i.e.*, the size of the oracle. FRI has been used in multiple recent zero-knowledge proof systems such as STARK [23], Aurora [28], and Fractal [64].

Since this oracle is a Reed-Solomon codeword, it represents the evaluations of a low-degree polynomial f on an evaluation set $S \subset \mathbb{F}$. In order to be used as a polynomial commitment scheme, the protocol needs to permit querying the polynomial outside of the evaluation set. DEEP-FRI [31] shows that this is possible and two recent works [169, 106] makes the connection explicit by building a polynomial commitment scheme from FRI. This FRI-based polynomial commitment scheme have evaluation proofs of size and verifier time $O(\lambda \log^2(d))$ where λ is the security parameter and $d = \deg(f)$. To date, no extension to multivariate polynomials exists for FRI. The commitment relies only on symmetric cryptography and is plausibly quantum resistant.

Comparison

In Table 2.2 we give a comparison between different polynomial commitment schemes in the literature. In particular, we evaluate the size of the reference string ($|\mathsf{pp}|$), the prover and verifier time, as well as the size of the evaluation proof ($|\pi|$). Column 2 indicates whether the setup is transparent, *i.e.*, whether the reference string is structured. The symbol $\mathbb{G}_U$ denotes a group of unknown order, $\mathbb{G}_B$ a group with a bilinear map (pairing), and $\mathbb{G}_P$ a group with prime (and known) order. Furthermore, MUL refers to scalar multiplications of a λ bit number in these groups, and H is either the size of a hash output, or the time it takes to compute a hash, depending on context.

Note that even when precise factors are given, the numbers should be interpreted as estimates. For example we chose to not display smaller order terms. Note also that the prover time for the group based schemes could be brought down by a log factor when using multi-scalar multiplication techniques.

Scheme	Transp.	$\lvert \mathsf{pp} \rvert$	Prover	Verifier	$\lvert\pi\rvert$
DARK *(this work)*	yes	$O(1)$	$O(d^{\mu}\mu\log(d))$ MUL	$3\mu\log(d)$ MUL	$2\mu\log(d)\ \mathbb{G}_U$
Based on Pairings	no	$d^{\mu}\ \mathbb{G}_B$	$O(d^{\mu})$ MUL	μ $\mathsf{Pairing}$	$\mu\ \mathbb{G}_B$
[43, $\sqrt{\cdot}$]	yes	$\sqrt{d^{\mu}}\mathbb{G}_P$	$O(d^{\mu})$ MUL	$O(\sqrt{d^{\mu}})\mathsf{MUL}$	$O(\sqrt{d^{\mu}})\ \mathbb{G}_P$
Bulletproofs	yes	$2d^{\mu}\mathbb{G}_P$	$O(d^{\mu})$ MUL	$O(d^{\mu})\mathsf{MUL}$	$2\mu\log(d)\ \mathbb{G}_P$
FRI-based ($\mu = 1$)	yes	$O(1)$	$O(\lambda d)$ H	$O(\lambda\log^2(d))$ H	$O(\lambda\log^2(d))$ H

Table 2.2: Comparison table between different polynomial commitment schemes for an μ-variate polynomial of degree d.

2.6 Transparent SNARKs via Polynomial IOPs

2.6.1 Algebraic Linear IOPs

An *interactive oracle proof (IOP)* [29, 139] is a multi-round interactive PCP: in each round of an IOP the verifier sends a message to the prover and the prover responds with a polynomial length proof, which the verifier can query via random access. A t-round ℓ-query IOP has t rounds of interaction in which the verifier makes exactly ℓ queries in each round. Linear IOPs [39] are defined analogously except that in each round the prover sends a *linear* PCP [101], in which the prover sends a single proof vector $\boldsymbol{\pi} \in \mathbb{F}^m$ and the verifier makes *linear queries* to π. Specifically, the PCP gives the verifier access to an oracle that receives queries of the form $\mathbf{q} \in \mathbb{F}^m$ and returns the inner product $\langle \boldsymbol{\pi}, \mathbf{q} \rangle$.

Bitansky *et al.* [36] defined a linear PCP to be of *degree* (d_Q, d_V) if there is an explicit circuit of degree d_Q that derives the query vector from the verifier's random coins, and an explicit circuit of degree d_V that computes the verifier's decision from the query responses. In a multi-query PCP, d_Q refers to the maximum degree over all the independent circuits computing each query. Bitansky *et al.* called the linear PCP *algebraic* for a security parameter λ if it has degree $(\mathsf{poly}(\lambda), \mathsf{poly}(\lambda))$. The popular linear PCP based on *Quadratic Arithmetic Programs* (QAPs) implicit in the GGPR protocol [89] and follow-up works is an algebraic linear PCP with $d_Q \in O(m)$ and $d_V = 2$, where m is the size of the witness.

For the purposes of the present work, we are only interested in the algebraic nature of the query circuit and not the verifier's decision circuit. Of particular interest are linear PCPs where each query-and-response interaction corresponds to the evaluation of a fixed μ-variate degree d polynomial at a query point in $\mathbb{F}^\mu$. This description is equivalent to saying that the PCP is a vector of length $m = \binom{d+\mu}{\mu}$ and the query circuit is the vector of all μ-variate monomials of degree at most d (in some canonical order) evaluated at a point in $\mathbb{F}^\mu$. We call this a (μ, d) *Polynomial PCP* and define *Polynomial IOPs* analogously. As we will explain, we are interested in Polynomial PCPs where $\mu \ll m$ because we can cryptographically compile them into succinct arguments using polynomial commitments, in the same way that Merkle trees are used to compile classical (point) IOPs.

In general, evaluating the query circuit for a linear PCP requires $\Omega(m)$ work. However, a general "bootstrapping" technique can reduce the work for the verifier: the prover expands the verifier's random coins into a full query vector, and then provides the verifier with a second PCP demonstrating that this expansion was computed correctly. It may also help to allow the verifier to perform $O(m)$ work in a one-time preprocessing stage (for instance, to check the correctness of a PCP oracle), enabling it to perform sublinear "online" work when verifying arbitrary PCPs later. We call this a *preprocessing IOP*. In fact, we will see that any t-round (μ, d) algebraic linear IOP can be transformed into a $(t+1)$-round Polynomial IOP in which the verifier preprocesses (μ, d) Polynomial PCPs, at most one for each distinct query.

We recall the formal definition of public-coin linear IOPs as well an algebraic linear IOPs. Since we are not interested in the algebraic nature of the decision algorithm, we omit specifying the decision polynomial. From here onwards we use algebraic linear IOP as shorthand for algebraic *query* linear IOP.

Definition 3 (Public-coin linear IOP)**.** *Let $\mathcal{R}$ be a binary relation and $\mathbb{F}$ a finite field. A t-round ℓ-query public-coin linear IOP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ϵ and knowledge error δ and query length $\mathbf{m} = (m_1, ..., m_t)$ consists of two stateful PPT algorithms,* *the* prover $\mathcal{P}$*, and the* verifier $\mathcal{V} = (\mathcal{Q}, \mathcal{D})$*, where the verifier consists in turn of a public deterministic* query generator $\mathcal{Q}$ *and a* decision algorithm $\mathcal{D}$*, that satisfy the following requirements:*

Protocol syntax. For each ith round there is a prover state $\mathsf{st}_i^{\mathcal{P}}$ and a verifier state $\mathsf{st}_i^{\mathcal{V}}$. For any common input x and $\mathcal{R}$ witness w, at round 0 the states are $\mathsf{st}_0^{\mathcal{P}} = (x, w)$ and $\mathsf{st}_0^{\mathcal{V}} = x$. In the ith round (starting at $i = 1$) the prover outputs a single[6] proof oracle $\mathcal{P}(\mathsf{st}_{i-1}^{\mathcal{P}}) \to \boldsymbol{\pi}_i \in \mathbb{F}^{m_i}$. The verifier samples public random coins $coins_i \overset{\$}{\leftarrow} \{0,1\}^*$ and the query generator computes a query matrix from the verifier state and these coins: $\mathcal{Q}(\mathsf{st}_{i-1}^{\mathcal{V}}, coins_i) \to \mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$. The verifier obtains the linear oracle response vector $\boldsymbol{\pi}_i^\top \mathbf{Q}_i = \mathbf{a}_i \in \mathbb{F}^{1\times\ell}$. The updated prover state is $\mathsf{st}_i^{\mathcal{P}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{P}}, \mathbf{Q}_i)$ and verifier state is $\mathsf{st}_i^{\mathcal{V}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{V}}, coins_i, \mathbf{a}_i)$ Finally, $\mathcal{D}(\mathsf{st}_t^{\mathcal{V}})$ returns 1 or 0.

*(*Querying prior round oracles*: The syntax can be naturally extended so that in the ith round the verifier may query any oracle, whether sent in the ith round or earlier.)*

Argument of Knowledge. As a proof system, $(\mathcal{P}, \mathcal{V})$ satisfies perfect completeness, soundness with respect to the relation $\mathcal{R}$ and with soundness error ϵ, and witness-extended emulation with respect $\mathcal{R}$ with knowledge error δ.

*Furthermore, a linear IOP is **stateless** if for each $i \in [t]$, $\mathcal{Q}(\mathsf{st}_{i-1}^{\mathcal{V}}, coins_i) = \mathcal{Q}(i, coins_i)$. It has **algebraic queries** if, additionally, for each $i \in [t]$, the map $coins_i \xmapsto{\mathcal{Q}(i,\cdot)} \mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$ decomposes into two maps, $coins_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \boldsymbol{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$, where $\boldsymbol{\Sigma}_i \in \mathbb{F}^{\mu_i \times \ell}$ is a matrix of $\mu_i < m_i$ rows and ℓ and $\mathcal{Q}_1(i, \cdot)$ is described by ℓ μ_i-variate polynomial functions of degree at most $d = \mathsf{poly}(\lambda)$: $\vec{p}_1, \ldots, \vec{p}_\ell : \mathbb{F}^{\mu_i} \to \mathbb{F}^{m_i}$ such that for all $k \in [\ell]$, $\vec{p}_k(\boldsymbol{\sigma}_{i,k}) = \mathbf{q}_{i,k}$, where $\boldsymbol{\sigma}_{i,k}$ and $\mathbf{q}_{i,k}$ denote the kth column of $\boldsymbol{\Sigma}_i$ and $\mathbf{Q}_i$, respectively.*

Definition 4 (HVZK for public-coin linear IOPs)**.** *Let $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle}(\mathcal{V})$ denote the view*

[6]The prover may also output more than one proof oracle per round, however this doesn't add any power since two proof oracles of the same size may be viewed as a single (concatenated) oracle of twice the length.

*of the verifier in the t-round ℓ-query interactive protocol described in Definition 3 on inputs (x, w) with prover algorithm $\mathcal{P}$ and verifier $\mathcal{V}$, consisting of all public-coin challenges and oracle outputs (this view is equivalent to the final state $\mathsf{st}_t^{\mathcal{V}}$). The interactive protocol has δ-**statistical honest-verifier zero-knowledge** if there exists a probabilistic polynomial time algorithm $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the distribution $\mathcal{S}(x)$ is δ-close to $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}(\mathcal{V})$ (as distributions over the randomness of $\mathcal{P}$ and random public-coin challenges).*

We note that the separation into two maps $coins_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \mathbf{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$ subtly relaxes the definition of Bitansky *et al.*, which instead requires that $\mathbf{Q}_i$ be determined via $\vec{p}_1, \ldots, \vec{p}_\ell$ evaluated at a random $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\mu_i}$. The Bitansky *et al.* definition corresponds to the special case that $\mathcal{Q}_0(i, \cdot)$ samples a random element of $\mathbb{F}^{\mu_i}$ based on $coins_i$. The point is that $\mathcal{Q}_0$ can also do other computations that do not necessarily sample $\boldsymbol{r}$ uniformly, or even output a matrix rather than a vector. The separation into two steps is only meaningful when μ_i is smaller than m_i. The significance to SNARK constructions is that the query can be represented compactly as $\mathbf{\Sigma}_i$, and the prover will take advantage of the algebraic map $\mathcal{Q}_1(i, \cdot)$ to demonstrate that $\mathbf{\Sigma}_i$ was expanded correctly into $\mathbf{Q}_i$ and applied to the proof oracle π_i. We first present a standalone definition of Polynomial IOPs, and then explain how it is a special case of Algebraic Linear IOPs.

Definition 5 (Public coin Polynomial IOP)**.** *Let $\mathcal{R}$ be a binary relation and $\mathbb{F}$ a finite field. Let $\mathbf{X} = (X_1, \ldots, X_\mu)$ be a vector of μ indeterminates. A (μ, d) Polynomial IOP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ϵ and knowledge error δ consists of two stateful PPT algorithms, the* prover $\mathcal{P}$, *and the* verifier $\mathcal{V}$, *that satisfy the following requirements:*

Protocol syntax. For each ith round there is a prover state $\mathsf{st}_i^{\mathcal{P}}$ and a verifier state $\mathsf{st}_i^{\mathcal{V}}$. For any common input x and $\mathcal{R}$ witness w, at round 0 the states are $\mathsf{st}_0^{\mathcal{P}} = (x, w)$ and $\mathsf{st}_0^{\mathcal{V}} = x$. In the ith round (starting at $i = 1$) the prover outputs a single proof oracle $\mathcal{P}(\mathsf{st}_{i-1}^{\mathcal{P}}) \to \pi_i$, which is a polynomial $\pi_i(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$. The verifier deterministically computes the query matrix $\mathbf{\Sigma}_i \in \mathbb{F}^{\mu \times \ell}$ from its state and a string of public random bits $coins_i \xleftarrow{\$} \{0, 1\}^*$, i.e,

$\mathcal{V}(\mathsf{st}^{\mathcal{V}}_{i-1}, coins_i) \to \mathbf{\Sigma}_i$. This query matrix is interpreted as a list of ℓ points in $\mathbb{F}^{\mu}$ denoted $(\boldsymbol{\sigma}_{i,1}, \ldots, \boldsymbol{\sigma}_{i,\ell})$. The oracle π_i is queried on all points in this list, producing the response vector $(\pi_i(\boldsymbol{\sigma}_{i,1}), \ldots, \pi_\ell(\boldsymbol{\sigma}_{i,\ell})) = \mathbf{a}_i \in \mathbb{F}^{1\times\ell}$. The updated prover state is $st^{\mathcal{P}}_i \leftarrow (st^{\mathcal{P}}_{i-1}, \mathbf{\Sigma}_i)$ and verifier state is $st^{\mathcal{V}}_i \leftarrow (st^{\mathcal{V}}_{i-1}, \mathbf{\Sigma}_i, \mathbf{a}_i)$. Finally, $\mathcal{V}(st^{\mathcal{V}}_t)$ returns 1 or 0.

*(*Extensions: multiple and prior round oracles; various arity. *The syntax can be naturally extended such that multiple oracles are sent in the ith round; that the verifier may query oracles sent in the ith round or earlier; or that some of the oracles are polynomials in fewer variables than μ.)*

Argument of Knowledge. As a proof system, $(\mathcal{P}, \mathcal{V})$ satisfies perfect completeness, soundness with respect to the relation $\mathcal{R}$ and with soundness error ϵ, and witness-extended emulation with respect $\mathcal{R}$ with knowledge error δ.

*Furthermore, a Polynomial IOP is **stateless** if for each $i \in [t]$, $\mathcal{V}(\mathsf{st}^{\mathcal{V}}_{i-1}, coins_i) = \mathcal{V}(i, coins_i)$.*

Polynomial IOPs as a subclass of Algebraic Linear IOPs In a Polynomial IOP, the two-step map $coins_i \xmapsto{\mathcal{V}(i,\cdot)} (\boldsymbol{\sigma}_{i,1}, \ldots, \boldsymbol{\sigma}_{i,\ell}) \xmapsto{\mathbf{M}} (\mathbf{q}_{i,1}, \ldots, \mathbf{q}_{i,\ell})$ is a special case of the two-step map $coins_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \mathbf{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$ in an algebraic linear IOP. Here $\mathbf{M} : \mathbb{F}^{\mu} \to \mathbb{F}^{m}$ represents the vector of monomials of degree at most d (in some canonical order) and the map associated with $\mathbf{M}$ is evaluation. Note that there are $m = \binom{\mu+d}{d}$ such monomials. Furthermore, for any $\mathbf{q}_{i,k}$, the inner product $\boldsymbol{\pi}_i^{\mathsf{T}}\mathbf{q}_{i,k}$ corresponds to the evaluation at $\boldsymbol{\sigma}_{i,k}$ of the polynomial $\pi_i(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, whose coefficient vector (in the same canonical monomial order) is equal to $\boldsymbol{\pi}_i$.

2.6.2 Polynomial IOP Reductions

In this section we show that one can construct any algebraic linear IOP from a (multivariate) Polynomial IOP. This construction rests on two tools for univariate Polynomial IOPs that we cover first:

- *Coefficient queries.* The verifier verifies that an indicated coefficient of a polynomial oracle has a given value.
- *Inner products.* The verifier verifies that the inner product of the coefficient vectors of two polynomial oracles equals a given value.

Coefficient queries

The following is a $(1, d)$-Polynomial IOP for the statement $f_i = a$ with respect to a polynomial $f(X) = \sum_{j=0}^{d} f_j X^j$.

- *Prover*: Split $f(X)$ about the term X^i into $f_L(X)$ (of degree at most $i-1$) and $f_R(X)$ (of degree at most $d-i-1$) such that $f(X) = f_L(X) + aX^i + X^{i+1}f_R(X)$. Send polynomials $f_L(X)$ and $f_R(X)$.
- *Verifier*: Sample uniform random $\beta \xleftarrow{\$} \mathbb{F}_p$ and query for $y_L \leftarrow f_L(\beta)$, $y_R \leftarrow f_R(\beta)$, and $y \leftarrow f(\beta)$. Check that $y = y_L + a\beta^i + \beta^{i+1}y_R \bmod p$ and return 0 (abort) if not. Otherwise output 1 (accept).

The verifier only accepts given proof oracles for polynomials f, f_L, and f_R in $\mathbb{F}_p[X]$ of degree at most d, $i-1$ and $d-i-1$ such that $f(\beta) = f_L(\beta) + a\beta^i + \beta^{i+1}f_R(\beta)$ for random $\beta \xleftarrow{\$} \mathbb{F}$. Via the Schwartz-Zippel lemma, if $f(X) \neq f_L(X) + aX^i + X^{i+1}f_R(X)$ then the verifier would accept with probability at most $d/|\mathbb{F}|$, because the highest degree term in this equation is $X^{i+1}f_R(X)$ and its degree is at most d. This implies that a is the ith coefficient of f.

Note that this description assumes that the verifier is assured that the proof oracles for f_L and f_R have degrees $i-1$ and $d-i-1$, respectively. If no such assurance is given, then $f_L(X)$ should be shifted by $d-i+1$ digits. In particular, the proof oracle should $f_L^\star(X) = X^{d-i+1}f_L(X)$, in which case the verifier obtains the evaluation $y_L^\star = y_L\beta^{d-i+1}$ along with an assurance that $f_L^\star(X)$ has degree at most d. The verifier then tests $y = (\beta^{d-i+1})^{-1}y_L^\star + a\beta^i + \beta^{i+1}y_R$. This test admits false positives with probability at most $2d/|\mathbb{F}|$.

Inner product

The following is an IOP where the prover first sends two degree d univariate polynomial oracles f, g and proves to the verifier that $\langle \mathbf{f}, \mathbf{g}^r \rangle = a$ where $\mathbf{f}, \mathbf{g}$ denote the coefficient vectors of f, g respectively and $\mathbf{g}^r$ is the reverse of $\mathbf{g}$. This argument is sufficient for our application to transforming algebraic linear IOPs into Polynomial IOPs. It is also possible to prove the inner product $\langle \mathbf{f}, \mathbf{g} \rangle$ by combining this IOP together with another one that probes the relation $g(X) = X^d g^r(X^{-1})$ in a random point $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$, and thereby shows that $\mathbf{g}$ and $\mathbf{g}^r$ have the same coefficients only reversed. We omit this more elaborate construction as it is not needed for any of our applications.

- *Prover*: Sends proof oracles for $f(X)$, $g(X)$, and the degree $2d$ polynomial product $h(X) = f(X) \cdot g(X)$ to the verifier.
- *Verifier*: Chooses $\beta \xleftarrow{\$} \mathbb{F}$ and queries for $y_1 \leftarrow f(\beta)$, $y_2 \leftarrow g(\beta)$, and $y_3 \leftarrow h(\beta)$. Check that $y_1 y_2 = y_3$ and return 0 (abort) if not.
- Prover and verifier engage in the 1 round IOP (Section 2.6.2) for proving that the dth coefficient (*i.e.*, on term X^d) of $h(X)$ is equal to a. (Note that the proof oracles for this subprotocol can all be sent in the first round, so this does not add an additional round).

Via Schwartz-Zippel, if $h(X) \neq f(X) \cdot g(X)$ then the verifier's check $y_1 y_2 = y_3$ at the random point β fails with probability at least $(|\mathbb{F}| - 2d)/|\mathbb{F}|$. Observe that the middle coefficient of $h(X)$ is equal to $\sum_{i=0}^{d} f_i g_{d-i} = \sum_{i=0}^{d} f_i g_i^r = \langle \mathbf{f}, \mathbf{g}^r \rangle = a$.

Reducing algebraic linear IOPs to Polynomial IOPs

Theorem 3. *Any public-coin t-round stateless algebraic linear IOP can be implemented with a $t+1$-round Polynomial IOP with preprocessing. Suppose the original ℓ-query IOP is (μ, d) algebraic with query length $(m_1, ..., m_t)$ then the resulting Polynomial IOP has for each $i \in [t]$: 2ℓ degree m_i univariate polynomial oracles, ℓ pre-processed multivariate oracles* *of degree d and $\mu + 1$ variables, ℓ degree $2m_i$ univariate polynomial oracles and 2ℓ degree $2m_i$ univariate polynomial oracles. There is exactly one query to each oracle on a random point in $\mathbb{F}$. The soundness loss of the transformation is* $\mathsf{negl}(\lambda)$ *for a sufficiently large field (*i.e.*, whose cardinality is exponential in λ).*

Proof. By definition of a (μ, d) algebraic linear IOP, in each ith round of the IOP there are ℓ query generation functions $\vec{p}_{i,1}, \dots, \vec{p}_{i,\ell} : \mathbb{F}^\mu \to \mathbb{F}^{m_i}$, where each $\vec{p}_{i,k}$ is a vector whose jth component is a μ-variate degree-d polynomial $p_{i,k,j}$. These polynomials are applied to a seed matrix $\boldsymbol{\sigma}_{i,k} \in \mathbb{F}^\mu$ (which is identifiable with or derived from the verifier's ith round public-coin randomness $coins_i$); this evaluation produces $\vec{p}_{i,k}(\boldsymbol{\sigma}_{i,k}) = \mathbf{q}_{i,k} \in \mathbb{F}^{m_i}$ for all $k \in [\ell]$. The vectors $\mathbf{q}_{i,k}$ are the columns of the query matrix $\mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$.

Preprocessed oracles For each round i of the original algebraic linear IOP, the prover and verifier preprocess $(\mu+1)$-variate degree-d polynomial oracles. For each $k \in [\ell]$, the vector of polynomials $\vec{p}_{i,k} = (p_{i,k,1}, \dots, p_{i,k,m_i}) \in (\mathbb{F}[\mathbf{X}])^{m_i}$ with $\mathbf{X} = (X_1, \dots, X_\mu)$ is encoded as a single polynomial in $\mu + 1$ variables as follows. Introduce a new indeterminate Z, and then define $\tilde{P}_{i,k}(\mathbf{X}, Z) := \sum_{j=1}^{m_i} p_{i,k,j}(\mathbf{X}) Z^j \in \mathbb{F}[\mathbf{X}, Z]$. The prover and verifier establish the oracle $\tilde{P}_{i,k}$, meaning that the verifier queries this oracle on enough points to be reassured that it is correct everywhere.

The transformed IOP The original algebraic linear IOP is modified as follows.

- Wherever the original IOP prover sends an oracle $\boldsymbol{\pi}_i$ of length m_i, the new prover sends a degree $m_i - 1$ univariate polynomial oracle f_{π_i} whose coefficient vector is *the reverse* of $\boldsymbol{\pi}_i$.

- Wherever the original IOP verifier makes ℓ queries within a round to a particular proof oracle $\boldsymbol{\pi}_i$, where queries are defined by query matrix $\mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$, consisting of column query vectors $(\mathbf{q}_{i,1}, ..., \mathbf{q}_{i,\ell})$, the new prover and verifier engage in the following interactive subprotocol for each $k \in [\ell]$ in order to replace the kth linear query $\langle \boldsymbol{\pi}_i, \mathbf{q}_{i,k} \rangle$:

- Verifier: Run the original IOP verifier to get the public coin seed matrix $\mathbf{\Sigma}_i$ and send it to the prover.
- Prover: Derive the query matrix $\mathbf{Q}_i$ from $\mathbf{\Sigma}_i$ using the polynomials $\vec{p}_{i,1}, \ldots, \vec{p}_{i,\ell}$. Send an oracle for the polynomial $F_{i,k}$ whose coefficient vector is $\mathbf{q}_{i,k}$.
- Verifier: Sample uniform random $\beta \xleftarrow{\$} \mathbb{F}$ and query both $F_{i,k}$ and $\tilde{P}_{i,k}$ (the kth preprocessed oracle for round i) at β in order to check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$. If the check fails, abort and output 0.
- Prover: Compute $a_{i,k} = \langle \boldsymbol{\pi}, \mathbf{q}_{i,k} \rangle$ and send $a_{i,k}$ to the verifier.
- The prover and verifier run the inner product Polynomial IOP from Section 2.6.2 on the oracles $F_{i,k}$ and f_{π_i} to convince the verifier that $a_{i,k} = \langle \mathbf{q}_{i,k}, \boldsymbol{\pi}_i \rangle$. If the inner product subprotocol fails the verifier aborts and outputs 0.

If all substeps succeed, then the verifier obtains correct output of each oracle query; in other words, the responses are identical in the new and original IOP. These outputs are passed to the original verifier decision algorithm, which outputs 0 or 1.

Soundness and completeness If the prover is honest then the verifier receives the same exact query-response pairs $(\mathbf{q}_{i,k}, a_{i,k})$ as the original IOP verifier and runs the same decision algorithm, and therefore the protocol inherits the completeness of the original IOP. As for soundness, an adversary who sends a polynomial oracle $F^*_{i,k}$ whose coefficient vector is *not* $\mathbf{q}_{i,k}$, fails with overwhelming likelihood. To see this, note that since $\mathbf{q}_{i,k} = \vec{p}_{i,k}(\boldsymbol{\sigma}_k)$, the check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$ at a random β fails with overwhelming probability by the Schwartz-Zippel lemma. Similarly, an adversary who provides an incorrect $a^*_{i,k} \neq \langle \boldsymbol{\pi}_i, \mathbf{q}_{i,k} \rangle$ fails the inner-product IOP with overwhelming probability. Therefore, if the original IOP soundness error is ϵ then by a union bound the new soundness error is $\epsilon + \mathsf{negl}(\lambda)$. A similar composition argument follows for knowledge extraction.

Round complexity The prover and verifier can first simulate the t-round original IOP on the verifier's public-coin challenges, proceeding as if all queries were answered honestly. Wherever the original IOP prover would send an oracle for the vector $\boldsymbol{\pi}_i$ the prover sends

f_{π_i}. Then, after the verifier has sent its final public coin challenge from the original IOP, there is one more round in which the prover sends all $F_{i,k}$ for the kth query vector in the ith round and all the purported answers $a_{i,k}$ to the kth query in the ith round. The prover and verifier engage in the protocol above to prove that these answers are correct. The inner product subprotocol for each $F_{i,k}$ with f_{π_i} can be done in parallel with the check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$. Therefore, there is only one extra round. $\square$

2.6.3 Compiling Polynomial IOPs

Let $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ be a multivariate polynomial commitment scheme. Given any t-round Polynomial IOP for $\mathcal{R}$ over $\mathbb{F}$, we construct an interactive protocol $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ as follows. For clarity in our explanation, Π consists of t *outer rounds* corresponding to the original IOP rounds and *subrounds* where subprotocols may add additional rounds of interaction between outer rounds.

- Setup: Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
- In any round where the IOP prover sends a (μ, d) polynomial proof oracle $\boldsymbol{\pi} : \mathbb{F}^\mu \to \mathbb{F}$, in the corresponding *outer round* of Π, $\mathcal{P}$ sends the commitment $c_{\boldsymbol{\pi}} \leftarrow \mathsf{Commit}(\mathsf{pp}; \boldsymbol{\pi})$
- In any round where the IOP verifier makes an *evaluation* query $\mathbf{z}$ to a (μ, d) polynomial proof oracle $\boldsymbol{\pi}$, in the corresponding *outer round* of Π, insert an interactive execution of $\mathsf{Eval}(\mathsf{pp}, c_\pi, \mathbf{z}, y, \mu, d; \boldsymbol{\pi})$ between $\mathcal{P}$ and $\mathcal{V}$, where $\boldsymbol{\pi}(\mathbf{z}) = y$.

If $\mathcal{V}$ does not abort in any of these subprotocols, then it receives a simulated IOP transcript of oracle queries and responses. It runs the IOP verifier decision algorithm on this transcript and outputs the result.

Optimization: delayed evaluation As an optimization to reduce round-complexity and enable batching techniques, all invocations of Eval can be delayed until the final round, and heuristically could be run in parallel. Delaying the evaluations until the final round does not affect our analysis. However, our analysis does not consider parallel execution of the Eval subprotocols. We assume the protocol transcript contains an isolated copy of each Eval instance and does not interleave messages or re-use randomness.

Theorem 4. *If the polynomial commitment scheme Γ has witness-extended emulation, and if the t-round Polynomial IOP for $\mathcal{R}$ has negligible knowledge error, then Π is a public-coin interactive argument for $\mathcal{R}$ that has witness-extended emulation. The compilation also preserves HVZK if Γ is hiding and* Eval *is HVZK.*

The full proof is provided in Appendix 2.10. HVZK is shown by a straightforward composition of the simulators for Eval and the original IOP simulator. The emulator E works as follows. Given the IP adversary P', E simulates an IOP adversary P'_O by using the Eval emulator E_{Eval} to extract proof oracles (*i.e.*, polynomials) from any commitment that P' sends and subsequently opens at an evaluation point. We argue that P'_O is successful whenever P' is successful, with negligible loss. (The only events that cause P'_O to fail when P' succeeds is if E_{Eval} fails to extract from a successful Eval or P' succesfully opens a commitment inconsistently with an extracted polynomial). E then runs the IOP knowledge extractor with P'_O to extract a witness for the input.

2.6.4 Concrete Instantiations

We consider examples of Polynomial IOPs to which this compiler can be applied: STARK [23]; Sonic [120] and its improvements PLONK [87] and Marlin [62]; $\mathsf{Spartan}$ [150], and the popular QAP of Gennaro *et al.* [89]. For the purpose of the following discussion, we refer to the complexity of an NP relation $\mathcal{R}$ in various forms:

- $\mathcal{R}$ has *arithmetic complexity* n if the function computing $\mathcal{R}(x, w)$ can be expressed as 2-fan-in arithmetic circuit with a total of n gates.
- $\mathcal{R}$ has *multiplicative complexity* n if the function computing $\mathcal{R}(x, w)$ can be expressed an arithmetic circuit with a total of n multiplication gates, where each multiplication gate has 2 inputs.

- $\mathcal{R}$ has *R1CS complexity*[7] n if the function computing $\mathcal{R}(x, w)$ can be expressed as an R1CS instance (A, B, C, v, w) where $A, B, C \in \mathbb{F}^{m\times(\ell+1)}$, $(v, w) \in \mathbb{F}^{\ell}$, and n is the maximum number of non-zero entries in either A, B, or C.

Theorem 5 provides the main theoretical result of this work, tying together the new DARK polynomial commitment scheme (Theorem 1), the compilation of HVZK Polynomial IOPs into zk-SNARKs with preprocessing using polynomial commitments (Theorem 4), and a concrete univariate Polynomial IOP introduced in Sonic [120] (Theorem 2.6.4) or follow-up works.

Sonic

Sonic is a zk-SNARK system that has a universal trusted setup, which produces a Structured Reference String (SRS) of n group elements that can be used to prove any statement represented as an arithmetic circuit with at most n gates. The SRS can also be updated without re-doing the initial setup, for instance, to enable proving larger circuits, or to increase the distribution of trust. The result in Sonic was not presented using the language of IOPs. Furthermore, the result also relied on a special construction of polynomial commitments (a modification of Kate *et al.* [105]) that forces the prover to commit to a Laurent polynomial with no constant term. Given our generic reduction from coefficient queries to evaluation queries (Section 2.6.2), we re-characterize the main theorem of Sonic as follows:

[120]'s Theorem (Sonic Bivariate). *There exists a 2-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with multiplicative complexity n) that makes 1 query to a bivariate polynomial oracle of degree n on each variable, and 6 queries to degree n univariate polynomial oracles. The preprocessing verifier does $O(n)$ work to check the single bivariate oracle.*

The number of univariate queries increased from the original 3 in Sonic (with special commitments) to 6 with our generic coefficient query technique. If we were to compile the

[7]The arithmetic complexity and R1CS complexity are similar, but vary because the R1CS constraints correspond to the wiring of an arithmetic circuit with unrestricted fan-in.

bivariate query directly using our multivariate commitment scheme this would result in $O(n^2)$ prover time (a bivariate polynomial with degree n on each variable is converted to a univariate polynomial of degree roughly n^2). However, Sonic also provides a way to replace the bivariate polynomial with several degree n univariate polynomials and more rounds of communication.

[120]'s Theorem (Sonic Univariate). *There is a 5-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with multiplicative complexity n) that makes* 39 *queries overall to* 27 *univariate degree* $2n$ *polynomial oracles. The total number of distinct query points is* 12*. The preprocessing verifier does* $O(n)$ *work to check* 12 *of the univariate degree* $2n$ *polynomials.*

The recent proof systems PLONK and Marlin improve on Sonic by constructing a different Polynomial IOP. They achieve the following:

[87]'s Theorem (PLONK). *There is a 3-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with arithmetic complexity n) that makes* 12 *queries overall to* 12 *univariate degree n polynomial oracles. The total number of distinct query points is* 2*. The preprocessing verifier does* $O(n)$ *work to check* 7 *of the univariate degree n polynomials.*

[62]'s Theorem (Marlin). *There exists a 4-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with R1CS complexity n) that makes* 20 *queries at* 3 *distinct query points to* 19 *univariate degree polynomial oracles of maximum degree* $6n$*. The preprocessing verifier does* $O(n)$ *work to check* 9 *univariate degree n polynomials.*

Combining the Sonic Polynomial IOP with the new transparent polynomial compiler of Section 2.5 gives the following result. Similar results are obtained by using PLONK or Marlin instead.

Theorem 5 (**New Transparent zk-SNARK**)**.** *There exists an* $O(\log n)$*-round public-coin interactive argument of knowledge for any NP relation of arithmetic complexity n that has* $O(\log n)$ *communication,* $O(\log n)$ *"online" verification, quasilinear prover time, and a preprocessing step that is verifiable in quasilinear time. The argument of knowledge has* *witness-extended emulation assuming it is instantiated with a group $\mathbb{G}$ for which the Strong RSA Assumption, and the Adaptive Root Assumption hold.*

Proof. We apply the univariate polynomial commitment scheme from Section 2.5 to the 5-round Sonic Univariate Polynomial IOP. Denote this commitment scheme by $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$

The preprocessing requires running Commit on 12 univariate degree n polynomials, which involves a quasilinear number of group operations in the group of unknown order $\mathbb{G}$ determined by Setup. The prover sends a constant number of proof oracles of degree $2n$ to the verifier, which also takes a quasilinear number of group operations. Finally, the 39 queries are replaced with at most 39 invocations of Eval, which adds $O(\log n)$ rounds and has $O(\log n)$ communication. By Theorem 1 (Γ has witness extended emulation) and Theorem 4, the compiled interactive argument has witness-extended emulation. □

2.7 Evaluation

We now evaluate Supersonic, the trustless-setup SNARK built on the Polynomial IOPs underlying Sonic [120], PLONK [87], and Marlin [62], and compiled using our DARK polynomial commitment scheme. As explained in Section 2.5.5, the commitment scheme has several batching properties that can be put to good use here. It is possible to evaluate k polynomials of degree at most d using only 2 group elements and $(k + 1)$ field elements. To take advantage of this we delay the evaluation until the last step of the protocol (see Section 2.6.3). We present the proof size for both the compilation of Sonic, PLONK and Marlin in Table 2.3. We use 1600 bits as the size of class group elements and $\lambda = 120$. The security of 1600 bit class groups is believed to be equivalent to 3048bit RSA groups and have 120 bits of security [48, 33]. This leads to proof sizes of 16.5KB for Sonic, 10.1KB using PLONK and 12.3KB using Marlin for circuits with $n = 2^{20}$ (one million) gates. Using 3048-bit RSA groups the proof sizes becomes 18.4KB for the compilation of PLONK. If 100 bits of security suffice then a 1200 bit class group can be used and the compiled PLONK proofs are 7.8KB for the same setting. In a 2048-bit RSA group this becomes 12.7KB.

The comparison between the Polynomial IOPs is slightly misleading because for Sonic n is the number of multiplication gates whereas for PLONK it is the sum of multiplication and addition gates. For Marlin it is the number of non-zero entries in the R1CS description of the circuit. A more careful analysis is therefore necessary, but this shows that there are Polynomial IOPs that can be compiled using the DARK polynomial commitment scheme to SNARKs of roughly 10 kilobytes in size. These numbers stand in contrast to STARKs which achieve proofs of 600KB for computation of similar complexity [23]. We compare $\mathsf{Supersonic}$ to different other proof systems in Table 2.4. $\mathsf{Supersonic}$ is the only proof system with efficient verifier time, small proof sizes that does not require a trusted setup.

Polynomial IOP	Polynomials	Eval points	|SNARK|	concrete size
Sonic [120]	12 in pp + 15	12	$(15+2\log_2(n))\mathbb{G}$ $+(12+13\log_2(n))\mathbb{Z}_p$	15.3 KB
PLONK [87]	7 in pp + 7	2	$(7+2\log_2(n))\mathbb{G}$ $+\ (2+3\log_2(n))\mathbb{Z}_p$	10.1 KB
Marlin [62]	9 in pp + 10	3	$(10+2\log_2(6n))\mathbb{G}$ $+\ (3+4\log_2(6n))\mathbb{Z}_p$	12.3 KB

Table 2.3: Proof size for $\mathsf{Supersonic}$. Column 2 says how many polynomials are committed to in the SRS (offline oracles) and how many are sent by the prover (online oracles). Column 3 states the number of distinct evaluation points. The proof size calculation uses $|\mathbb{Z}_p| = 120$ and $|\mathbb{G}| = 1600$ for $n = 2^{20}$ gates.

Prover and Verifier cost We use the notation $O_\lambda(\cdot)$ to denote asymptotic complexity for a fixed security parameter λ, *i.e.* how the prover and verifier costs scale as a function of variables other than λ. The main cost for the $\mathsf{Supersonic}$ prover consists of computing the commitments to the polynomial oracles and producing the single combined Eval proof. This proof requires calculating the commitments to the polynomials $f_L(q)$ and $f_R(q)$ in each round and performing the $\mathsf{PoE}(\mathsf{C}_R, \mathsf{C} - \mathsf{C}_L, q^{d'+1})$. Using precomputation, *i.e.*, computing G^{q^i} for all i and using multi-scalar multiplication, the commitments can be computed in $O_\lambda(\frac{d}{\log(d)})$ group operations. The same techniques can be used to reduce the number of group operations for the PoEs to $O_\lambda(d)$. The total number of group operations is therefore linear in the maximum degree of the polynomial oracles and the number of online oracles. Interestingly, the number of offline oracles hardly impacts the prover time and proof size.

The verifier time is dominated by the group operations for scalar multiplications in various places in the single combined Eval protocol. It consists of 3 λ-bit scalar multiplications in each round: 1 for combining C_L and C_R and two for verifying the PoE. In the final round, the verifier does another $\lambda \log_2(d+1)$-bit scalar multiplication to open the commitment but this could also be outsourced to the prover using yet another PoE. The total verifier time, therefore, consists of roughly a scalar multiplication with $3\lambda \log_2(d+1)$ group operations. Using 10μs per group operation[8], this gives us for $\lambda = 120$ and $n = 2^{20}$ a verification time of around 72ms.

Scheme	Transp.	$\lvert\mathsf{pp}\rvert$	Prover	Verifier	$\lvert\pi\rvert$	$n = 2^{20}$
Supersonic	yes	$O(1)$	$O(n\log(n))$ MUL	$3\log(n)$ MUL	$2\log(n)\ \mathbb{G}_U$	10.1KB
PLONK [87]	no	$2n\ \mathbb{G}_B$	$O(n)$ MUL	1 Pairing	$O(1)\ \mathbb{G}_B$	720b
Groth16 [96]	no	$2n\ \mathbb{G}_B$	$O(n)$ MUL	1 Pairing	$O(1)\ \mathbb{G}_B$	192b
BP [51]	yes	$2n\ \mathbb{G}_P$	$O(n)$ MUL	$O(n)$ MUL	$2\log(n)\ \mathbb{G}_P$	1.7KB
STARK	yes	$O(1)$	$O(\lambda T)$ H	$O(\lambda\log^2(T))$ H	$O(\lambda\log^2(T))$ H	600 KB
Virgo[169]	yes	$O(1)$	$O(\lambda n)$ H	$O(\lambda\log^2(n))$ H	$O(\lambda\log^2(n))$ H	271 KB

Table 2.4: Comparison table between different succinct arguments. In column order, we compare by transparent setup, CRS size, prover and verifier time, asymptotic proof size, and concrete proof for an NP relation with arithmetic complexity 2^{20}. Even when precise factors are given the numbers should be seen as estimates. For example, we chose to not display smaller order terms. The symbol $\mathbb{G}_U$ denotes an element in a group of unknown order, $\mathbb{G}_B$ one in a group with a bilinear map (pairing), $\mathbb{G}_P$ one in a prime order group with known order. Furthermore, MUL refers to scalar multiplication of λ-bit numbers in these groups, and H is either the size of a hash output or the time it takes to compute a hash. The prover time for the group based schemes can be brought down by a log factor when using multi-scalar-multiplication techniques.

2.8 Security analysis and almost-special soundness

Although DARK does not satisfy *special soundness*, it does satisfy a property that we will call *almost-special-soundness*, which turns out to be sufficient (i.e., using our new forking

[8]The estimate comes from the recent Chia Inc. class group implementation competition. The competition used a larger 2048bit discriminant but only performed repeated squaring. `https://github.com/Chia-Network/vdfcontest2results`

lemma we can prove that such protocols are knowledge sound). Almost-special-soundness is a weaker property than special soundness, as all protocols that satisfy special soundness also satisfy almost-special-soundness.

2.8.1 Integer Polynomials

If f is a multivariate polynomial, then $||f||_\infty$ denotes the maximum over the absolute values of all coefficients of f.

Lemma 6 (Evaluation Bound)**.** *For any μ-linear integer polynomial f and $m \geq 2$:*

$$\mathbb{P}_{\mathbf{x} \leftarrow [0,m)^\mu}[|f(\mathbf{x})| \leq \frac{1}{m^\mu} \cdot ||f||_\infty] \leq \frac{3\mu}{m}$$

Proof. Let $f^{(0)} := f$. Given a vector $\mathbf{x} = (x_1, .., x_\mu)$, for each $j \in [1, \mu]$ define $f_{\mathbf{x}}^{(j)}$ to be the $\mu - j$-variate partial evaluation $f_{\mathbf{x}}^{(j)} := f(x_1, ..., x_j, X_{j+1}, ..., X_\mu)$. Then we can rewrite the lemma statement as:

$$P_{\mathbf{x} \leftarrow [0,m)^\mu}[||f_{\mathbf{x}}^{(\mu)}||_\infty \leq \frac{1}{m^\mu} \cdot ||f^{(0)}||_\infty] \leq \frac{3\mu}{m}$$

We will bound the probability for random $\mathbf{x}$ that there exists any j for which $||f_{\mathbf{x}}^{(j)}|| < \frac{1}{m} \cdot ||f_{\mathbf{x}}^{(j-1)}||$. If no such j exists, then $||f^{(\mu)}|| \geq \frac{1}{m^\mu} \cdot ||f^{(0)}||$.

For any j, we can write $f_{\mathbf{x}}^{(j)} = g(X_{j+1}, ..., X_\mu) + x_j \cdot h(X_{j+1}, ..., X_\mu)$ where g, h are $\mu - j$ variate multilinear integer polynomials and $||f_{\mathbf{x}}^{(j-1)}|| = \max(||g||, ||h||)$ because the coefficients of g and h are a partition of the coefficients of $f_{\mathbf{x}}^{(j-1)}$. Suppose now that $||f_{\mathbf{x}}^{(j)}|| < \frac{1}{m} \cdot ||f_{\mathbf{x}}^{(j-1)}||$, i.e. that $||g + x_j \cdot h|| < \frac{1}{m} \cdot \max(||g||, ||h||)$ and consider two cases:

Case 1: $||h|| = \max(||g||, ||h||)$. For any integer $\Delta \neq 0$, using the triangle inequality:

$$||g + (x_j + \Delta)h|| = ||g + x_j h + \Delta h|| \geq ||\Delta h|| - ||g + x_j h|| > (1 - \frac{1}{m}) \cdot ||h|| \geq \frac{1}{m} \cdot ||h||$$

The last part of the inequality holds because $1 - \frac{1}{m} \geq \frac{1}{m}$ for any $m \geq 2$.

Case 2: $||g|| = \max(||g||, ||h||)$. Using the triangle inequality,

$$\frac{1}{m}||g|| > ||g + x_j \cdot h|| \geq ||g|| - ||x_j \cdot h||$$

This implies, for $m \geq 2$, that $||h|| > \frac{1}{m} \cdot ||g||$ because:

$$||x_j \cdot h|| > (1 - \frac{1}{m}) \cdot ||g|| \implies ||h|| > \frac{m-1}{x_j \cdot m} \cdot ||g|| \geq \frac{1}{m}||g||$$

The last step uses that $x_j \in [1, m)$. For $x_j = 0$, $||g + x_j h|| = ||g||$. Finally, for any integer Δ, by the triangle inequality:

$$||g + (x_j + \Delta) \cdot h|| \geq ||\Delta h|| - ||g + x_j \cdot h|| > \frac{|\Delta|}{m} \cdot ||g|| - \frac{1}{m} \cdot ||g|| = \frac{|\Delta| - 1}{m} \cdot ||g||$$

When $|\Delta| \geq 2$ this implies that $||g + (x_j + \Delta) \cdot h|| > \frac{1}{m} \cdot ||g||$.

In both cases, we conclude that for any choice of $(x_1, ..., x_{j-1})$ for the first $j-1$ components of the random $\mathbf{x}$, which define g and h, there are at most three choices of x_j such that the event $||f_{\mathbf{x}}^{(j)}|| < \frac{1}{m} \cdot ||f_{\mathbf{x}}^{(j-1)}||$ holds true (i.e., if true for x_j, then it is also true for at most $x_j + 1$ and $x_j - 1$). Thus this event occurs with probability at most $\frac{3}{m}$. Finally, by a union bound over j, the probability this event occurs for some index j is at most $\frac{3\mu}{m}$.

□

The next lemma below (Lemma 7) is proven as Theorem 9 in Chapter 3.

Lemma 7 (Multilinear Composite Schwartz-Zippel)**.** *For all $m \geq 2$, any μ-linear integer polynomial f, and $N \in \mathbb{Z}$ coprime to f, if either $\mu = 1$ and $\log_2 N \geq \lambda$ or $\mu \geq 2$ and $\log_2 N \geq 8\mu^2 + \log_2(2\mu) \cdot \lambda$ then:*

$$\mathbb{P}_{x \leftarrow [0,m)^\mu}[f(x) \equiv 0 \bmod N] \leq \frac{1}{2^\lambda} + \frac{\mu}{m}$$

Chapter 3 also provides an algorithm for computing tighter values for the MCSZ for concrete parameters for μ and λ. We present a range of values for different μ and $\lambda = 120$.

The following lemma is a consequence of Theorem 10 in Chapter 3 and the algorithm discussed in Section 3.5.2.

Lemma 8 (Concrete MCSZ for 120-bit security)**.**

$$\mathbb{P}_{x \leftarrow [0,m)^\mu}[f(x) \equiv 0 \bmod N] \leq 2^{-120} + \frac{\mu}{m}$$

μ	$\lambda = 120$	μ	$\lambda = 120$	μ	$\lambda = 120$
1	*120*	*11*	*301*	*21*	*429*
2	*156*	*12*	*315*	*22*	*437*
3	*175*	*13*	*331*	*23*	*448*
4	*197*	*14*	*344*	*24*	*464*
5	*212*	*15*	*354*	*25*	*472*
6	*234*	*16*	*366*	*26*	*481*
7	*244*	*17*	*381*	*27*	*492*
8	*260*	*18*	*391*	*28*	*506*
9	*277*	*19*	*407*	*29*	*516*
10	*289*	*20*	*416*	*30*	*527*

Fact 2. *Let $q \in \mathbb{Z}$ be any positive integer. For any integer $E \in \mathbb{Z}$ such that $|E| \leq \frac{q^{d+2}-q}{2(q-1)}$ there exists a unique degree d integer polynomial $f \in \mathbb{Z}[X]$ with $||f||_\infty \leq q/2$ such that $f(q) = E$.*

We extend the integer encoding from Section 2.5.2 to rational multi-linear polynomials.

Lemma 9 (Rational Encoding of multi-linear polynomials)**.** *Let $q \in \mathbb{Z}$ be any positive integer. Let $\vec{q} = [q^{2^{i-1}}]_{i=1}^{\mu} \in \mathbb{Z}^\mu$. Consider any $\beta_d, \beta_n \in \mathbb{N}$ such that $\beta_d \cdot \beta_n \leq \frac{q}{2}$. Let $Z = \{z \in \mathbb{Z} : |z| \leq \beta_d\}$, let $\mathcal{F} = \{f \in \mathbb{Z}[X_1, \dots, X_\mu] : ||f||_\infty \leq \beta_n\}$ be a μ-linear polynomial, and let $\mathcal{H} = \{f/z \in \mathbb{Q}[X_1, \dots X_\mu] : f \in \mathcal{F} \wedge z \in Z\}$. Then for any $h_1, h_2 \in \mathcal{H}$, if $h_1(\vec{q}) = h_2(\vec{q})$ then $h_1 = h_2$.*

Proof. Let $h_1 = \frac{f_1}{z_1}$ and $h_2 = \frac{f_2}{z_2}$. If $h_1(\vec{q}) = h_2(\vec{q})$ then $z_1 f_2(\vec{q}) = z_2 f_1(\vec{q})$. Since $||z_2 \cdot f_1||_\infty \leq \beta_d \cdot \beta_n \leq \frac{q}{2}$ and likewise $||z_1 \cdot f_2||_\infty \leq \frac{q}{2}$. Note that there exist a unique univariate degree $2^\mu - 1$ polynomial $\hat{f}_1$ that has the same coefficients as f_1 such that for all q $f_1(\vec{q}) = \hat{f}_1(q)$. Let f_2 be the univariate degree $2^\mu - 1$ polynomial with the same coefficients as $\hat{f}_2$. It then follows from Fact 2 that if $z_1 f_2(\vec{q}) = z_1 \hat{f}_2(q) = z_2 \hat{f}_1(q) = z_2 f_1(\vec{q})$ then $z_1 f_2 = z_2 f_1$, or equivalently, $h_1 = h_2$. □

2.8.2 DARK commitments

We restate the DARK commitment scheme as a commitment scheme to μ-linear polynomials with bounded rational coefficients. If f is a μ-linear polynomial then it can represent a degree $2^{\mu-1}$ univariate polynomial $\hat{f}$ with the same coefficients, as $\hat{f}(z) = f(z, z^2, \ldots, z^{2^{\mu-1}})$. While the honest prover will commit to the integer representation of a polynomial defined over a prime field, this representation is useful in the security proof. The extractor will extract bounded rational polynomials instead of integer ones. Fortunately, we can show that given sufficiently large parameters, the commitment scheme is still binding.

Given the security parameter λ, the commitment scheme setup selects a group $\mathbb{G}$ for which the random order assumption holds (with λ-bit security) and a random generator $\mathsf{G} \in \mathbb{G}$. A parameter $q \in \mathbb{N}$ determines a commitment message space $\mathcal{M} = \{f(X_1, \ldots, X_\mu) : ||f||_\infty \leq q/2 \wedge \forall i \deg_{X_i}(f_i) \leq 1\}$. The commitment to $f(X_1, \ldots, X_\mu) \in \mathcal{M}$ is $f(\vec{q}) \cdot \mathsf{G}$ for $\vec{q} = (q, q^2, \ldots, q^{2^{\mu-1}})$. Commitments are binding over $\mathcal{M}$ because if $f(\vec{q}) \cdot \mathsf{G} = f'(\vec{q}) \cdot \mathsf{G}$ then $(f'(\vec{q}) - f(\vec{q}))\mathsf{G} = 0$. This breaks the order assumption for $\mathbb{G}$ unless $f'(\vec{q}) = f(\vec{q})$, in which case $f' = f$ by Fact 2.

If we expand the valid openings of the commitment scheme to include rational polynomials of the form f/z, where $f \in \mathbb{Z}[X]$, $z \in \mathbb{Z}$, so that (f, D) is an opening of C to f/z iff $z \cdot \mathsf{C} = f(\vec{q}) \cdot \mathsf{G}$, then the scheme is binding over the message space $\mathcal{M}(\beta_n, \beta_d) = \{f(X)/z : f \in \mathbb{Z}[X], z \in \mathbb{Z}, \gcd(f, z) = 1, ||f||_\infty \leq \beta_n, |z| \leq \beta_d\}$ so long as $\beta_n \cdot \beta_d \leq \frac{q}{2}$. Openings to rationals can also be described as a *relaxed opening*, whereby C is opened to f by opening $z \cdot \mathsf{C}$ to $z \cdot f$ for $z \in \mathbb{Z}$.

Lemma 10 (DARK commitment Security)**.** *The commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Vf})$ *to integer* μ*-linear polynomials:*

- $\mathsf{Setup}(\lambda, \beta_n \in \mathbb{Z}, \beta_d \in \mathbb{Z})$*: sample* $\mathbb{G} \leftarrow GGen(\lambda), \mathsf{G} \leftarrow \mathbb{G}$*, return* $\mathsf{pp} := (\mathbb{G}, \mathsf{G}, q = 2\beta_n\beta_d)$
- $\mathsf{Commit}(\mathsf{pp}, f \in \mathcal{M}(\beta_n, \beta_d))$ *: return* $f(\vec{q}) \cdot \mathsf{G}$ *for* $\vec{q} = (q, q^2, \dots, q^{\mu-1}) \in \mathbb{Z}^\mu$
- $\mathsf{Open}(\mathsf{pp}, C \in \mathbb{G}, h = f/N \in \mathcal{M}(\beta_n, \beta_d))$*: return* $\mathbf{1}[N \cdot \mathsf{C} = f(\vec{q}) \cdot \mathsf{G}]$

is binding over the rational polynomial set $\mathcal{M}(\beta_n, \beta_d)$ *under the random order assumption (1), i.e. for any polynomial time adversary* $\mathcal{A}$

$$P\left[\begin{array}{c|c} \mathsf{Open}(\mathsf{pp}, C, h) = \mathsf{Open}(\mathsf{pp}, C, h') = 1 & (h, h', C) \in \mathcal{M}(\beta_n, \beta_d)^2 \times \mathbb{G} \leftarrow \mathcal{A}(\mathsf{pp}) \\ h \neq h' & \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda, \beta_n, \beta_d) \end{array}\right] < \mathsf{negl}(\lambda)$$

Proof. To see why this is binding over $\mathcal{M}(\beta_n, \beta_h)$, suppose that both $h = \frac{f}{N}$ and $h' = \frac{f'}{N'}$ are valid openings of a commitment C to distinct rational polynomials $h, h' \in \mathcal{M}(\beta_n, \beta_d)$. This implies that $N \cdot \mathsf{C} = f(\vec{q}) \cdot \mathsf{G}$ and $z' \cdot \mathsf{C} = f'(\vec{q}) \cdot \mathsf{G}$. Hence, $v = z \cdot f'(\vec{q}) - z' \cdot f(\vec{q})$ is a multiple of the order of G, i.e. $v \cdot \mathsf{G} = 0$. Moreover, by Lemma 9, $h(\vec{q}) \neq h'(\vec{q})$, which implies $v \neq 0$. This breaks the random order assumption for $\mathbb{G}$. It is important to note that the openings (f, z) and (f', z') need not be co-prime elements of bounded norm; it suffices that $h = f/z$ and $h' = f'/z'$ have bounded norm numerators and denominators in reduced fraction form, i.e. $h, h' \in \mathcal{M}(\beta_n, \beta_d)$. □

2.8.3 IP Transcript Trees

Let (P, V) be a μ-round public coin interactive protocol. A μ-round public-coin protocol (P, V) consists of μ-rounds of messages between the prover and verifier, where in each round the prover sends a message to the verifier and the verifier responds with $x \leftarrow \mathcal{X}$ sampled uniformly from the challenge space $\mathcal{X}$. At the end of the protocol, the verifier outputs either *accept* or *reject*. By convention, the protocol starts with the prover's first message and ends with the prover's last message. A *transcript* thus contains $\mu + 1$ prover messages and μ

challenges. We will denote transcripts by a $\mu \times 2$ matrix A such that $A(0,0)$ is the protocol input x, $A(0,1)$ is the prover's first message, and for all $i \geq 1$, $A(i,0)$ is the verifier's *ith* round challenge and $A(i,1)$ is the prover's *ith* round response. We restrict our attention to protocols in which the verifier's decision is a deterministic function D_V of the transcript, which is true of the DARK protocol, but is also without loss of generality. An *accepting transcript* is an array A such that $D_V(A) = accept$.

A k-ary *transcript tree* for (P,V) is a labelling of a μ-depth k-ary tree such that the labels on every root-to-leaf path forms an accepting (P,V) transcript. It will be convenient to order the nodes of the tree according to a depth-first reverse topological sort (aka post-order tree traversal). This is a topological sorting of the tree with directed edges flowing from leaves to root which places left subtrees before right subtrees. This ordering associates each node with an index in $[1,N]$ where $N = \mathsf{size}(\mu,k) = \frac{k^{\mu+1}-1}{k-1}$.

A *post-order labelling* of the tree is a function $L : [1,N] \to \mathcal{X} \times \mathcal{M}$ where $\mathcal{X}$ is the verifier's challenge set and $\mathcal{M}$ is the space of prover messages. We can think of the first component (i.e., the verifier challenge) as a label on the node's incoming edge and the second component (i.e., prover's response) as a label on the node itself. The root has no incoming edge, but the root label's first component is the protocol input. For any root-to-leaf path of nodes with indices $\{v_0, ..., v_\mu\}$ the labelling L defines the matrix A such that $L(v_i) = (A(i,0), A(i,1))$ and A is an accepting transcript. Given a label $L(v)$ for $v < N$ (non-roots) we will use the notation $L(v)_0$ to denote the first component of the label containing the verifier's challenge and $L(v)_1$ the second component containing the prover's response. Finally, we define a k-ary *forking transcript tree* to be a k-ary transcript tree in which the challenge labels on all edges sharing a common parent are distinct.

We may refer to the *level* of a node in the tree. The root of a tree is always at *level 0* and the leaves of a depth μ tree are at *level μ*. The *height* of node at level ℓ within a μ-depth tree is $\mu - \ell$. For each $v \in [1,N]$ let v^* denote the largest index $v^* < v$ such that the node at index v^* does not belong to the subtree extending from v. Note that for nodes on the leftmost path of the tree v^* does not exist so we denote it by $\perp$. For each $v \in [1,N]$ let $L_v : [1,v] \to \mathcal{X} \times \mathcal{M}$ denote the restriction of L to the subset $[1,v^*]$. Similarly, for any

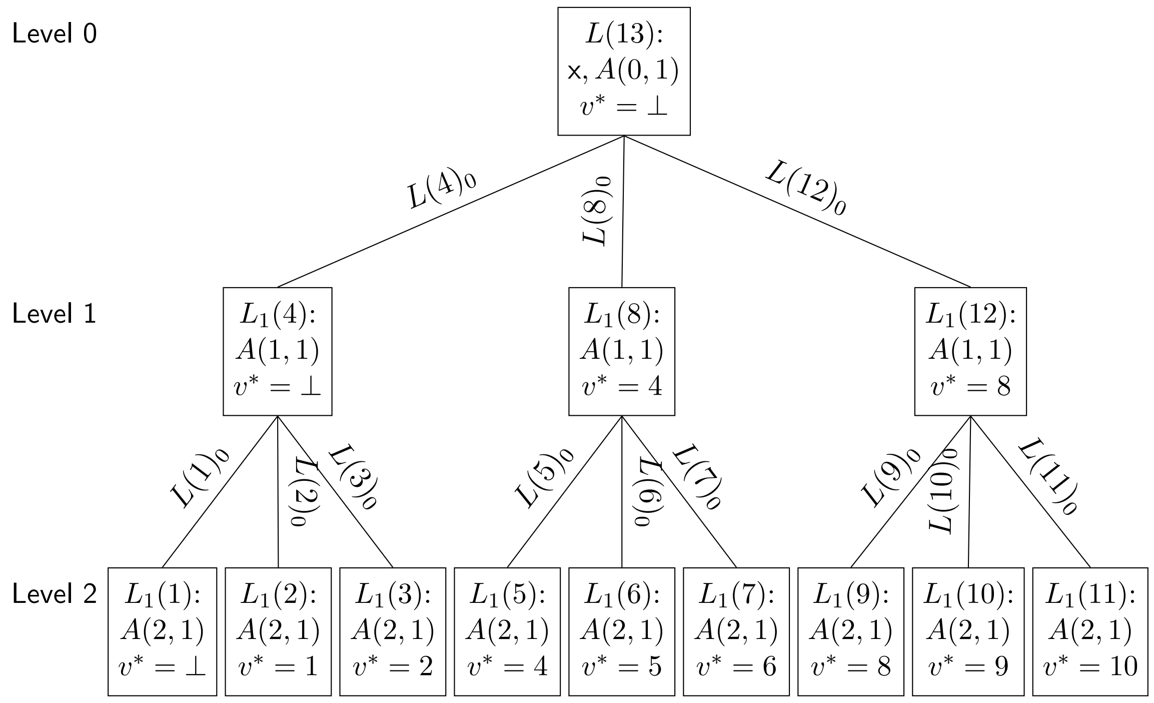
Level 0
Level 1
Level 2
$L(13)$: x, $A(0,1)$ $v^* = \perp$
$L(4)_0$
$L(8)_0$
$L(12)_0$
$L_1(4)$: $A(1,1)$ $v^* = \perp$
$L_1(8)$: $A(1,1)$ $v^* = 4$
$L_1(12)$: $A(1,1)$ $v^* = 8$
$L(1)_0$ $L(2)_0$ $L(3)_0$ $L(5)_0$ $L(6)_0$ $L(7)_0$ $L(9)_0$ $L(10)_0$ $L(11)_0$
$L_1(1)$: $A(2,1)$ $v^* = \perp$
$L_1(2)$: $A(2,1)$ $v^* = 1$
$L_1(3)$: $A(2,1)$ $v^* = 2$
$L_1(5)$: $A(2,1)$ $v^* = 4$
$L_1(6)$: $A(2,1)$ $v^* = 5$
$L_1(7)$: $A(2,1)$ $v^* = 6$
$L_1(9)$: $A(2,1)$ $v^* = 8$
$L_1(10)$: $A(2,1)$ $v^* = 9$
$L_1(11)$: $A(2,1)$ $v^* = 10$

Figure 2.1: IP transcript tree for $\mu = k = 3$. Nodes and edges are labeled using post-order labeling. We also indicate v^* for every node.

$S \subseteq [1, N]$ let $L_S : S \to \mathcal{X} \times \mathcal{M}$ denote the restriction of the labelling L to the subset of node indices S. For any $v \in [1, N]$ let $S_v \subseteq [1, N]$ denote the indices of all nodes in the subtree rooted at node v. L_{S_v} thus denotes the labelling of the subtree S_v. Note that L_{v^*} is not the same as $L_{S_{v^*}}$.

2.8.4 Path Predicate Forking Lemma

The standard forking lemma for μ-round public coin interactive protocols characterizes the efficiency of generating a k-ary μ-depth transcript tree for which the challenges labeling the children within the tree *fork*, i.e. are distinct. More precisely, the forking lemma says that given any adversarial prover $\mathcal{A}$ that may deviate from the honest protocol but causes the verifier to accept with probability ϵ, there is a tree generation algorithm that has only black-box access to $\mathcal{A}$, runs in time $t \in O(\frac{\lambda}{\epsilon} \cdot k^\mu \cdot (\mu + t_V))$, where t_V is the running time of the verifier's decision algorithm, and succeeds with probability $1 - t \cdot \mathsf{negl}(\lambda)$ in producing a transcript tree with the forking property.

Our *path predicate forking lemma* generalizes the property of the transcript tree that can be generated by considering arbitrary predicates on partial labelings of the tree. In the standard forking lemma, the predicate would simply be that new challenges are distinct from previous challenges. The lemma considers predicates for each node $v \in [1, N]$ at level ℓ_v of the form $\pi_v : (\mathcal{X} \times \mathcal{M})^{[1,v^*]} \times \mathcal{X}^{\mu-\ell_v} \to \{0, 1\}$, i.e. each predicate π_v takes as input a labelling function L_{v^*} for the partial set of nodes $[1, v^*]$ and a vector of challenges $\mathbf{x} \in \mathcal{X}^{\mu-\ell_v}$. The vector of challenges will represent *the leftmost* path down the tree starting from v, which by definition is independent of the partial labeling L_{v^*}. We denote the indices of the leftmost path from v to the leaves as lpath_v and the challenge labels along this path assigned by L as $L(\mathsf{lpath}_v)_0$. For example in Figure 2.1 the predicate π_8 for node 8 would take as input the subtree spanned by 4 and the challenge $L(5)_0$. The lemma says that if $\pi_v(L_{v^*}, \mathbf{x}) = 1$ with overwhelming probability $1 - \mathsf{negl}(\lambda)$ for any post-order labeling $L : [1, N] \to \mathcal{X} \times \mathcal{M}$ of the k-ary μ-depth tree, any node v in the tree, and $\mathbf{x}$ sampled randomly, then the transcript generation algorithm produces a transcript tree represented by some post-order labeling L for which $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 1$ for all v in the tree. In fact, the lemma is even more general as it has a weaker requirement that $\pi_v(L^*_v, \mathbf{x}) = 1$ with overwhelming probability conditioned on $\pi_u(L^*_u, L(\mathsf{lpath}_u)_0) = 1$ for all $u \leq v^*$. The standard forking lemma is a special case where π_v checks that the challenge label on v is distinct from the challenge labels on any of its left siblings. The challenge label $L(v)_0$ on v is the first component of $L(\mathsf{lpath}_v)_0$ and the challenge labels on the left sibling(s) of v, assuming v is not the first child, are included in L_{v^*}.

Proof Overview We will begin with a high level overview of the proof. The algorithm is exactly the same as the recursive tree generation algorithm for the standard forking lemma. The difference is only in the analysis. The standard forking lemma considers predicates $\pi_v(L_{v^*}, x)$ that are functions only of the challenges assigned by L_{v^*} to left sibling nodes of v and a single (fresh) $x \in \mathcal{X}$ rather than a vector, and are independently true with overwhelming probability.

Just as in the standard forking lemma, the analysis is a simple union bound. First, the tree generation algorithm is transformed to a Monte Carlo algorithm that runs for $t \in \mathsf{poly}(\lambda)$ steps and succeeds with overwhelming probability. The standard forking lemma is based on the observation that a t-step algorithm makes at most t samples from $\mathcal{X}$ and thus the predicates hold true for all sampled challenges with probability at least $1-t\cdot\mathsf{negl}(\lambda)$. In our case, the analysis is very similar. Let L denote the labelling returned by the Monte Carlo tree generation algorithm. We begin with the observation that this tree generation algorithm constructs the labelling in depth-first post-order. In particular, when the transcript tree generation algorithm visits a node v at heigh h_v it has already derived a partial labelling L_{v^*}. It samples a random vector $\mathbf{x} \in \mathcal{X}^{h_v}$ and attempts to derive a valid transcript for lpath_v using this challenge vector $\mathbf{x}$. If it succeeds then it sets $L(\mathsf{lpath}_v)_0 = \mathbf{x}$, otherwise the entire vector $\mathbf{x}$ is discarded and it tries again starting from v. Suppose there exists some v such that $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 0$ and let v be the lowest index node with this property. This would imply that there occurred an event where the algorithm had already constructed L_{v^*} satisfying $\pi_u(L_{u^*}, L(\mathsf{lpath}_u)_0) = 1$ for all $u \leq v^*$ and then sampled $\mathbf{x} \leftarrow \mathcal{X}^{h_v}$, setting $L(\mathsf{lpath}_v)_0 = \mathbf{x}$, such that $\pi_v(L_{v^*}, \mathbf{x}) = 0$. However, by hypothesis this event occurs with probability $\mathsf{negl}(\lambda)$ over random $\mathbf{x}$. Since the algorithm runs for only $t \in \mathsf{poly}(\lambda)$ steps, an event of this kind occurs with probability at most $t \cdot \mathsf{negl}(\lambda)$.

Thus, we obtain a Monte Carlo algorithm that returns a transcript tree where all the predicates are satisfied with overwhelming probability.

Lemma 11 (Path Predicate Forking Lemma)**.** *Let (P, V) be a μ-round public-coin protocol with prover message space $\mathcal{M}$ and verifier challenge space $\mathcal{X}$. For each node $v \in [1, N]$ of a μ-depth k-ary balanced tree on $N = \mathit{size}(\mu, k)$ nodes, let h_v denote the height of v. Let $\{\pi_v : v \in [1, N]\}$ denote a set of predicates, where $\pi_v(L_{v^*}, \mathbf{x})$ is a function of the partial labelling L_{v^*} and challenge vector $\mathbf{x} \in \mathcal{X}^{h_v}$, with the property that for any post-order labelling function $L : [1, N] \to \mathcal{X} \times \mathcal{M}$ and any $v \in [1, N]$:*

$$Pr_{\mathbf{x}\leftarrow\mathcal{X}^{h_v}}[\pi_v(L_{v^*}, \mathbf{x}) = 1 \mid \forall_{u\leq v^*}\pi_u(L_{u^*}, L(\mathit{lpath}_u)_0) = 1] \geq 1 - \delta$$

Let t_V denote the worst-case running time of the verifier's decision algorithm D_V. There is an algorithm $\mathsf{Tree}^{\mathcal{A}}(\mathsf{z})$ that, given a security parameter $\lambda \in \mathbb{N}$ and oracle access to an adversarial prover $\mathcal{A}$ that causes V to accept with probability ϵ on public input z, runs in time at most $t = 2\lambda \cdot \frac{k^\mu}{\epsilon} \cdot (\mu + t_V)$ and with probability at least $1 - t \cdot \delta - 2^{-\lambda}$ outputs a k-ary transcript tree with post-order labeling $L : [1, N] \to \mathcal{X} \times \mathcal{M}$ such that $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 1$ for all $v \in [1, N]$.

Proof. We will first describe a Las Vegas tree-finding algorithm that runs in expected polynomial time as we can then transform it to a Monte Carlo algorithm with a finite runtime and overwhelming success probability.

Tree finding algorithm The tree-finding algorithm $\mathsf{Tree}(k, \mathsf{z})$ begins by sampling a random tape σ for the adversary. Let $A(\sigma)$ denote the *deterministic*[9] adversary with fixed random tape σ. For all $i \in [0, \mu]$ define $T_i(\sigma, k, \mathsf{z}, x_1, ..., x_i)$ as follows:

Algorithm $T_i(\sigma, k, \mathsf{z}, x_1, ..., x_i)$:

- **If $i = \mu$:** Simulate the protocol with $\mathcal{A}(\sigma)$ as the prover and fixing the verifier's challenges μ ordered challenges to the values $x_1, ..., x_\mu$. If the verifier outputs 1 during this simulation then return the protocol transcript tr, and otherwise return fail.
- **Else if $0 \leq i < \mu$:** Sample $x_{i+1} \leftarrow \mathcal{X}$ and run $T_i(\sigma, k, y, x_1, ..., x_{i+1})$. This either returns fail or a transcript tree denoted tree. If it returns fail, then output fail. Otherwise, save the pair (x_{i+1}, tree). If $i < \mu - 1$ then tree is a tree of accepting transcripts that share a common prefix for the first $i + 1$ rounds, which includes the challenges $x_1, ..., x_{i+1}$. If $i + 1 = \mu$ then tree is a single accepting transcript. Repeat this process as many times as needed, each time sampling a fresh x'_{i+1}, running $T_i(\sigma, y, x_1, ..., x'_{i+1})$, ignoring the runs that fail, saving the succesful challenge/tree pairs until k pairs have been recorded. Together the transcripts in all k recorded trees form one larger tree

[9] Any probabilistic adversarial algorithm can be represented by a deterministic algorithm that takes as input a random tape.

of accepting transcripts that share a commmon prefix $\mathsf{tr}_{\mathsf{pre}}$ for the first i rounds of messages with fixed challenges $x_1, ..., x_i$.

$\mathsf{Tree}(k, \mathsf{z})$ repeatedly samples σ and runs $T_0(\sigma, k, \mathsf{z})$ until it outputs a tree of accepting transcripts.

We now analyze the expected runtime of $\mathsf{Tree}(k, \mathsf{z})$ and success probability of returning an k-ary tree of accepting transcripts given that $\mathcal{A}$ succeeds with probability ϵ. $T_0(\sigma, k, \mathsf{z})$ returns fail iff the first iteration of each subroutine T_i returns fail for $i = 1$ to μ. The probability this happens is equal to the probability that $T_\mu(\sigma, y, x_1, ..., x_\mu)$ outputs fail for a uniformly distributed challenge tuple $(x_1, ..., x_\mu)$. This is equal to the failure probability of $\mathcal{A}(\sigma)$, i.e. $1 - \epsilon$. Thus, $\mathsf{Tree}(k, \mathsf{z})$ calls T_0 in expectation $1/\epsilon$ times. Letting t_0 be a random variable for the runtime of $T_0(\sigma, \mathsf{z})$ over random σ, the expected runtime of $\mathsf{Tree}(k, \mathsf{z})$ is t_0/ϵ.

It remains to analyze the expected runtime $\mathbb{E}[t_0]$ of $T_0(\sigma, \mathsf{z})$. Each call to $T_i(\sigma, k, \mathsf{z}, x_1, ..., x_i)$ for $i \in [1, \mu]$ that occurs in the execution trace of $T_0(\sigma, k, \mathsf{z})$ is on i.i.d. uniformly distributed challenges $x_1, ..., x_i$. Let t_i be a random variable denoting the runtime of $T_i(\sigma, k, \mathsf{z}, x_1, ..., x_i)$ over a uniformly distributed challenge prefix $\mathbf{x}_i = (x_1, ..., x_i)$ and uniformly distributed σ. We omit the time to sample a random challenge from the runtime analysis as this will only affect the runtime up to a constant factor. Since $T_\mu(\sigma, k, \mathsf{z}, \mathbf{x}_\mu)$ makes μ calls to the oracle $\mathcal{A}$ and one call to the verifier's decision algorithm D_V its runtime is at most $\mu + t_V$, where t_V is the worst case running time of D_V.

For $i < \mu$, $T_i(\sigma, k, y, \mathbf{x}_i)$ outputs fail iff the first call to each T_j subroutine for $j \in [i+1, \mu]$ returns fail, in which case the runtime is $t_{\mathcal{A}}$. The probability $T_i(\sigma, k, \mathsf{z}, \mathbf{x}_i)$ outputs fail for random σ and $\mathbf{x}_i$ is again equal to the failure probability of $\mathcal{A}(\sigma)$, i.e. $1 - \epsilon$. If it does not output fail, then in expectation it runs an additional $(k-1)/\epsilon$ iterations of $T_{i+1}(\sigma, k, \mathsf{z}, \mathbf{x}_i, x_{i+1})$ sampling a fresh x_{i+1} for each iteration. Thus, the expected runtime $\mathbb{E}(t_i)$ is:

$$\mathbb{E}[(1-\epsilon) \cdot t_{\mathcal{A}} + (k-1) \cdot t_{i+1}] \leq \mathbb{E}[t_{i+1} \cdot k]$$

This recurrence relation shows:

$$\mathbb{E}[t_0] \leq \mathbb{E}[t_\mu \cdot k^\mu]$$

Thus, we have shown that the expected runtime of $\mathsf{Tree}(k, \mathsf{z})$ is $\mathbb{E}[t] \leq \frac{k^\mu}{\epsilon} \cdot (\mu + t_V)$.

By standard techniques[10], the Las Vegas algorithm $\mathsf{Tree}(k, \mathsf{z})$ may be transformed to a Monte Carlo algorithm that runs for $2\lambda \cdot \mathbb{E}[t])$ steps and succeeds except with probability $1 - 2^{-\lambda}$.

Transcript tree property analysis The transcript tree labels returned by $\mathsf{Tree}(k, \mathsf{z})$ are computed in depth-first post-order by the Monte Carlo tree generation algorithm. Let L denote this post-order labeling. Consider any node v that is labeled with challenge $L(v)_0 = x$ in the tree. Let i denote the level of v within the tree and let $\mathbf{x} = (x_1, .., x_{i-1}, x, x_{i+1}, ..., x_\mu)$ denote the vector of challenge labels assigned to the path starting from the root to v and following the left-most path down the tree from v. During the execution of $\mathsf{Tree}(k, \mathsf{z})$ the following event occurred: immediately after x was sampled as a candidate label for v, the challenges $x_{i+1}, ..., x_\mu$ were sampled uniformly and independently such that $\mathcal{A}(\sigma, k, \mathsf{z}, \mathbf{x})$ succeeded (i.e., produced a valid transcript). If this event had not occurred (i.e., $\mathcal{A}(\sigma, k, \mathsf{z}, \mathbf{x})$ failed) then x would have been discarded and the process would have been repeated.

In other words, when the transcript tree generation algorithm visits a node v it has already derived a partial labelling L_{v^*} for $[1, v^*]$ where $v^* \leq v$ is not in any subtree extending from v. It samples a random vector $\mathbf{x} \in \mathcal{X}^{h_v}$ and attempts to derive a valid transcript for lpath_v using this challenge vector $\mathbf{x}$. If it succeeds then it sets $L(\mathsf{lpath}_v)_0 = \mathbf{x}$, otherwise the entire vector $\mathbf{x}$ is discarded and it tries again starting from v. Suppose there exists some v such that $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 0$ and let v be the lowest index node with this property. This would imply that there occurred an event where the algorithm had already partially constructed L such that $\pi_u(L_{u^*}, L(\mathsf{lpath}_u)_0) = 1$ for all $u \leq v^*$ and then subsequently sampled $\mathbf{x} \leftarrow \mathcal{X}^{h_v}$, setting $L(\mathsf{lpath}_v)_0 = \mathbf{x}$, such that $\pi_v(L_{v^*}, \mathsf{x}) = 0$. However, by hypothesis

[10]Run λ independent instances in parallel for $2 \cdot \mathbb{E}[t]$ steps. By Markov, each instance terminates (i.e., succeeds) with probability at least 1/2. The probability none succeed is at most $2^{-\lambda}$

this event occurs with probability δ over random $\mathbf{x}$. The algorithm runs for at most $t = \frac{2\lambda}{\epsilon}k^{\mu} \cdot (\mu + t_V)$ steps in total, hence by a union bound the probability that an event of this kind occurs at all is at most $t \cdot \delta$.

Thus, we obtain a Monte Carlo extraction algorithm that returns a transcript tree where all the predicates are satisfied with overwhelming probability (for appropriate setting of the parameters). More precisely, for any security parameter $\lambda \in \mathbb{N}$ and for $t = \frac{2\lambda}{\epsilon} \cdot k^{\mu} \cdot (\mu + t_V)$ the extraction algorithm runs in time at most t and (by a union bound) succeeds in returning a transcript tree labeling L where, for all v, $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 1$ with probability at least $1 - t \cdot \delta - 2^{-\lambda}$.

□

2.8.5 Knowledge Soundness

An NP relation $\mathcal{R}$ is a subset of strings $x, w \in \{0,1\}^*$ such that there is a decision algorithm to decide $(x, w) \in \mathcal{R}$ that runs in time polynomial in $|x|$ and $|w|$. The language of $\mathcal{R}$, denoted $\mathcal{L}_R$, is the set $\{x \in \{0,1\}^* : \exists w \in \{0,1\}^* \ s.t. \ (x, w) \in \mathcal{R}\}$. The string w is called the *witness* and x the *instance*. An **interactive proof of knowledge** for an NP relation $\mathcal{R}$ is a special kind of two-party interactive protocol between a prover denoted $\mathcal{P}$ and a verifier denoted $\mathcal{V}$, where $\mathcal{P}$ has a private input w and both parties have a common public input x such that $(x, w) \in \mathcal{R}$. Informally, the protocol is *complete* if $\mathcal{P}(x, w)$ always causes $\mathcal{V}(x)$ to output 1 for any $(x, w) \in \mathcal{R}$. The protocol is *knowledge sound* if there exists an extraction algorithm $\mathcal{E}$ called the *extractor* such that for every x and adversarial prover $\mathcal{A}$ that causes $\mathcal{V}(x)$ to output 1 with non-negligible probability, $\mathcal{E}$ outputs w such that $(x, w) \in \mathcal{R}$ with overwhelming probability given access[11] to $\mathcal{A}$.

Definition 6 (Interactive Proof of Knowledge)**.** *An interactive protocol* $\Pi = (\mathcal{P}, \mathcal{V})$ *between a prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$ *is a proof of knowledge for a relation* $\mathcal{R}$ *with knowledge error* $\delta : \mathbb{N} \to [0, 1]$ *if the following properties hold, where on common input* x *and prover witness* w *the output of the verifier is denoted by the random variable* $\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$*:*

[11]The extractor can run $\mathcal{A}$ for any specified number of steps, inspect the internal state of $\mathcal{A}$, and even rewind $\mathcal{A}$ to a previous state.

- *Perfect Completeness: for all $(x, w) \in \mathcal{R}$*

$$\Pr[\ \langle \mathcal{P}(x, w), \mathcal{V}(x)\rangle = 1] = 1$$

- *δ-Knowledge Soundness: There exists a polynomial $\mathsf{poly}(\cdot)$ and a probabilistic oracle machine $\mathcal{E}$ called the* extractor *such that given oracle access to any adversarial interactive prover algorithm $\mathcal{A}$ and any input $x \in \mathcal{L}_R$ the following holds: if*

$$\mathbb{P}\left[\langle \mathcal{A}(x), \mathcal{V}(x)\rangle = 1\right] = \epsilon(x)$$

then $\mathcal{E}^{\mathcal{A}}(x)$ with oracle access to $\mathcal{A}$ runs in time $\frac{\mathsf{poly}(|x|)}{\epsilon(x)}$ and outputs w such that $(x, w) \in R$ with probability at least $1 - \frac{\delta(|x|)}{\epsilon(x)}$.

An interactive proof is "knowledge sound", or simply a "proof of knowledge", if has negligible knowledge error δ.

Remark 1. *Definition 6 places no restriction on the runtime of the adversary, however, it does not guarantee extraction from an adversary that succeeds with sufficiently small $\epsilon(x)$ such that $\epsilon(x) \leq \delta(|x|)$. For $\mathcal{R}$ in NP, this definition of knowledge soundness implies the alternative formulation of Bellare and Goldreich [19], which says that the protocol has knowledge error $\delta(|x|)$ if there exists an extractor that succeeds in expected time $\frac{\mathsf{poly}(|x|)}{\epsilon(x)-\delta(|x|)}$. An extractor which succeeds with probability $p = 1 - \frac{\delta(|x|)}{\epsilon(x)}$ in $t = \frac{\mathsf{poly}(|x|)}{\epsilon(x)}$ steps can run repeatedly (for t steps per iteration) on fresh randomness until it obtains a witness for the relation, which it can verify efficiently. It will succeed in an expected $\frac{t}{p} = \frac{\mathsf{poly}(|x|)}{\epsilon(x)-\delta(|x|)}$ steps. Finally, this has been shown to imply another equivalent formulation which requires the extractor to run in $O(\mathsf{poly}(|x|))$ steps and succeed with probability $\frac{\epsilon(x)-\delta(|x|)}{q(|x|)}$ for some polynomial q. It is easy to see this implies the former because such an extractor can be repeated, succeeding in expected time $\frac{q\cdot \mathsf{poly}(|x|)}{\epsilon(x)-\delta(|x|)}$.*

Interactive arguments Knowledge soundness holds against unbounded provers. The DARK protocol does not satisfy knowledge soundness because it relies on the computational binding property of cryptographic commitments. Interactive proofs that are only secure against computationally bounded adversaries are called *interactive arguments*. Adapting Definition 6 for arguments is more subtle than simply restricting the runtime of the adversary. The issue comes from the fact that the knowledge soundness definition quantifies the success of the extractor over all inputs x. For example, there could exist an input x that encodes the factorization of an RSA modulus which allows the adversarial prover to break the binding property of commitments that are based on the difficulty of factoring. For this input, the adversarial prover could succeed while the extractor would fail. This particular problem is fixed by requiring the adversary to generate the input x. If the trapdoor is exponentially hard to compute the polynomial time adversary will not be able to embed the trapdoor in x with non-negligible probability. (See Damgård and Fujisaki [71] for a broader discussion of these issues).

Witness-extended emulation A property called witness-extended emulation [115] strengthens the knowledge-soundness definition so that the extractor outputs not only a witness but also a simulated transcript of the messages between the prover and verifier. This property is helpful for composability. In particular, if the interactive proof is used as a subprotocol within a larger protocol, it may be necessary in the security analysis to construct a simulator that needs to both obtain the adversary's witness as well as simulate its view in the subprotocol. Fortunately, Lindell [115] proved that every knowledge sound protocol also satisfies witness-extended emulation. Groth and Ishai [97] further adapt the definition of witness-extended emulation for interactive arguments with setup (i.e., SRS model). This is the definition we will use in the present work.

Before presenting the definition we will introduce some useful notations. In the SRS model, there is a setup algorithm Setup that generates public parameters pp that are common inputs to the prover $\mathcal{P}$ and verifier $\mathcal{V}$. The setup, which may or may not require a trusted party to sample trapdoor secrets, typically generates these parameters based on a security parameter λ necessary for computational security. Without loss of generality, the length of $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ is at least λ bits. For any prover algorithm $\mathcal{P}^*$ interacting with a verifier algorithm $\mathcal{V}$, which may deviate arbitrarily from the honest prover algorithm $\mathcal{P}$, let $\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})$ denote the message transcript between $\mathcal{P}^*$ and $\mathcal{V}$ on shared inputs x and pp and initial prover state st. For $\mathsf{tr} \leftarrow \mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})$ let $V_{\mathsf{check}}(\mathsf{tr})$ denote the verifier's decision algorithm to accept or reject the transcript. Furthermore, let $\mathcal{E}^{\mathsf{Record}(\mathcal{P}^*,\mathsf{pp},x,\mathsf{st})}$ denote a machine $\mathcal{E}$ with a transcript oracle for this interaction that can be rewound to any round and run again on fresh verifier randomness.

Definition 7 (Witness-extended emulation [97, 115])**.** *An interactive proof in the SRS model $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ satisfies witness-extended emulation for relation $\mathcal{R}$ if for every deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for any non-uniform*[12] *adversary $\mathcal{A}$ and distinguisher $\mathcal{D}$ that runs in time $\mathsf{poly}(\lambda)$ the following condition holds:*

$$\Pr\left[\mathcal{D}(tr) = 1 : \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(pp) \\ tr \leftarrow Record(\mathcal{P}^*, pp, x, \mathsf{st}) \end{array}\right] \approx_\lambda$$

$$\Pr\left[\begin{array}{c} \mathcal{D}(tr) = 1 \text{ and} \\ V_{check}(tr) = 1 \Rightarrow (x, w) \in \mathcal{R} \end{array} : \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(pp) \\ (tr, w) \leftarrow \mathcal{E}^{Record(\mathcal{P}^*, pp, x, \mathsf{st})}(pp, x) \end{array}\right]$$

where $X \approx_\lambda Y$ denotes that $|X - Y| \le \mathsf{negl}(\lambda)$.

Lemma 12 (Lindell [115])**.** *Any proof of knowledge for relation $\mathcal{R}$ also satisfies witness-extended emulation for $\mathcal{R}$.*

Lemma 13. *Let $\mathcal{R}$ denote any NP relation. Given a commitment scheme $\mathsf{com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, for any $pp \leftarrow \mathsf{Setup}(\lambda)$ let $\mathcal{R}'(pp)$ denote the relation:*

$$\mathcal{R}'(pp) = \{(\mathsf{x}, \mathsf{w}) : \mathcal{R}(\mathsf{x}, \mathsf{w}) = 1 \vee [\mathsf{w} = (C, \sigma_1, \sigma_2) \ \wedge \ \sigma_1 \neq \sigma_2 \ \wedge \ Open(pp, C, \sigma_1) = Open(pp, C, \sigma_2) = 1]\}$$

[12] A non-uniform adversary may run a different algorithm for each input length.

Let $\Pi(\mathsf{pp})$ denote the interactive protocol between $\mathcal{P}$ and $\mathcal{V}$ parameterized by the setup parameter pp. If for all $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ the protocol $\Pi(\mathsf{pp})$ is a proof of knowledge (Definition 6) for $\mathcal{R}'(\mathsf{pp})$ then the tuple $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ as an interactive proof with setup satisfies witness-extended emulation (Definition 7) for $\mathcal{R}$.

Proof. By Lemma 12 a knowledge sound interactive proof for $\mathcal{R}'(\mathsf{pp})$ also satisfies witness-extended emulation for $\mathcal{R}'(\mathsf{pp})$. It remains to show that this implies witness-extended emulation for $\mathcal{R}$. It suffices to show that:

$$\Pr\left[\begin{array}{c} \mathcal{D}(\mathsf{tr}) = 1 \text{ and} \\ V_{\mathsf{check}}(\mathsf{tr}) = 1 \Rightarrow \ (x, w) \in \mathcal{R} \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})}(\mathsf{pp}, x) \end{array}\right] \approx_\lambda$$

$$\Pr\left[\begin{array}{c} \mathcal{D}(\mathsf{tr}) = 1 \text{ and} \\ V_{\mathsf{check}}(\mathsf{tr}) \Rightarrow \ (x, w) \in \mathcal{R}'(\mathsf{pp}) \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})}(\mathsf{pp}, x) \end{array}\right]$$

The difference between these two probabilities is bounded by the probability, over the distribution on the right side of the equation, that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}'(\mathsf{pp})$ but $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}$. This event implies that w encodes a break to the commitment scheme with parameters pp. Since $\mathcal{A}$, P^*, Setup and $\mathcal{E}$ all run in time $\mathsf{poly}(\lambda)$ this occurs with probability at most $\mathsf{negl}(\lambda)$ over randomly sampled $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ by the computational binding property of the commitment scheme. More precisely, supposing that the difference between these two probabilities is $\epsilon(\lambda)$, then we can use $\mathcal{A}$, P^*, and $\mathcal{E}$ to construct an algorithm $\mathcal{A}'$ which on input $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ simulates $(x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$ and $(\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})}(\mathsf{pp}, x)$ returning w such that:

$$\Pr\left[w = (C, \sigma_1, \sigma_2) \ \wedge \ \sigma_1 \neq \sigma_2 \ \wedge \ \mathsf{Vf}_{\mathsf{pp}}(C, \sigma_1) = \mathsf{Vf}_{\mathsf{pp}}(C, \sigma_2) = 1 : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ w \leftarrow \mathcal{A}'(\mathsf{pp}) \end{array}\right] = \epsilon(\lambda)$$

If $\epsilon(\lambda)$ is non-negligible this contradicts the binding property of the commitment scheme.

□

Zero knowledge We recall the definition of *honest verifier zero-knowledge* (HVZK) for interactive proofs. HVZK only considers simulating the view of a verifier that follows the protocol honestly. The Fiat-Shamir transform compiles public-coin proofs that have HVZK into non-interactive proofs that have statistical zero-knowledge (for malicious verifiers).

Definition 8 (HVZK for interactive arguments)**.** *Let $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle}$ denote the view of the verifier in an interactive protocol on common input x and prover witness input w. The interactive protocol has δ-statistical honest verifier zero-knowledge if there exists a probabilistic polynomial time algorithm $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the distribution $\mathcal{S}(x)$ is δ-close to $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle}$ (as distributions over the randomness of $\mathcal{P}$ and $\mathcal{V}$).*

2.8.6 Almost-Special-Soundness Theorems

Definition 9 (Almost-Special-Soundness)**.** *A μ-round public-coin interactive proof for a relation $\mathcal{R}$ with challenge space of size 2^λ is $(k^{(\mu)}, \delta(\cdot), \mathsf{com}, \phi)$-**almost-special-sound** if it satisfies the following conditions with respect to some commitment scheme $\mathsf{com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ with message space $\mathcal{M}$ and opening space $\mathcal{W}$, a pair of predicates $\phi = (\phi_a, \phi_b)$ where $\phi_a, \phi_b : [\mu] \times \mathcal{M} \rightarrow \{0, 1\}$, and a negligible function $\delta : \mathbb{N} \rightarrow \mathbb{R}$:*

1. *The setup for the interactive proof includes generation of the public parameters for the commitment scheme $\mathsf{pp} \leftarrow \mathsf{com.Setup}(\lambda)$.*

2. *In any accepting transcript, the prover's ith round message for $i \in [1, \mu)$ is a valid commitment $\mathcal{C}_i$ for the scheme com, and the final prover message is a commitment $\mathcal{C}_\mu$ together with a valid opening (m_μ, o_μ) such that $\mathsf{com.Open}(\mathsf{pp}, C_\mu, m_\mu, o_\mu) = 1$ and $\phi_a(\mu, m_\mu) = 1$.*

3. *There is a $poly(\lambda)$ time algorithm $\mathsf{Extract}(i, \nu, C_\nu, \mathsf{openSubtree}) \rightarrow (m, o)$ where $i \in [1, \mu]$ and $\mathsf{openSubtree}$ is a list of openings for the commitments on all internal nodes (excluding the root and leaves) of a k-ary depth $\mu - i$ subtree of a transcript tree rooted at a node ν on the ith level with commitment label C_ν. If the challenge labels on the first two children of any node in the subtree are distinct, and the openings for*

all internal (non-root) nodes of the subtree satisfy predicate ϕ_a (i.e., for any $j > i$ and node u on the jth level of the transcript tree that is a member of this subtree, its opening (m_u, o_u) in $\mathsf{openSubtree}$ satisfies $\phi_a(j, m_u) = 1$) then the algorithm returns a valid opening (m, o) for c_ν such that $\phi_b(i, m) = 1$.

4. *$\mathsf{Extract}(0, x, \mathsf{openTree}) \to w$ takes as inputs openings for the commitments on all nodes in an entire transcript tree satisfying predicate ϕ_a (same condition as above for subtrees) and returns a witness w for the public input x such that $\mathcal{R}(\mathsf{x}, \mathsf{w}) = 1$.*

5. *$\mathsf{Extend}(i, m, \alpha_1, .., \alpha_{\mu-i})$ is a deterministic $\mathsf{poly}(\lambda)$-time algorithm that is given an index $i \in [\mu - 1]$, a message m in the message space of the commitment scheme com, $\mu - i$ challenges from $\mathcal{X}$, and outputs $\mu - i$ messages $m'_1, ..., m'_{\mu-i}$ in the message space of the commitment scheme.*

6. *For any $i \in [\mu - 1]$ and m where $\phi_a(i, m) = 0$, the probability over $\alpha_i, ..., \alpha_\mu$ sampled uniformly i.i.d. from $\mathcal{X}$ that the last message $m'_{\mu-i}$ in the list returned by $\mathsf{Extend}(i, m)$ satisfies $\phi_a(\mu, m'_{\mu-i}) = 1$ is bounded by $\delta(\lambda)$.*

7. *$\mathsf{Break}(i, m, \alpha_1, ..., \alpha_\mu, C_0, ..., C_\mu, (m_i, o_i), ..., (m_\mu, o_\mu))$ first runs $\mathsf{Extend}(i, m, \alpha_i, .., \alpha_\mu)$, which returns messages $m'_1, ..., m'_{\mu-i}$. If either $\phi_b(i, m_i) = 0$ or $\forall_j \; m'_j = m_{i+j-1}$ then it outputs $\perp$. Otherwise it outputs an attempted opening (m', o') of C_j for some index $j \geq i$ where $m' \neq m_j$.*

8. *For any $i \in [\mu - 1]$, given a valid (accepting) transcript with commitments $\mathbf{C} = (C_0, ..., C_\mu)$, round challenges $\mathbf{r} = (\alpha_1, ..., \alpha_\mu)$, and openings $\mathbf{open} = ((m_i, o_i), ..., (m_\mu, o_\mu))$ to the last $\mu - i + 1$ commitments, where $\phi_b(i, m_i) = 1$ and $\phi_a(j, m_j) = 1$ for all $j \in [i + 1, \mu]$, either $\mathsf{Extend}(i, m_i, \alpha_{i+1}, ..., \alpha_\mu)$ returns $m_{i+1}, ..., m_\mu$ or $\mathsf{Break}(i, m_i, \mathbf{r}, \mathbf{C}, \mathbf{open})$ returns an opening (m', o') of some C_j to a conflicting message $m' \neq m_j \in \mathcal{M}$, which breaks the binding of the commitment scheme over $\mathcal{M}$.*

Short-hand notation: *An interactive proof is $(k^{(\mu)}, \delta)$-almost-special-sound if it is $(k^{(\mu)}, \delta, \mathsf{com}, \phi)$-almost-special-sound for some commitment scheme com and some predicate* *pair ϕ. We may omit δ and simply write $k^{(\mu)}$-almost-special-soundness if this holds for some negligible function $\delta : \mathbb{N} \to \mathbb{R}$.*

Remark 2. *Any special sound protocol satisfies almost-special-soundness as 3) essentially captures the special soundness definition. More precisely a $k^{(\mu)}$-special sound satisfies $k^{(\mu)}$-almost-special-soundness by setting the commitment scheme to be trivial (i.e., identity function) and the ith round commitment C_i to the prover's ith round message and setting the predicates $\phi_a = 1, \phi_b = 0$ to be trivial as well (i.e., always return 1 and 0 respectively). The algorithm Extend can output an arbitrary set of messages because the condition on the algorithm is vacuously true as $\phi_a(i, m) \neq 0$ for any (i, m). The algorithm $\mathsf{Extract}(i, \nu, C_\nu, *)$ is trivial because C_ν is the message itself. The algorithm Break is also trivial as ϕ_b is always 0. The algorithm $\mathsf{Extract}(0, x, openTree) \to w$ exists by the definition of $k^{(\mu)}$-special soundness.*

Theorem 6. *If a μ-round interactive proof for a relation $\mathcal{R}$ with λ-bit challenges, $\mu \in O(\log(\lambda + |\mathsf{x}|))$, and verifier decision algorithm runtime $t_V \in \mathsf{poly}(|\mathsf{pp}|, |\mathsf{x}|, \lambda)$ on input $\mathsf{x} \in \mathcal{L}_{\mathcal{R}}$ and parameters $\mathsf{pp} \leftarrow \mathsf{com.Setup}(\lambda)$ is $(k^{(\mu)}, \delta, \mathsf{com}, \phi)$-almost-special-sound then for $\delta'(\lambda) = 2\lambda(k+1)^{\mu}(\mu + t_V) \cdot \max(\delta(\lambda), k \cdot 2^{-\lambda}) + 2^{-\lambda}$ it is δ'-knowledge sound for the modified relation:*

$$\mathcal{R}'(\mathsf{pp}) = \{(\mathsf{x}, \mathsf{w}) : \mathcal{R}(\mathsf{x}, \mathsf{w}) = 1 \ \vee \mathsf{w} \in \mathcal{L}_{\mathsf{break}}(\mathsf{pp})\}$$

where

$$\mathcal{L}_{\mathsf{break}}(\mathsf{pp}) = \{(\mathcal{C}, \sigma_1, \sigma_2) : \sigma_1 \neq \sigma_2 \ \wedge \ \mathsf{Open}(\mathsf{pp}, \mathcal{C}, \sigma_1) = \mathsf{Open}(\mathsf{pp}, \mathcal{C}, \sigma_2) = 1\}$$

Remark 3. *$\delta'(\lambda)$ is a negligible function if $\delta(\lambda)$ is negligible, assuming $k \in O(1)$, $\mu \in O(\log(\lambda + |\mathsf{x}|))$, $t_V \in \mathsf{poly}(|\mathsf{x}|, \lambda)$, and $|\mathsf{x}| \in \mathsf{poly}(\lambda)$.*

By Lemma 13, this theorem has the following corollary:

Corollary 1. *An interactive proof with λ-bit challenges that is $k^{(\mu)}$-almost-special-sound for a relation $\mathcal{R}$ and has at most $\mu \in O(\log(\lambda + |\mathsf{x}|))$ rounds on any instance $\mathsf{x} \in \mathcal{L}_{\mathcal{R}}$ has witness-extended emulation for $\mathcal{R}$.*

Proof. Suppose we have a protocol that is $(k^{(\mu)}, \delta)$-almost-special-sound with challenge space $\mathcal{X}$ of size 2^λ for some negligible function $\delta : \mathbb{N} \to \mathbb{R}$. We will make use of algorithms $\mathsf{Extract}$, Extend, and Break and their properties that are guaranteed to exist by the definition of almost-special-soundness (Definition 9).

For any node ν of a $(k+1)$-ary transcript tree let S^*_ν denote the *left* k-ary subtree rooted at ν defined by a breadth first search from ν that visits only the first k children of each node reached (i.e., prunes the rightmost branch from each node of the complete $(k+1)$-ary subtree S_ν).

In Definition 10, we define an algorithm $\mathsf{TreeExtract}(\ell_\nu, \nu, \mathcal{C}_\nu, L_{S_\nu})$ that operates on a labeled subtree of a $(k+1)$-ary transcript tree that has depth μ, where ℓ_ν is the level of ν, $\mathcal{C}_\nu = L(\nu)_1$ is the commitment label on ν and L_{S_ν} is a labeling of the $(k+1)$-ary subtree S_ν rooted at ν. If $\mathsf{TreeExtract}(\ell_\nu, \nu, \mathcal{C}_\nu, L_{S_\nu})$ succeeds it returns $\mathsf{openSubtree}$, which contains openings of all the commitment labels L assigned to nodes in S_ν including the label $\mathcal{C}_\nu$ on node ν. Otherwise it returns $\perp$. The $\mathsf{TreeExtract}$ algorithm is not guaranteed to succeed. In particular, the internal calls to $\mathsf{Extract}$ are only guaranteed to succeed when the openings of subtrees satisfy predicate ϕ_a and the challenge labels are distinct within the pruned subtrees S^*_ν. Definition 10 also defines $\mathsf{TreeExtract}^*(\ell_\nu, \mathcal{C}_\nu, L_{S^*_\nu})$, an algorithm that only extracts openings of the commitments in $L_{S^*_\nu}$ and returns an opening of C_ν. This runs similarly to $\mathsf{TreeExtract}$, but it is only a function of nodes present in the left k-ary subtree S^*_ν. While it is possible that $\mathsf{TreeExtract}$ fails and $\mathsf{TreeExtract}^*$ succeeds, they will always output the same opening of C_ν in the event that both succeed.

Let $\mathsf{size}(k, \mu) = \frac{k^{\mu+1}-1}{k-1}$, which is the number of nodes is a k-ary depth μ tree. Given any μ-round protocol that satisfies $(k^{(\mu)}, \delta)$-almost-special-soundness, setting $N = \mathsf{size}(k+1, \mu)$ we will define a collection of predicates $\{\pi_\nu : \nu \in [1, N]\}$ for the nodes of a k-ary transcript tree with post-order labeling L, such that each π_ν is a function of the partial labeling L_{ν^*} and a $\mu - \ell_\nu$-length challenge vector $\mathbf{r} \in \mathcal{X}^{\mu - \ell_\nu}$, where ℓ_ν is the level of node ν in the tree. Recall that $\nu^* < \nu$ is the node of highest index smaller than ν that is not a member of the subtree of ν, and L_{ν^*} are the labels of all nodes numbered $[1, \nu^*]$. Let ω denote the parent node of ν. The predicate $\pi_\nu(L_{\nu^*}, \mathbf{r})$ is defined as follows:

- If ν has no left-sibling, then π_ν always returns 1.
- If ν has $0 < i < k$ left-siblings (i.e., it is neither the first nor last child) then $\pi_\nu(L_{\nu^*}, \mathbf{r}) = 1$ iff the challenge label $L(\nu)_0$ assigned to ν is distinct from the challenge labels assigned to its i left-siblings. Note that if ν' is a left-sibling of ν then $\nu' \in [1, \nu^*]$ so $L(\nu')$ is included in the input L_{ν^*} to π_ν.
- If ν has no right-sibling (i.e., is rightmost child) then let $\mathcal{C}_\omega = L(\omega)_1$ denote the commitment label on ω, let (m_ω, o_ω) denote the opening of $\mathcal{C}_\omega$ returned by $\mathsf{TreeExtract}^*(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S^*_\omega})$ if successful, let m' denote the last message in the output list of $\mathsf{Extend}(\ell_\omega, m_\omega, \mathbf{r})$ and finally:

$$\pi_\nu(L_{\nu^*}, \mathbf{r}) = \begin{cases} 1 & \text{if } \mathsf{TreeExtract}^*(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S^*_\omega}) = \bot \\ 1 & \text{if } \phi_a(\ell_\omega, m_\omega) = 1 \\ 1 & \text{if } \phi_a(\ell_\omega, m_\omega) = 0 \wedge \phi_a(\mu, m') = 0 \\ 0 \text{ otherwise} \end{cases}$$

As remarked above, while $\mathsf{TreeExtract}$ operates on the entire $(k+1)$-ary subtree of labels rooted at ω, the algorithm $\mathsf{TreeExtract}^*$ takes as input only the labeling of the right k-ary subtree S^*_ω and $L_{S^*_\omega} \subseteq L_{\nu^*}$. By the definition of $(k^{(\mu)}, \delta)$-almost-special-soundness (Definition 9, pt. 5), for any $\nu \in [0, N)$ that has no right-sibling and any L_{ν^*}:

$$\mathbb{P}_{\mathbf{r} \leftarrow \mathcal{X}^{\mu - \ell_\nu}}[\pi_v(L_{v^*}, \mathbf{r}) = 1] \geq 1 - \delta(\lambda)$$

If ν has $0 < i < k$ left-siblings then by a union bound:

$$\mathbb{P}_{\mathbf{r} \leftarrow \mathcal{X}^{\mu - \ell_\nu}}[\pi_v(L_{v^*}, \mathbf{r}) = 1] \geq 1 - \frac{i}{2^\lambda}$$

Let $\mathsf{lpath}(\nu)_0$ denote the challenge labels $L(\cdot)_0$ along the leftmost branch from ν to a leaf starting with the label $L(\nu)_0$ on ν. By Lemma 11 (Path Predicate Forking Lemma) there is an algorithm $\mathsf{Tree}^{\mathcal{A}}(\mathsf{z})$ that, given a security parameter $\lambda \in \mathbb{N}$, an input $\mathsf{x} \in \mathcal{L}_{\mathcal{R}}$, and oracle access to an adversarial prover $\mathcal{A}$ that causes V to accept on input x with probability ϵ, runs in time at most $t = 2\lambda \cdot \frac{(k+1)^\mu}{\epsilon} \cdot (\mu + t_V)$, where t_V is the worst-case running time of verifier's decision algorithm, and returns with probability at least $1 - t \cdot \max(\delta(\lambda), \frac{k}{2^\lambda}) - 2^{-\lambda}$ a $(k+1)$-ary transcript tree with post-order labelling $L : [1, N] \to \mathcal{X} \times \mathcal{M}$ such that $\pi_v(L_{v^*}, L(\mathsf{lpath}_v)_0) = 1$ for all $v \in [1, N]$.

In particular, L defines a $(k+1)$-ary transcript tree with the properties:

1. The challenge labels on the first k children of any node are distinct, i.e., if ω has children $\nu_1, ..., \nu_{k+1}$ ordered from left-to-right, then for any $i, j \in [1, k]$ if $i \neq j$ then $L_0(\nu_i) \neq L_0(\nu_j)$.

2. If ν is the $(k+1)$th child of ω and running $\mathsf{TreeExtract}(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S^*_\omega})$ at level ℓ_ω returns an opening of $\mathcal{C}_\omega$ to m_ω such that $\phi_a(\ell_\omega, m_\omega) \neq 1$, then the final output of $\mathsf{Extend}(\ell_\omega, m_\omega, \mathsf{lpath}(\nu)_0)$ is a message m' such that $\phi_a(\mu, m') \neq 1$.

By Lemma 14 there is a deterministic extraction algorithm that takes any L with the above properties and computes a witness w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}'$.

In conclusion, for any adversarial prover that succeeds on input x with probability $\epsilon(x)$, there is a probabilistic extractor that runs in time at most $t = 2\lambda\frac{(k+1)^\mu}{\epsilon(x)}(\mu + t_V)$ and with probability at least $1 - t \cdot \max(\delta(\lambda), \frac{k}{2^\lambda}) - 2^{-\lambda}$ returns a witness for $\mathcal{R}'$. Since $t \in \frac{\mathsf{poly}(|x|,\lambda)}{\epsilon(x)}$ assuming $\mu \in O(\log(\lambda + |x|))$ and $t_V \in \mathsf{poly}(|x|, \lambda)$, this satisfies the definition of δ'-knowledge soundness with $\delta'(\lambda) = 2\lambda(k+1)^\mu(\mu + t_V) \cdot \max(\delta(\lambda), k \cdot 2^{-\lambda}) + 2^{-\lambda}$, which is a negligible function of λ as long as $\delta(\lambda)$ is negligible.

□

Definition 10 (Tree Extractor)**.** *We define an algorithm* $\mathsf{TreeExtract}(k, \ell, \nu, \mathcal{C}_\nu, L_{S_\nu})$ *that operates on a labeled subtree of a* $(k+1)$*-ary transcript tree that has depth* μ*, where* ℓ_ν *is the level of* ν*,* $\mathcal{C}_\nu = L(\nu)_1$ *is the commitment label on* ν *and* L_{S_ν} *is a labeling of the* $(k+1)$*-ary subtree* S_ν *rooted at* ν*. If* $\mathsf{TreeExtract}(\ell_\nu, \nu, \mathcal{C}_\nu, L_{S_\nu})$ *succeeds it returns* $\mathsf{openSubtree}$*, which contains openings of all the commitment labels* L *assigned to nodes in* S_ν *including the label* $\mathcal{C}_\nu$ *on node* ν*. Otherwise it returns* $\perp$*. The algorithms runs as follows:*

- *For all leaf nodes ν, return the opening of $\mathcal{C}_\nu$, which is included in the label on ν.*
- *For each node $\omega \in S_\nu$ on the second to last level with label $\mathcal{C}_\omega = L(\omega)_1$, set $\mathsf{openLeaves}$ to include the first k opened leaves of ω, and run $\mathsf{Extract}(\mu - 1, \omega, \mathcal{C}_\omega, \mathsf{openLeaves})$ to get an opening of $\mathcal{C}_\omega$.*
- *Continue iteratively: once openings for all commitment labels of all subtrees rooted at the ith level have been computed, for each node ω on the $(i+1)$st level with label $\mathcal{C}_\omega = L(\omega)_1$ run $\mathsf{Extract}(i, \omega, \mathcal{C}_\omega, \mathsf{openSubtree}^*_\omega)$ on the commitment label openings $\mathsf{openSubtree}^*_\omega$ of the left k-ary subtree S^*_ω (excluding node ω), which were computed in prior iterations.*

Finally, $\mathsf{TreeExtract}^*(\ell_\nu, \mathcal{C}_\nu, L_{S^*_\nu})$ denotes the algorithm that only extracts openings of the commitments in $L_{S^*_\nu}$ and returns an opening of $\mathcal{C}_\nu$. This runs exactly like $\mathsf{TreeExtract}$ except that it only iterates over nodes that are present in the left k-ary subtree S^*_ν.

The $\mathsf{TreeExtract}$ algorithm is not guaranteed to succeed. In particular, the internal calls to $\mathsf{Extract}$ are only guaranteed to succeed when the openings of subtrees satisfy predicate ϕ_a and the challenge labels are distinct within the pruned subtrees S^*_ν. By convention, if any internal step fails then $\mathsf{TreeExtract}$ outputs $\perp$. While it is possible that $\mathsf{TreeExtract}$ fails and $\mathsf{TreeExtract}^*$ succeeds, it is easy to see that they output the same opening of C_ν assuming both succeed. Furthermore, while $\mathsf{TreeExtract}$ operates on $(k+1)$-ary transcript tree, the internal calls to $\mathsf{Extract}$ run on k-ary transcript trees because it is defined for a protocol that is $(k^{(\mu)}, \delta)$-almost-special-sound. The reason we always pass the labeling/opening of the left k-ary subtree (as opposed to an arbitrary k-ary subtree) to $\mathsf{Extract}$ is to ensure that the opening of $\mathcal{C}_\nu$ included in the output of $\mathsf{TreeExtract}(\ell_\nu, \mathcal{C}_\nu, L_{S_\nu})$ is a function of only the labels on the left k-ary subtree S^*_ν, and in particular is computed independently from any of the labels in the (rightmost) subtree rooted at the $(k+1)$th (rightmost) child of ν. This fact is used in the proof of Theorem 6.

Definition 11 (Predicate Special Soundness)**.** *Let ρ denote any binary predicate that takes as input any k-ary μ-depth transcript tree. A μ-round public coin interactive proof for a* *relation $\mathcal{R}$ with λ-bit challenges is $(k^{(\mu)}, \rho)$-**special sound** if there exists a deterministic extraction algorithm $\mathcal{E}$ that takes as input an instance $\mathsf{x} \in \mathcal{L}_{\mathcal{R}}$, any k-ary forking transcript tree rooted at x with labelling L such that $\rho(L) = 1$, and returns a witness w such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$ in time $\mathsf{poly}(\lambda, k^{\mu})$.*

Setting $\rho = 1$, i.e. the trivial predicate that is always true, recovers the standard definition of $k^{(\mu)}$-special soundness. Recall that we defined a *forking* transcript tree (Section 2.8.3) as a transcript tree in which the challenge labels on edges that share the same parent node are distinct.[13]

Lemma 14. *Let $\Pi(\mathsf{pp})$ denote a $(k^{(\mu)}, \delta, \mathsf{com}, \phi)$-almost-special-sound protocol for a relation $\mathcal{R}$ and any $\delta \in [0, 1]$, parametrized by $\mathsf{pp} \leftarrow \mathsf{com.Setup}(\lambda)$. Define the binary predicate ρ as a function of a $(k + 1)$-ary μ-depth forking transcript tree given by labelling L, which uses the algorithms $\mathsf{TreeExtract}$ from Definition 10 and Extend from Definition 9 and returns 1 iff the following condition holds:*

> *For any node ω with $(k+1)$st child ν, if the result of running $\mathsf{TreeExtract}(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S^*_\omega})$ at level ℓ_ω returns an opening of $\mathcal{C}_\omega$ to m_ω such that $\phi_a(\ell_\omega, m_\omega) \neq 1$, then the final output of $\mathsf{Extend}(\ell_\omega, m_\omega, \mathsf{lpath}(\nu)_0)$ is a message m' such that $\phi_a(\mu, m') \neq 1$.*

$\Pi(\mathsf{pp})$ is $((k + 1)^{(\mu)}, \rho)$-special sound for the relation $\mathcal{R}'(\mathsf{pp})$ defined in Theorem 6.

Remark 4. *The value of δ does not affect $((k + 1)^{\mu}, \rho)$-special soundness. The value of δ affects the runtime of the extraction algorithm that is able to generate a transcript tree satisfying the predicate ρ (in Theorem 6).*

Proof. We will argue that, assuming the $(k + 1)$-ary forking transcript tree has property ρ, for any $\omega \in [1, N]$, either $\mathsf{TreeExtract}(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S_\omega})$ returns a subtree $\mathsf{openSubtree}$ of openings of the commitment labels in L_{S_ω} satisfying ϕ_a (i.e., each opening of a label C_ω to m_ω for a node ω on level ℓ_ω satisfies $\phi_a(\ell_\omega, m_\omega) = 1$) or else there is an efficient algorithm that uses $\mathsf{openSubtree}$ and L to break the commitment scheme.

[13] We could have defined predicate special soundness in an even more general way such that the forking property of the tree is not required, yet can be encapsulated in the predicate. However, this would not be useful for our present work and less convenient for notational purposes.

Step 1: Suppose that ω is a node of highest level ℓ_ω for which this fails, i.e. the output of $\mathsf{TreeExtract}$ satisfies ϕ_a for any node of higher level than ℓ_ω. This means that all the openings of internal (non-root) nodes of the subtree L_{S_ω} computed while running $\mathsf{TreeExtract}$ on ω satisfy ϕ_a. Furthermore, L has the property that all labels on the first k siblings are distinct.

Step 2: By the definition of almost-special-soundness and the hypothesis in **Step 1**, the algorithm $\mathsf{TreeExtract}(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S_\omega})$ succeeds in returning $\mathsf{openSubtree}$ consisting of the openings of L_{S_ω} such that the openings of all internal (non-root) nodes satisfy ϕ_a, and the opening (m_ω, o_ω) of the subtree root ω satisfies $\phi_b(\ell_\omega, m_\omega) = 1$. The opening (m_ω, o_ω) is also identical to the output of $\mathsf{TreeExtract}^*(\ell_\omega, \omega, \mathcal{C}_\omega, L_{S^*_\omega})$.

Step 3: Let ν denote the rightmost child of ω. By hypothesis, if $\phi_a(\ell_\omega, m_\omega) = 0$ then the final output of $\mathsf{Extend}(\ell_\omega, m_\omega, \mathsf{lpath}(\nu))$ is a message m' such that $\phi_a(\mu, m') = 0$. However, this implies that m' must be distinct from the label L assigns to the leaf node of the rightmost branch extending from ν. Let $v_1, ..., v_\mu$ denote the nodes along the root-to-leaf path passing through node ω and ending with its leftmost branch so that $\mathsf{lpath}(\nu) = (L(v_{\ell_\nu})_0, ..., L(v_\mu)_0)$. For each $i \in [1, \mu]$ let $\hat{C}_i = L(v_i)_1$ and $\hat{\mathbf{C}} = (\hat{C}_1, ..., \hat{C}_\mu)$. Finally, since $\phi_b(\ell_\omega, m_\omega) = 1$ and $\mathsf{openSubtree}$ contains openings $(m_{\ell_\nu}, o_{\ell_\nu}), ...(m_\mu, o_\mu)$ of the commitments $\hat{C}_{\ell_\nu}, ...\hat{C}_\mu$ that all satisfy predicate $\phi_a(i, m_i) = 1$, if $m' \neq m_\mu$ then by the definition of almost-special-soundness this implies that $\mathsf{Break}(\ell_\omega, m_\omega, \mathsf{rpath}(\nu), \hat{\mathbf{C}}, (m_{\ell_\nu}, o_{\ell_\nu}), ...(m_\mu, o_\mu))$ outputs a conflicting opening of some commitment label in $\hat{\mathbf{C}}$.

Let $\mathcal{C}_1 = L(1)_1$, the transcript tree root. The extractor runs $\mathsf{TreeExtract}(0, 1, \mathcal{C}_1, L_{S_1})$, which returns $\mathsf{openTree}$. If every opening in $\mathsf{openTree}$ satisfies predicate ϕ_a then it runs $\mathsf{Extract}(0, \mathsf{x}, \mathsf{openTree})$ to obtain witness w satisfying $R(\mathsf{x}, \mathsf{w}) = 1$. Otherwise, it uses the Break algorithm (as described in the previous step) to output conflicting openings of a commitment, which is a witness for $R'(\mathsf{pp})$.

□

2.8.7 DARK is Almost-Special-Sound

We now prove Lemma 5 from Section 2.5.3, which states that the DARK polynomial commitment scheme is correct for μ-linear polynomials in $\mathbb{Z}_p[X]$.

Proof. In order to ensure correctness we must ensure that $b < q/2$ and that $|f| \leq b$. To show this we show that in each recursion step the honest prover's witness polynomial has coefficients bounded by b and is μ-linear. We argue inductively that for each recursive call of EvalB the following constraints on the inputs are satisfied: $f(X_1 \ldots, X_\mu)$ is μ-linear. C encodes the polynomial, *i.e.*, $\mathsf{C} = \mathsf{G}^{f(\vec{q})}$ and $f(X) \in \mathbb{Z}(b)$. Also $f(z_1, \ldots, z_\mu) = y \bmod p$.

Initially, during the execution of Eval, the prover maps the coefficients of a polynomial $\tilde{f}(X_1, \ldots, X_\mu) \in \mathbb{Z}_p$ to a μ-linear integer polynomial $f(X_1, \ldots, X_\mu)$ with coefficients in $\mathbb{Z}(p-1)$ such that $\mathsf{C} = \mathsf{G}^{f(\vec{q})}$. Additionally $f(z_1, \ldots, z_\mu) \bmod p = \tilde{f}(z_1, \ldots, z_\mu) = y$.

If f is μ-linear then in round i of the protocol the $\mathcal{P}$ can compute $i-1$ linear polynomials f_L and f_R such that $f_L(X_1, \ldots, X_{i-1}) + X_i f_R(X_1, \ldots, X_{i-1}) = f(X_1, \ldots, X_i)$. Consequently $f(z_1, \ldots, z_i) \bmod p = f_L(z_1, \ldots, z_{i-1}) + z_i f_R(z_1, \ldots, z_{i-1}) \bmod p = y_L + z_i y_R \bmod p = y$. The PoE protocol has perfect correctness so $\mathsf{G}^{f_L(q) + q^{\frac{d+1}{2}} f_R(X)} = \mathsf{C}$. Finally $f' = f_L + \alpha \cdot f_R \in \mathbb{Z}(2^\lambda \cdot b)$ is an $i-1$-linear polynomial with coefficients bounded in absolute value by $(2^\lambda - 1) \cdot b + b = 2^\lambda b$, as $\alpha \in [0, 2^\lambda)$. This is precisely the value of b' the input to the next call of EvalB. The value y' is also correct: $f'(z_1, \ldots, z_{i-1}) \bmod p = f_L(z_1, \ldots, z_{i-1}) + \alpha \cdot f_R(z_1, \ldots, z_{i-1}) \bmod p = y_L + \alpha \cdot y_R \bmod p = y'$

In the final round, the prover sends f, and the verifier checks that $|f| < b$ which is true by construction. $\square$

Security of PoE substitutions We first begin by showing that we can safely replace all of the PoE evaluations with direct verification checks. Concretely, under the Adaptive Root Assumption, the Eval protocol is as secure as the protocol Eval' in which all PoEs are replaced by direct checks. We show that the witness-extended emulation for Eval' implies the same property for Eval. This is useful because we will later show how to can build an extractor for Eval', thereby showing that the same witness-extended emulation property extends to Eval.

Lemma 15. *Let* Eval' *be the protocol that is identical to* Eval *but in line 15 of* $\mathsf{EvalB}\ \mathcal{V}$ *directly checks* $\mathsf{C}_L + q^{(2^{\mu-1})} \cdot \mathsf{C}_R = \mathsf{C}$ *instead of using a* PoE*. If the Adaptive Root Assumption holds for GGen, and* Eval' *has witness-extended emulation for* $O(\log(\lambda))$*-linear polynomials, then so does* Eval*.*

Proof. We show that if an extractor E', as defined in Definition 7, exists for the protocol Eval' then we can construct an extractor E for the protocol Eval. Specifically, E simulates E' and presents it with a $\mathsf{Record}'(\cdots)$ oracle, while extracting the witness from its own $\mathsf{Record}(\cdots)$ oracle.

Whenever E' queries the Record' oracle, E queries its Record oracle and relays the response after dropping those portions of the transcript that correspond to the PoE proofs. Whenever E' rewinds its prover, so does E rewind its prover. When E' terminates by outputting a transcript-and-witness pair (tr', f), E adds PoEs into this transcript to obtain tr and outputs (tr, f).

For each PPT adversary $(\mathcal{A}, P^*)$, E will receive a polynomial number of transcripts from its Record oracle. Any transcript tr of Eval such that $\mathcal{A}(\mathsf{tr}) = 1$ and tr is accepting contains exactly μ $\mathsf{PoE}s$ transcripts. So in total E sees only a polynomial number of PoE transcripts generated by a probabilistic polynomial-time prover and verifier. By Lemma 3 under the Adaptive Root Assumption, the probability that a polynomial time adversary can break the soundness of PoE, *i.e.*, convince a verifier on an instance $(\mathsf{C}_R, \mathsf{C} - \mathsf{C}_L, q^{(2^{\mu-1})}) \notin \mathcal{R}_{\mathsf{PoE}}$, is negligible. Consequently, the probability that the adversary can break PoE on *any* of the polynomial number of executions of PoE is still negligible.

This means that with overwhelming probability all transcripts are equivalent to having the verifier directly check $(\mathsf{C}_R, \mathsf{C} - \mathsf{C}_L, q^{(2^{\mu-1})}) \in \mathcal{R}_{\mathsf{PoE}}$. By assumption, the witness-candidate f that E' outputs is a valid witness if the transcript tr' that E' also outputs is accepting. The addition of honest PoE transcripts to tr' preserves the transcript's validity. So tr is an accepting transcript for Eval if and only if tr' is an accepting transcript for Eval'. Therefore, E' outputs a valid witness $f(X)$ whenever E outputs a valid witness. This suffices to show that Eval has witness-extended emulation if Eval' has, and if the Adaptive Root Assumption holds for $GGen$. □

Theorem 1. *Let* $\mathsf{CSZ}_{\mu,\lambda} = 8\mu^2 + \log_2(2\mu)\lambda$. *Let* $\mathsf{EBL}_{\mu,\lambda} = \lambda \cdot \mu$ *and* $\mathsf{CB}_{p,\mu,\lambda} = \lambda \cdot \mu + \log_2 p$. *Let* com *be the DARK commitment scheme as described in Lemma 10. There exists a pair of predicates* ϕ *such that the* μ*-round DARK polynomial commitment evaluation protocol* Eval' *with* λ*-bit challenges, a group of unknown order GGen, and* $\log q \geq 4(\lambda+1+\mathsf{CSZ}_{\mu,\lambda}) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} + 1$ *is* $(2^{(\mu)}, \frac{3\mu}{2^\lambda}, \mathsf{com}, \phi)$*-almost-special-sound* .[14]

As a corollary, under the adaptive root assumption for GGen, the DARK polynomial commitment scheme with the same parameters has witness-extended-emulation (Definition 7).

Remark 5. $\mathsf{CSZ}_{\mu,\lambda}$ *is derived from the Multilinear Composite Schwartz Zippel Lemma (lemma 7).* $\mathsf{EBL}_{\mu,\lambda}$ *is derived from the Evaluation Bound Lemma (lemma 6) and* CB *refers to the final round check bound in the DARK protocol. We can also substitute any value for* $\mathsf{CSZ}_{\mu,\lambda}$ *using the table of concrete bounds in Lemma 8 for fixed* 120*-bit security in place of the analytical bound from Lemma 7).*

Proof. For any $\beta_{\mathsf{n}}, \beta_{\mathsf{d}} \in \mathbb{R}$, let $\mathcal{M}(\beta_{\mathsf{n}}, \beta_{\mathsf{d}}) = \{f/N \in \mathbb{Q}[X] : \gcd(f, N) = 1, ||f||_\infty \leq \beta_{\mathsf{n}}, |N| \leq \beta_{\mathsf{d}}\}$. In the relation $R(\mathsf{x}, \mathsf{w})$ for the DARK evaluation protocol, the input $\mathsf{x} = (C, \vec{z} = (z_1, \ldots, z_\mu), y)$ consists of a DARK commitment C, an evaluation point $\vec{z} \in \mathbb{Z}^\mu$ and a claimed evaluation $y \in \mathbb{Z}$, while the witness w is an opening of C to a rational μ-linear polynomial h such that $h(\vec{z}) = y \bmod p$. For any parameters $\beta_{\mathsf{n}}, \beta_{\mathsf{d}}$ such that $\beta_{\mathsf{n}} \cdot \beta_{\mathsf{d}} \leq \frac{q}{2}$ this is binding to rational polynomials in $\mathcal{M}(\beta_{\mathsf{n}}, \beta_{\mathsf{d}})$. Setup parameters include p and q.

For reasons that will become clear we will set:

$$\log q = 4(\lambda + 1 + \mathsf{CSZ}_{\mu,\lambda}) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} + 1$$

$$\log_2 \beta_{\mathsf{n}} = \frac{1}{2}(\mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} + \log_2 q - 1) \qquad \log_2 \beta_{\mathsf{d}} = \frac{1}{2}(\log_2 q - 1 - \mathsf{EBL}_{\mu,\lambda} - \mathsf{CB}_{p,\mu,\lambda})$$

so that $\log_2(\beta_{\mathsf{n}} \cdot \beta_{\mathsf{d}}) = \log_2 q - 1$ as desired.

[14]The $\mathsf{CSZ}_{\mu,\lambda}$ value can be replaced with values from the table in Lemma 8

We begin by defining $\mathsf{com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ and predicates ϕ_a and ϕ_b for DARK special-soundness (Definition 9).

- The commitment setup $\mathsf{com.Setup}(\lambda, \beta_\mathsf{n}, \beta_\mathsf{d})$ runs the setup procedure for the DARK commitment scheme, which samples a group $\mathbb{G} \leftarrow \mathsf{GGen}(\lambda)$, a generator $\mathsf{G} \leftarrow \mathbb{G}$, sets $q = 2\beta_\mathsf{n}\beta_\mathsf{d}$, and returns $\mathsf{pp} = (\mathbb{G}, \mathsf{G}, q)$.

- The commitments of com are pairs $\mathcal{C} = ((C_L, y_L), (C_R, y_R)$ where C_L, C_R are DARK commitments and $y_L, y_R \in \mathbb{Q}$. Recall that $(f, N) \in \mathbb{Z}[X_1, \ldots, X_\mu] \times \mathbb{Z}$ is an opening of a DARK commitment C to the μ-linear rational polynomial $h = f/N$ provided that $f(q, \ldots, q^{\mu-1}) \cdot \mathsf{G} = N \cdot C$, where q is a parameter of the DARK commitment scheme. This is binding to rational polynomials in the set $\mathcal{M}(\beta_\mathsf{n}, \beta_\mathsf{d})$.

  We define a valid opening of $\mathcal{C} = ((C_L, y_L), (C_R, y_R))$ to a rational μ-linear polynomial $h \in \mathbb{Q}[X_1, \ldots, X_\mu]$ as a pair $(f, N) \in \mathbb{Z}[X_1, \ldots, X_\mu] \times \mathbb{Z}$ where $f = f_L + X_\mu f_R$ for $f_L, f_R \in \mathbb{Z}[X_1, \ldots, X_{\mu-1}]$ such that (f_L, N) and (f_R, N) are valid openings of the DARK commitments C_L and C_R respectively, provided that $N \cdot h = f$, $f_L(\vec{z}) = N \cdot y_L \bmod p$, $f_R(\vec{z}) = N \cdot y_R \bmod p$. This also implies that (f, N) is a valid opening of the homomorphically derived DARK commitment $C = C_L + q^{2^{\mu-1}} C_R$ to h and $h(z_1, \ldots, z_\mu) = y_L + z_\mu y_R \bmod p$, i.e. $N \cdot C = f(q, \ldots, q^{2^{\mu-1}}) \cdot \mathsf{G}$ and $h \in \mathcal{M}(\beta_\mathsf{n}, \beta_\mathsf{d})$.

  Additionally, a rational number is also considered a valid (trivial) commitment to itself. In the DARK protocol the prover's messages are commitments of the first kind for all but its last message, which is a single integer.

- We define the numerator bounds $B_0 \geq \cdots \geq B_\mu \in \mathbb{N}$ and denominator bounds $D_0 \geq \cdots \geq D_\mu$ such that $\log B_\mu = \mathsf{CB}_{p,\mu,\lambda}$ is the verification bound on the prover's final integer message in the DARK protocol, $D_\mu = 1$ (i.e., prover's final message is an integer), $\log D_i = \mathsf{CSZ}_{\mu-i,\lambda}$ and $\log B_i = \mathsf{CSZ}_{\mu-i,\lambda} + \mathsf{EBL}_{\mu-i,\lambda} + \mathsf{CB}_{p,\mu,\lambda}$.

  For $i \in [\mu - 1]$ and any $h \in \mathbb{Q}[X_1, \ldots, X_{\mu-i}]$, we define $\phi_a(i, h) = 1$ if and only if h is $\mu - i$ linear and $h \in \mathcal{M}(B_i, D_i)$. In particular, $\phi_a(\mu, h) = 1$ iff $h \in \mathbb{Z}$ and $|h| \leq B_\mu$.

- For $i \in [\mu - 1]$ and an opening $h \in \mathbb{Q}[X_1, \ldots X_{\mu-i}]$ define $\phi_b(i,h) = 1$ if and only if h is $\mu - i$ linear and $h \in \mathcal{M}(2^{\lambda+1} B_i D_i, 2^{\lambda+1} D_i^2)$.

By setting q sufficiently large so that $\log q \geq 4(\lambda + 1 + \mathsf{CSZ}_{\mu,\lambda}) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} + 1$, for any $i \in [\mu]$, $\phi_b(i,h) = 1$ implies that $2^{\lambda+1} \cdot h \in \mathcal{M}(\beta_{\mathsf{n}}, \beta_{\mathsf{d}})$. To see this, the log of the numerator bound on $2^{\lambda+1}h$ is:

$$\begin{aligned} 2(\lambda+1) + \log(B_0 D_0) &= 2(\lambda + \mathsf{CSZ}_{\mu,\lambda} + 1) + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda} \\ &\leq \frac{1}{2}(\log q - 1 + \mathsf{EBL}_{\mu,\lambda} + \mathsf{CB}_{p,\mu,\lambda}) = \log \mathsf{n} \end{aligned}$$

And the log of the denominator bound on $2^{\lambda+1}h$ is:

$$2(\lambda + 1 + \mathsf{CSZ}_{\mu,\lambda}) \leq \frac{1}{2}(\log q - 1 - \mathsf{EBL}_{\mu,\lambda} - \mathsf{CB}_{p,\mu,\lambda}) = \log \beta_{\mathsf{d}}$$

Next, we define the algorithms $\mathsf{Extract}$ and Extend.

- $\mathsf{Extract}(i, \nu, C_\nu, \mathsf{openSubtree})$ for $i < \mu$ operates as follows. Let $\mathcal{C}_\nu = (C_L, C_R)$. The node ν has two children. Let α_1 denote the label on the edge to the first child and α_2 the label on the edge to the second child. For $j \in \{1,2\}$, let $\mathcal{C}_j = ((C_{j,L}, y_{j,L}), (C_{j,R}, y_{j,R}))$ denote the commitment label of the ith child with openings (f_j, N_j) to $h_j = f_j/N_j$ where for $i < \mu - 1$ $f_j = f_{j,L} + X_{\mu-i-1} f_{j,R}$, and $h_j(z) = y_{j,L} + z_{\mu-i-1} \cdot y_{j,R} \bmod p$. Set $N = (\alpha_2 - \alpha_1) N_1 N_2$, $f_L = \alpha_2 N_2 f_1 - \alpha_1 N_1 f_2$, and $f_R = N_1 f_2 - N_2 f_1$. Set $f = f_L + X_{\mu-i} \cdot f_R$. Return (f, N) as the opening for C_ν to $h = f/N$.

- $\mathsf{Extract}(0$, (C, z, y), $\mathsf{openTree})$ simply returns the opening (f, N) for the root level commitment $((C_L, y_L), (C_R, y_R))$ to $h = f/N$, which satisfies $N \cdot C = f(q) \cdot \mathsf{G}$ and $h(z) = y$, since in a valid transcript tree $C = C_L + q^{2^{\mu-1}} C_R$ and $y_L + z_\mu \cdot y_R = y$. Furthermore, if $\phi_a(0,h) = 1$ then $h \in \mathcal{M}(B_0, D_0) \subset \mathcal{M}(\beta_{\mathsf{n}}, \beta_{\mathsf{d}})$, and hence $\mathsf{w} = (f, N)$ is a witness for $\mathsf{x} = (C, z, y)$ such that $R(\mathsf{x}, \mathsf{w}) = 1$.

- $\mathsf{Extend}(i, h, \alpha_{i+1}, ..., \alpha_\mu)$ on $h \in \mathbb{Q}[X_1, \ldots, X_{\mu-i}]$ returns $\perp$ if h is not $\mu - i$ linear, and otherwise sets $h_i := h$ and runs the following iterative algorithm: for $j = i$ to $\mu - 1$ set $h_{j+1} := h_{j,L} + \alpha_{j+1} \cdot h_{j,R}$ where, treating each h_j as a $\mu - j$ linear polynomial (padding with zero coefficients), $h_{j,L}$ and $h_{j,R}$ are each $\mu - j - 1$ linear consisting of the left/right coefficients (i.e. the constant and linear part of $X_{\mu-j}$) of h_j, i.e. $h_j = h_{j,L} + X_{\mu-j} \cdot h_{j,R}$; return $h_i, ..., h_\mu$.

Notes: The runtime is $O(\lambda \cdot 2^{\mu-i})$.

If $h_i = h_{i,L} + X_{\mu-i} \cdot h_{i,R} \in \mathbb{Z}[X_1, \ldots, X_{\mu-i}]$ is the prover's committed polynomial in the ith round of the (honest) interactive DARK protocol and $\alpha_{i+1}, ..., \alpha_\mu$ are the last $\mu - i$ round challenges then h_μ is the last prover's message sent to the verifier in the interactive DARK protocol. Then $h_\mu = h_i(\alpha_\mu, \ldots, \alpha_{i+1})$

- $\mathsf{Break}(i, h, \alpha_1, ..., \alpha_\mu, C_0, ..., C_\mu, (f_i, N_i), ..., (f_\mu, N_\mu))$[15] first runs $\mathsf{Extend}(i, h, \alpha_i, .., \alpha_\mu)$, which returns rational polynomials $h'_{i+1}, ..., h'_\mu$. If $\forall_{j \geq i}\ h'_j = f_j/N_j$ then it outputs $\perp$. Otherwise, let $j \in [i, \mu)$ be the *first* index where $h'_{j+1} \neq f_{j+1}/N_{j+1}$ and output $(N_j, f_{j,L} + \alpha_{j+1} \cdot f_{j,R})$, where $f_{j,L}$ and $f_{j,R}$ are the left/right halves of f_j, as the attempted opening for C_{j+1}.

Subclaim 1. *For $i \in [\mu-1]$, if all openings of commitments on children of ν in $\mathsf{openSubtree}$ satisfy $\phi_a(i+1, *) = 1$ then the tuple (f_L, f_R, N) returned by $\mathsf{Extract}(i, \nu, \mathcal{C}_\nu, \mathsf{openSubtree})$ is a valid opening for $\mathcal{C}_\nu$ to a rational polynomial h that satisfies $\phi_b(i, h) = 1$*

Proof. We will show why this is a correct opening for $\mathcal{C}_\nu$ and bound its norm. Based on the properties of a valid transcript tree, $\forall_{j \in \{1,2\}} C_L + \alpha_j C_R = C_{j,L} + q^{2^{\mu-i-2}} C_{j,R}$ and $y_L + \alpha_j y_R = y_{j,L} + z_{\mu-i12} y_{j,R}$. Furthermore, $\forall_{j \in \{1,2\}} N_j \cdot (C_L + \alpha_j C_R) = f_j(q)$ and $h_j(z) = N_j^{-1} f_j(z) = y_L + \alpha_j y_R \bmod p$ by the assumption that (f_j, N_j) are valid openings to the two children commitments $\mathcal{C}_j$. Let $L_q(\cdot) : \mathbb{Z}[X]^2 \to \mathbb{Z}^2$ denote the linear operator corresponding to component-wise evaluation of each polynomial at $\vec{q}$. Using linear algebra, the following

[15]For simplicity, we omit the messages $h_i = f_i/N_i$ from the inputs because it can be computed from the opening (f_i, N_i).

holds true if $\alpha_1 \neq \alpha_2$:

$$\begin{bmatrix} N_1 & 0 \\ 0 & N_2 \end{bmatrix} \begin{bmatrix} 1 & \alpha_1 \\ 1 & \alpha_2 \end{bmatrix} \begin{bmatrix} C_L \\ C_R \end{bmatrix} = L_q\left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) \cdot \mathsf{G}$$
$$\Downarrow$$
$$(\alpha_2 - \alpha_1) N_1 N_2 \begin{bmatrix} C_L \\ C_R \end{bmatrix} = L_q\left(\begin{bmatrix} \alpha_2 N_2 & -\alpha_2 N_1 \\ -N_2 & N_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) \cdot \mathsf{G}$$

and

$$\begin{bmatrix} N_1 & 0 \\ 0 & N_2 \end{bmatrix} \begin{bmatrix} 1 & \alpha_1 \\ 1 & \alpha_2 \end{bmatrix} \begin{bmatrix} y_L \\ y_R \end{bmatrix} = L_z\left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) \bmod p$$
$$\Downarrow$$
$$(\alpha_2 - \alpha_1) N_1 N_2 \begin{bmatrix} y_L \\ y_R \end{bmatrix} = L_z\left(\begin{bmatrix} \alpha_2 N_2 & -\alpha_2 N_1 \\ -N_2 & N_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}\right) \bmod p$$

This shows that $N \cdot C_L = f_L(q) \cdot \mathsf{G}$, $N \cdot C_R = f_R(\vec{q}) \cdot \mathsf{G}$, $N \cdot y_L = f_L(\vec{z}) \bmod p$, and $N \cdot y_R = f_R(\vec{z}) \bmod p$. Furthermore, if f_1 and f_2 are $\mu - i$ linear, then so are f_L and f_R.

If $\forall_j ||f_j||_\infty \leq B_i$ and $\forall_j |N_j| \leq B_i$, then $|N| \leq 2^{\lambda+1} D_i^2$ and $||f||_\infty \leq 2^{\lambda+1} B_i D_i$. Thus, if the openings of the children at level $i+1$ satisfy $\phi_a(i+1, f_j/N_j) = 1$ for $j \in \{1, 2\}$, then $\phi_b(i, f/N) = 1$. $\square$

Subclaim 2. *For any $i \in [\mu]$ and $h \in \mathbb{Q}[X_1, \ldots, X_{\mu-i}]$, if $\phi_a(i, h) = 0$ then the probability over uniform i.i.d. $\alpha_{i+1}, ..., \alpha_\mu$ that h_μ returned by $\mathsf{Extend}(i, h, \alpha_{i+1}, ..., \alpha_\mu)$ satisfies $\phi_a(\mu, h_\mu) = 1$ is at most $\frac{3(\mu-i)}{2^\lambda}$.*

Let $f/N = h$ for $\gcd(f, N) = 1$ denote the reduced form of $h \in \mathbb{Q}[X_1, \ldots, X_{\mu-i}]$. If $\phi_a(i, h) = 0$ then either $N > D_i$ or $||f||_\infty > B_i$ while $\phi_a(\mu, h_\mu) = 1$ implies $h_\mu \in \mathbb{Z}$ and $|h_\mu| \leq B_\mu$.

Let $\mu' = \mu - i$. Observe that $h_\mu = \frac{1}{N} \cdot f(\alpha_\mu, ..., \alpha_{i+1})$.

Case 1 $N > D_i$:

If $|N| > D_i$ then since $\log D_i = \mathsf{CSZ}_{\mu-i}$, the probability that $h_\mu \in \mathbb{Z}$ is:

$$\mathbb{P}_{(\alpha_{i+1},\dots,\alpha_\mu)\leftarrow[0,2^\lambda)^{\mu-i}}[f(\alpha_\mu,\dots,\alpha_{i+1}) \equiv 0 \bmod N] \leq \frac{\mu-i+1}{2^\lambda}$$

by the Multilinear Composite Schwartz-Zippel Lemma (Lemma 7).

Case 2 $N \leq D_i \ \wedge \ ||f||_\infty > B_i$:

In this case if $|h_\mu| \leq B_\mu$ then $|f(\alpha_\mu,\dots,\alpha_{i+1})| \leq N \cdot B_\mu \leq \mathsf{CSZ}_{\mu-i,\lambda} \cdot B_\mu$. Furthermore, the fact that $||f||_\infty > B_i$ and $\log B_i = \mathsf{CSZ}_{\mu-i,\lambda} + \mathsf{EBL}_{\mu-i,\lambda} + \log B_\mu$ imply:

$$\log(\mathsf{CSZ}_{\mu-i,\lambda} \cdot B_\mu) \leq \log B_i - \mathsf{EBL}_{\mu-i,\lambda} < \log ||f||_\infty - \mathsf{EBL}_{\mu-i,\lambda}$$

Hence by the Evaluation Bound Lemma (Lemma 6):

$$\mathbb{P}[h_\mu \leq B_\mu] \leq \mathbb{P}[|f(\alpha_\mu,\dots,\alpha_{i+1})| \leq D_i \cdot B_\mu] \leq \mathbb{P}[|f(\alpha_\mu,\dots,\alpha_{i+1})| \leq \frac{1}{2^{\mathsf{EBL}_{\mu-i,\lambda}}} \cdot ||f||_\infty] \leq \frac{3(\mu-i)}{2^\lambda}$$

Together these imply that if f is $\mu-i$ linear but $\phi_a(i, f/N) = 0$ then, since either $|N| > B_i$ or $||f||_\infty > B_i$, the probability over the random challenges that the final element h_μ of the list returned by Extend satisfies $\phi_a(\mu, h_\mu) = 1$ is negligible.

□

Subclaim 3. *For any $i \in [\mu-1]$, given a valid (accepting) transcript with commitments $(\mathcal{C}_0,\dots,\mathcal{C}_\mu)$, round challenges $(\alpha_1,\dots,\alpha_\mu)$, and openings $(o_i,\dots,o_\mu)$ of the last $\mu-i+1$ commitments to rational polynomials $(h_i,,\dots,h_\mu)$, where $\phi_b(i,h_i) = 1$ and $\phi_a(j,h_j) = 1$ for all $j \in [i+1,\mu]$, then either $\mathsf{Extend}(i,h_i,\alpha_{i+1},\dots,\alpha_\mu)$ returns $h_{i+1},\dots,h_\mu$ or $\mathsf{Break}(i,h_i,(h_i,o_i),\dots,(h_\mu,o_\mu))$ returns for some $j \geq i$ a valid opening of $\mathcal{C}_j$ to $h'_j \neq h_j$.*

Proof. Let $(h'_{i+1},\dots,h'_\mu)$ denote the output of $\mathsf{Extend}(i,h_i,\alpha_{i+1},\dots,\alpha_\mu)$ and suppose it is not equal to $(h_{i+1},\dots,h_\mu)$. Let $j \in [i,\mu)$ denote the *first* index for which $h_{j+1} \neq h'_{j+1}$. This means that $h_j = h'_j$ and thus $h'_{j+1} = h_{j,L} + \alpha_{j+1} \cdot h_{j,R}$ where $h_{j,L}$ and $h_{j,R}$ are the left/right halves of h_j. Additionally, $\phi_b(j,h_j) = 1$ implies $h'_{j+1} \in \mathcal{M}(2^{2\lambda+2}B_jD_j, 2^{\lambda+1}B_j^2) \subseteq$

$\mathcal{M}(\beta_{\mathsf{n}}, \beta_{\mathsf{d}})$.

(Note that for $j \geq i+1$, the condition $\phi_a(j, h_j) = 1$ also implies $\phi_b(j, h_j) = 1$).

Let $\mathcal{C}_{j+1} = ((C_{j+1,L}, y_{j+1,L}), (C_{j+1,R}, y_{j+1,R}))$ and $\mathcal{C}_j = ((C_{j,L}, y_{j,L}), (C_{j,R}, y_{j,R}))$. Let $o_j = (f_j, N_j)$ denote the opening of $\mathcal{C}_j$ to $h_j = \frac{f_j}{N_j}$ where $f_j = f_{j,L} + X_{\mu-j} f_{j,R}$. Validity of the opening implies $N_j \cdot C_{j,L} = f_{j,L}(q) \cdot \mathsf{G}$ and $N_j \cdot C_{j,R} = f_{j,R}(q) \cdot \mathsf{G}$. Furthermore, in a valid transcript:

$$C_{j+1} = C_{j,L} + \alpha_{j+1} \cdot C_{j,R} = C_{j+1,L} + q^{2^{\mu-j}} C_{j+1,R}$$

Thus, $(N_j, f_{j,L} + \alpha_{j+1} \cdot f_{j,R})$ is a valid opening of C_{j+1} to h'_{j+1} as $N_j \cdot C_{j+1} = (f_{j,L}(q) + \alpha_{j+1} \cdot f_{j,R}(q)) \cdot \mathsf{G}$.

Witness-extended emulation By corollary 1 and if the DARK commitment is binding then this shows that Eval' has witness extended emulation for the relation $\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp})$ for polynomials in $\mathbb{F}_p$. The binding property of DARK depends on the random order assumption which is implied by the adaptive root assumption (Lemmas 2 and 10). Further by Lemma 15 this implies that Eval has witness-extended emulation for the same relation under the adaptive root assumption.

□

2.9 Fiat-Shamir Transform of Almost-Special-Sound Protocols

Recently, [165, 13] showed that for μ-round special sound protocols the non-interactive Fiat-Shamir transform of these protocols only suffers a security loss that is linear in the number of queries the adversary makes to the random oracle. These proofs do not directly apply to almost-special-sound protocols. In particular, in a non-interactive protocol, we cannot guarantee that the challenges on $\mathsf{lpath}(\nu)$ are mutually independent. An adversary might grind each challenge and pick one depending on the previous challenges. Concretely, for

DARK the composite Schwartz-Zippel lemma analyzes the probability that $f(x_1, \dots, x_\mu) \equiv 0 \bmod N$ for *independently* sampled $x_1, \dots, x_\mu$. If the adversary, can grind challenges, i.e. try different challenges per prover message, then it can for each challenge x_i ensure that $f(x_1, \dots, x_i, X_{i+1}, \dots, X_\mu) \equiv 0 \bmod N_i$ where N_i is a factor of N of size roughly $N^{\frac{1}{\mu}}$. The proof of theorem 1 relies on the fact that $\mathbb{P}_{(\alpha_{i+1},\dots,\alpha_\mu)\leftarrow[0,2^\lambda)^{\mu-i}}[g(\alpha_{i+1}, \dots, \alpha_\mu) \equiv 0 \bmod N] = \delta$ is negligible for sufficiently large N. Analyzing a grinding adversary of the non-interactive protocol that makes at most T queries to the random oracle corresponds to analyzing the probability that for any set of T values:

$$\mathbb{P}_{S=\{\alpha_{i,j}\}_{i\in[\mu],j\in[T]})\leftarrow[0,2^\lambda)^{\mu\cdot T}} \exists (j_1, \dots, j_\mu) \text{ s.t. } [g(\alpha_{1,j_1}, \dots, \alpha_{\mu,j_\mu}) \equiv 0 \bmod N]$$

which is less than $T^\mu \cdot \delta$ by the union bound.

However, what if it is impossible for the adversary to actually grind the prover message. We know that in almost-special-sound protocols the prover message is a commitment. If that commitment is unique, i.e. for a given opening there exists only one possible commitment, then a grinding adversary couldn't choose a different commitment. We capture this notion formally and show that even if the commitment is only *computationally unique* the Fiat-Shamir transform of these almost-special-sound protocols is secure. The DARK protocol has precisely this property.

Definition 12 (Random Oracle)**.** *In our version of the random oracle model, all random oracle algorithms have black-box access to a shared random function $\mathcal{H} : \mathcal{M}^{\leq u} \to \mathcal{X}$ where $\mathcal{M}^{\leq u}$ consists of all vectors $(m_1, \dots, m_i) \in \mathcal{M}^i$ for each $i \leq u$. $\mathcal{H}$ assigns to each of the $\frac{|\mathcal{M}|^{u+1}-1}{|\mathcal{M}|-1}$ unique elements of $\mathcal{M}^{\leq u}$ an output independently and uniformly distributed in $\mathcal{X}$. Random oracles may be assigned indices from a set $\mathcal{I}$, where for any distinct $i, j \in \mathcal{I}$ the oracles $\mathcal{H}_i$ and $\mathcal{H}_j$ are independently distributed random functions. For any $k < u$ and fixed $\mathbf{m} = (m_1, \dots, m_k)$ we will use the notation $\mathcal{H}_{\mathbf{m}} : \mathcal{M}^{\leq u-k} \to \mathcal{X}$ where $\mathcal{H}_{\mathbf{m}}(m'_1, \dots, m'_i) = \mathcal{H}(\mathbf{m}, m'_1, \dots, m'_i)$ for any $i \leq u - k$. This is equivalent to a random oracle family indexed by $\mathcal{M}^k$.*

A *Q-query random oracle algorithm* is an algorithm that makes at most Q queries to the random oracle.

Definition 13 (Non-interactive Proof of Knowledge in RO). *An non-interactive protocol $\Pi = (\mathcal{P}, \mathcal{V})$ between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ in the random oracle model (i.e., with shared oracle access to the random function $\mathcal{H}$) is a proof of knowledge for a relation $\mathcal{R}$ with knowledge error $\delta : \mathbb{N}^2 \to [0,1]$ if the following properties hold:*

- *Perfect Completeness: for all $(x, w) \in \mathcal{R}$*

$$\mathbb{P}\left[\mathcal{V}^{\mathcal{H}}(x, \pi) = 1 : \pi \leftarrow \mathcal{P}^{\mathcal{H}}(x, w)\right] = 1$$

- *δ-Knowledge Soundness: There exists a polynomial $\mathsf{poly}(\cdot)$ and a probabilistic oracle machine $\mathcal{E}$ called the* extractor *such that given oracle access to any Q-query random oracle algorithm $\mathcal{A}$ and any input $x \in \mathcal{L}_R$ the following holds*[16]*: if*

$$\mathbb{P}\left[\mathcal{V}^{\mathcal{H}}(x, \pi) = 1 : \pi \leftarrow \mathcal{A}^{\mathcal{H}}(x)\right] = \epsilon(x)$$

  then $\mathcal{E}^{\mathcal{A}}(x)$ outputs w such that $(x, w) \in \mathcal{R}$ in an expected $\frac{\mathsf{poly}(|x|)}{\epsilon(x) - \delta(|x|, Q)}$ number of steps. In the course of running with black-box access to $\mathcal{A}$, $\mathcal{E}^{\mathcal{A}}(x)$ implements the random oracle for $\mathcal{A}$, i.e. it intercepts all queries that $\mathcal{A}$ makes to $\mathcal{H}$ and simulates the response.

Π is called "knowledge sound" or a "proof of knowledge" for $\mathcal{R}$ if for all Q polynomial in $|x|$ the knowledge error $\delta(|x|, Q)$ is a negligible function of $|x|$.

Definition 14 (Non-interactive argument of knowledge in SRS/RO). *A non-interactive proof system $\Pi = (\mathsf{Setup}, \mathcal{P}^{\mathcal{H}}, \mathcal{V}^{\mathcal{H}})$ with setup procedure $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ in the random oracle model, where $\mathcal{P}^{\mathcal{H}}$ and $\mathcal{V}^{\mathcal{H}}$ are given shared access to both the random oracle $\mathcal{H}$ and the parameters pp (where $|\mathsf{pp}| \geq \lambda$), is an argument of knowledge for relation $\mathcal{R}$ with knowledge error $\mathsf{err} : \mathbb{N}^2 \to [0,1]$ if there exists a polynomial time extractor $\mathcal{E}$ such that for any non-uniform polynomial time adversary $\mathcal{A}$ and deterministic Q-query polynomial time*

[16]The algorithm $\mathcal{A}$ may explicitly hardcode a witness or may not have one, so no witness is given to $\mathcal{A}$ as input.

prover P^* the following holds:[17]

$$\mathbb{P}\left[\begin{array}{c} (x,w)\in\mathcal{R} \text{ and} \\ \mathcal{V}^{\mathcal{H}}(\mathsf{pp},x,\pi)=1 \end{array} : \begin{array}{c} \mathsf{pp}\leftarrow \mathsf{Setup}(\lambda) \\ (x,\mathsf{st})\leftarrow\mathcal{A}(\mathsf{pp}) \\ \pi\leftarrow P^*(\mathsf{st}) \\ w\leftarrow\mathcal{E}^{P^*(\mathsf{st})}(\mathsf{pp},x) \end{array}\right] \geq \mathbb{P}\left[\mathcal{V}^{\mathcal{H}}(\mathsf{pp},x,\pi)=1 : \begin{array}{c} \mathsf{pp}\leftarrow \mathsf{Setup}(\lambda) \\ (x,\mathsf{st})\leftarrow\mathcal{A}(\mathsf{pp}) \\ \pi\leftarrow P^*(\mathsf{st}) \end{array}\right] - \mathsf{err}(\lambda,Q)$$

Definition 15 (FS Transform). *For any μ-round public-coin interactive proof Π the FS transform $\Pi_{FS}^{\mathcal{H}}$ of Π with respect to the random oracle $\mathcal{H}$ is a non-interactive proof in the random oracle model which on public input x simulates the interactive protocol Π by replacing each ith round public-coin challenge, for $i\in[1,\mu]$, with $\mathcal{H}(\mathsf{x},m_1,...,m_i)$, where $m_1,...,m_i$ denote the first i prover messages.*

Lemma 16 (FS for special-sound multiround protocols [13, 165]). *For any μ-round interactive proof $\Pi=(\mathcal{P},\mathcal{V})$ for relation $\mathcal{R}$ and its FS transform $\Pi_{FS}=(\mathcal{P}^{\mathcal{H}},\mathcal{V}^{\mathcal{H}})$ with random oracle $\mathcal{H}$, there exists a random oracle algorithm Tree which given black-box access to any Q-query deterministic prover algorithm $\mathcal{P}^*$, input $x\in\mathcal{L}_R$, and $k\in\mathbb{N}$ makes at most $Q+\mu$ queries to $\mathcal{H}$ and returns a k-ary forking transcript tree for Π in expected time $k^\mu+Q\cdot(k^\mu-1)$ and succeeds with probability $\frac{\epsilon(x)-(Q+1)\cdot\kappa}{(1-\kappa)}$ where $\kappa=1-(1-\frac{k-1}{2^\lambda})^\mu$ and $\epsilon(x)$ is the probability (over $\mathcal{H}$) that $\mathcal{P}^*$ outputs a non-interactive proof for x that $\mathcal{V}^{\mathcal{H}}$ accepts. Moreover, the transcript tree satisfies additional properties:*

- *Every root-to-leaf labelled path (i.e., transcript included in the tree) matches the output of $\mathcal{P}^{*\mathcal{H}'}(x)$ with a partially fresh random oracle $\mathcal{H}'$, and thus has the format of a valid Π_{FS} proof with respect to $\mathcal{H}'$.*
- *For any node ν, the labels L_{S_ν} on the subtree S_ν are generated by a $(Q+\mu)$-query*

[17]This definition says that there is overwhelming intersection between the event where the adversary generates an input x and corresponding proof π that convinces the verifier, and the event where the extractor succeeds in obtaining a witness from the input x generated by the adversary. This not only ensures that extraction succeeds with close to the same probability of the adversary's success over randomly sampled parameters, but also excludes the pathological case that both the adversary and extractor succeed with noticeable probability on disjoint sets of inputs. This definition is also equivalent to fixing the transcript distinguisher in the definition of witness-extended emulation (Definition 7) to be the verifier decision algorithm. In WEE the transcript distinguisher could be arbitrary, which is a stronger property important for simulation analysis.

random oracle algorithm independently from all labels L_{ν^*}, i.e. labels computed on lower indexed nodes that do not belong to S_ν. In particular, if ν is the kth child of ω then L_{S_ν} is independent of the labels on the $(k-1)$-ary left subtree of ω.

Lemma 16 immediately implies that the FS transform of any $k^{(\mu)}$-special sound protocol for $\mathcal{R}$ is knowledge sound in the RO model. However, we need to work a bit harder to apply this lemma to almost-special-sound protocols. We will only be able to show computational knowledge soundness for protocols that are almost-special-sound with respect to a computationally-unique commitment scheme. The DARK proof system has this property.

Definition 16 (RO relation hardness)**.** *Let $\mathcal{R}^{\mathcal{H}}(\mathsf{pp})$ denote a family of relations parametrized by a random oracle $\mathcal{H}$ and setup $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ with security parameter λ where $|\mathsf{pp}| \geq \lambda$. $\mathcal{R}^{\mathcal{H}}(\mathsf{pp})$ is $(Q, \epsilon(\lambda))$-hard in the RO model if for any pair of polynomial time random oracle algorithms $\mathcal{A}_1, \mathcal{A}_2$ where $\mathcal{A}_1$ makes at most Q queries to a random oracle $\mathcal{H}'$ (possibly distinct from $\mathcal{H}$) and $\mathcal{A}_2$ makes at most Q queries to random oracle $\mathcal{H}$:*

$$\mathbb{P}\left[(\mathsf{x},\mathsf{w}) \in \mathcal{R}^{\mathcal{H}}(\mathsf{pp}) : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ \mathsf{x} \leftarrow \mathcal{A}_1^{\mathcal{H}'}(\mathsf{pp}) \\ \mathsf{w} \leftarrow \mathcal{A}_2^{\mathcal{H}}(\mathsf{pp},\mathsf{x}) \end{array}\right] \leq \epsilon(\lambda)$$

Definition 17 (Computationally Unique Commitments)**.** *A commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is computationally-unique if for any polynomial time adversary $\mathcal{A}$*

$$\Pr\left[b_0 = b_1 = 1 \wedge C_0 \neq C_1 \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (C_0, C_1, x, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, C_0, x, r_0) \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, C_1, x, r_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Lemma 17. *The DARK commitment scheme satisfies computational uniqueness (Definition 17) under the adaptive root assumption.*

Proof. Given two commitments to the same message, we will construct a known order element. This element can be used to break the adaptive root assumption. Concretely, assume there exists an adversary $\mathcal{A}_{\mathsf{CU}}$ that with non-negligible probability ϵ outputs C and C' as well as $h(X) = \frac{f(X)}{N}$ such that $N \cdot \mathsf{C} = f(q) \cdot \mathsf{G}$ and $N \cdot \mathsf{C}' = f(q) \cdot \mathsf{G}$. This implies that $N \cdot (\mathsf{C} - \mathsf{C}') = 0$, i.e. that N is a multiple of the order of $\mathsf{C} - \mathsf{C}'$, which by assumption is a non trivial group element. We can use this to construct an adversary $\mathcal{A}_{\mathsf{AR}} = (\mathcal{A}_1, \mathcal{A}_2)$ for the adaptive root assumption, where $\mathcal{A}_1$ outputs $\mathsf{W} = \mathsf{C} - \mathsf{C}'$ and $\mathcal{A}_2$ while $\gcd(N, \ell) \neq 1$ computes $N' \leftarrow N / \gcd(N, \ell^k)$ for $k = \lceil \log_\ell(N) \rceil$ and computes $r \leftarrow \ell^{-1} \bmod N'$ and outputs $\mathsf{U} \leftarrow \mathsf{W}^r$. If ℓ is co-prime with the order of $\mathbb{G}$ and thus W then $\mathsf{U}^\ell = \mathsf{W}^{r \cdot \ell}$ which equals W if N' is a multiple of the order of U. This is the case with overwhelming probability as N is a multiple of the order of U and ℓ divides the order of U with only negligible probability. Thus $\mathcal{A}_{\mathsf{AR}}$ succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. This is a contradiction and shows that Γ is computationally unique. □

Lemma 18. *For any μ and $Q = \mathsf{poly}(\lambda)$, the relation $\mathcal{R}^{\mathcal{H}}_{\mathsf{ext}}(\mathsf{pp}, \mu, \rho)$ defined with respect to any computationally-unique commitment scheme com, predicate $\rho : \mathcal{M}^2 \times \mathcal{X}^\mu \to \{0,1\}$, and $\mathsf{ext} : \mathcal{M} \times \mathcal{X}^{\leq \mu} \to \mathcal{M}$ such that $\forall i, m \in \mathcal{M}$:*

$$\mathbb{P}_{\alpha_1,\ldots,\alpha_\mu}[\rho_i(m, m_\mu, \alpha_1, ..., \alpha_\mu) = 1 : \mathsf{ext}(m, \alpha_1, ..., \alpha_\mu) = m_\mu] \leq \delta$$

and

$$\mathcal{R}^{\mathcal{H}}_{\mathsf{ext}}(\mathsf{pp}, \mu, \rho) = \left\{ (\mathsf{x} = (C, m, o), \mathsf{w} = (\mathsf{tr}, M)) : \begin{array}{c} \mathsf{tr} = (C_1, ..., C_\mu) \\ \forall_{i \in [\mu]}\ \alpha_i = \mathcal{H}(C, C_1, ..., C_{i-1}) \\ M = ((m_1, o_1), ..., (m_\mu, o_\mu)) \\ \mathsf{com.Open}(\mathsf{pp}, C, m, o) = 1 \\ \forall_{i \in [\mu]} \mathsf{com.Open}(\mathsf{pp}, C_i, m_i, o_i) = 1 \\ \forall_{i \in [\mu]} \mathsf{ext}(m, \alpha_1, ..., \alpha_i) = m_i \\ \rho(m, m_\mu, \alpha_1, ..., \alpha_\mu) = 1 \end{array} \right\}$$

is $(Q, \delta + \mathsf{negl}(\lambda))$-hard in the RO model.

Proof. Consider any pair of polynomial time random oracle algorithms $\mathsf{x} \leftarrow \mathcal{A}_1^{\mathcal{H}'}(\mathsf{pp})$ and $\mathsf{w} \leftarrow \mathcal{A}_2^{\mathcal{H}}(\mathsf{pp}, \mathsf{x})$ that for setup parameter λ make at most $Q = \mathsf{poly}(\lambda)$ queries to their respective oracles $\mathcal{H}'$ and $\mathcal{H}$. For fixed x let $T_{\mathcal{H}}(\mathsf{x}) = (T_1, ..., T_\mu)$ denote a random variable representing the output tr included in w *conditioned on* the event that w satisfies <u>at least</u> all validity criteria other than possibly the last, i.e. $\rho(m, m_\mu, \alpha_1, ..., \alpha_\mu) = 1$. Note that $T_{\mathcal{H}}(\mathsf{x})$ is also dependent on the randomness of the oracle $\mathcal{H}$. Presuming this event occurs with probability at least $\epsilon(\mathsf{x})$, we can repeat $\mathcal{A}_2^{\mathcal{H}}(\mathsf{pp}, \mathsf{x})$ on fresh internal randomness (not changing $\mathcal{H}$) in expectation $1/\epsilon(\mathsf{x})$ times until it outputs w satisfying this event with a particular assignment $\mathsf{tr} = (C_1, ..., C_\mu)$ to the random variable $T_{\mathcal{H}}(\mathsf{x})$. Suppose that for any $i \in [\mu]$ we then reprogrammed $\mathcal{H}$ to an oracle $\mathcal{H}^*$ by sampling new answers to any subset of the queries $\{q_j = (C, C_1, ..., C_j)\}_{i \leq j < \mu}$ but keeping all other queries consistent with $\mathcal{H}$. For $i = \mu$ we do not change the oracle and $\mathcal{H}^* = \mathcal{H}$. Define the random variable $T_{\mathcal{H}^*}(\mathsf{x})$ in the same way for the new oracle $\mathcal{H}^*$. Suppose further that repeating the same experiment with $\mathcal{A}_2^{\mathcal{H}^*}(\mathsf{pp}, \mathsf{x})$ were to return with probability greater than $\delta'(\mathsf{x})$ a vector $\mathsf{tr}' = (C'_1, ..., C'_\mu) \neq \mathsf{tr}$ where the first distinct index between tr' and tr is some $k \leq i$. For $i = \mu$ we just repeat the same experiment with $\mathcal{H}$. For all $i \in [\mu]$ let $\alpha_i = \mathcal{H}(C, C_1, ..., C_{i-1})$, let $m_i = \mathsf{ext}(m, \alpha_1, ..., \alpha_i)$, let $\alpha'_i = \mathcal{H}^*(C, C'_1, ..., C'_{i-1})$, and let $m'_i = \mathsf{ext}(m, \alpha'_1, ..., \alpha'_i)$. Since the oracle answers to queries C and $\{q_j = (C, C_1, ..., C_\ell)\}_{1 \leq \ell < k}$ have not changed and $(C_1, ..., C_{k-1}) = (C'_1, ..., C'_{k-1})$ it follows that $(\alpha_1, ..., \alpha_k) = (\alpha'_1, ..., \alpha'_k)$ and $(m_1, ..., m_k) = (m'_1, ..., m'_k)$. The result of these two experiments would thus include openings of the distinct commitments $C_k \neq C'_k$ to the same message $m_k = m'_k$. By computational uniqueness of the commitment scheme, for $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ and $\mathsf{x} \leftarrow \mathcal{A}_1^{\mathcal{H}'}(\mathsf{pp})$ either $\delta'(\mathsf{x})$ or $\epsilon(\mathsf{x})$ is negligible in λ, as we have shown that it is possible to construct and adversary using $\mathcal{A}_1$ and $\mathcal{A}_2$ that on input pp breaks the uniqueness of the commitment scheme (Definition 17) in expected time $(\frac{1}{\epsilon(\mathsf{x})} + \frac{1}{\delta'(\mathsf{x})}) \cdot \mathsf{poly}(\lambda)$.

We draw two conclusions from this. For $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ and $\mathsf{x} \leftarrow \mathcal{A}_1^{\mathcal{H}'}(\mathsf{pp})$, if $\mathcal{A}_2^{\mathcal{H}}(\mathsf{pp}, \mathsf{x})$ succeeds in returning w satisfying the aforementioned event (i.e., all criteria except the last predicate) with non-negligible probability then there is a unique vector $(C_1, ..., C_\mu)$ such that:

(a) $\mathbb{P}[T_{\mathcal{H}}(\mathsf{x}) = (C_1, ..., C_\mu)] \geq 1 - \mathsf{negl}(\lambda)$

(b) While this unique vector of high support may depend on $\mathcal{H}$, for all $i \in [\mu]$ the first i components $(C_1, ..., C_i)$ are independent of the answers to the values $\mathcal{H}(q)$ for $q \in \{(C, C_1, ..., C_j)\}_{i \leq j < \mu}$.

If (a) were false then running the experiment above for case $i = \mu$ would succeed with non-negligible probability δ' in returning a distinct assignment to $\mathcal{T}_{\mathcal{H}}(\mathsf{x})$, which contradicts the computational uniqueness of the commitment scheme as shown above. If (b) were false, then for some $i < \mu$ the experiment would succeed with non-negligible probability δ' in returning a distinct assignment to $\mathcal{T}_{\mathcal{H}}(\mathsf{x})$ where the first index of distinction is $k \leq i$, again contradicting the computational uniqueness of the commitment scheme as shown above.

Finally, (b) implies that $(\alpha_1, ..., \alpha_\mu)$ where $\alpha_i = \mathcal{H}(C, C_1, ..., C_i)$ is uniformly distributed and hence $\mathbb{P}[\rho(m, m_\mu, \alpha_1, ..., \alpha_\mu) = 1] \leq \delta$ as stated in the hypothesis. Thus, conditioned on the event that w satisfies at least all validity criteria except the predicate, then its first component is $\mathsf{tr} = (C_1, ..., C_\mu)$ with probability $1 - \mathsf{negl}(\lambda)$, in which case it fails the last criteria (i.e., the predicate) with probability $1 - \delta$. In conclusion, by a union bound it satisfies all criteria with probability at most $\delta + \mathsf{negl}(\lambda)$. □

Theorem 7. *If Π is a $(k^{(\mu)}, \delta, \mathsf{com}, \phi)$-almost-special-sound protocol for a relation $\mathcal{R}$ and a computationally-unique commitment scheme* com *(Definition 17) whose setup runs $\mathsf{pp} \leftarrow \mathsf{com.Setup}(\lambda)$ then its FS transform Π_{FS} is an argument of knowledge for $\mathcal{R}$ in the RO model (Definition 14) with knowledge error:*

$$\mathsf{err}(\lambda, Q) = \frac{(Q+1)\kappa}{1-\kappa} + 2\lambda(k^\mu + Q \cdot (k^\mu - 1)) \cdot \delta + \mathsf{negl}(\lambda)$$

where $\kappa = 1 - (1 - \frac{k}{2^\lambda})^\mu$.

Proof. Let $\mathcal{V}^{\mathcal{H}}$ denote the resulting verifier for Π_{FS}. We will construct an extractor $\mathcal{E}$ which is given black-box access to any *deterministic* Q-query prover algorithm $\mathcal{P}^*$, where Q is assumed to be polynomial in λ. $\mathcal{E}$ has the power to intercept and respond to the queries $\mathcal{P}^*$ makes to the random oracle, simulating (i.e., reprogramming) the oracle responses. On input

x and parameters pp, $\mathcal{E}$ first tests that $\mathcal{P}^*$ outputs a proof π such that $\mathcal{V}^{\mathcal{H}}(\mathsf{pp}, x, \pi) = 1$ and otherwise aborts. If this first step succeeds, then $\mathcal{E}$ continues by running the tree generation algorithm from Lemma 16 to generate a $(k+1)$-ary forking transcript tree. Given black-box access to a deterministic Q-query prover algorithm, this tree generation algorithm runs in expected polynomial time $k^\mu + Q \cdot (k^\mu - 1)$ succeeding with probability $\frac{\epsilon(x)-(Q+1)\cdot\kappa}{(1-\kappa)}$ where $\kappa = 1 - (1 - \frac{k}{2^\lambda})^\mu$ and $\epsilon(x)$ is the probability over the randomness of $\mathcal{H}$ that $\mathcal{P}^*$ outputs a non-interactive proof that $\mathcal{V}^{\mathcal{H}}$ accepts. $\mathcal{E}$ will repeat λ iterations of running this tree generation algorithm for $2(k^\mu + Q \cdot (k^\mu - 1))$ steps each time, and returns the first trial that succeeds. By Markov, this results in a new tree generation algorithm that runs in strict polynomial time $2\lambda(k^\mu + Q \cdot (k^\mu - 1))$ with negligible loss $2^{-\lambda}$ in its probability of success.

For any $(k+1)$-ary transcript tree, there is a polynomial time procedure (Definition 10) which as shown in Lemma 14 either:

(a) Extracts a witness w such that $(x, w) \in \mathcal{R}$

(b) Extracts a break to the binding of the commitment scheme, i.e. an element of $\mathcal{L}_{\mathsf{break}}(\mathsf{pp})$ defined in Theorem 6.

(c) Extracts an opening (m_ω, o_ω) for the commitment label on some node ω at some level i with rightmost child ν at level $i+1$ such that $\phi_b(i, m_\omega) = 1$, $\phi_a(i, m_\omega) = 0$, openings of all commitment labels on the leftmost path $\mathsf{lpath}(\nu)$ extending down from ν to messages equal to $\mathsf{Extend}(i, m, \alpha_i, ..., \alpha_\mu) = (m_i, ..., m_\mu)$ where $(\alpha_i, ..., \alpha_\mu)$ are the verifier challenges along this path such that $\forall_{j \geq i} \phi_a(j, m_j) = 1$.

The third extraction event was ruled out (with overwhelming probability) from any transcript tree generated via the Path Predicate Forking Lemma, see Theorem 6 and Lemma 11. In particular, the transcript tree generated there was shown to satisfy a predicate (with overwhelming probability) that eliminates the possibility that $\phi_a(i, m_\omega) = 0$ yet $\phi_a(\mu, m_\mu) = 1$. The analysis leveraged the way that transcripts are sampled by that tree generation algorithm. However, we will need to use a slightly different analysis this time.

First, we define the relation for all $i \in [\mu]$, commitment scheme parameters pp, and random oracle $\mathcal{H}^*$:

$$\mathcal{R}_{\mathsf{ext}}^{\mathcal{H}^*}(\mathsf{pp}, \mu{-}i) = \left\{ (\mathsf{x} = (C, m, o), \mathsf{w} = (\mathsf{tr}, M)) : \begin{array}{c} \mathsf{tr} = (\mathsf{x}, C_1, ..., C_{\mu-i}) \\ \forall_{j \in [\mu-i]}\ \alpha_j = \mathcal{H}^*(C, C_1, ..., C_{j-1}) \\ M = ((m_1, o_1), ..., (m_{\mu-i}, o_{\mu-i})) \\ \forall_{j \in [\mu-i]} \mathsf{com.Open}(\mathsf{pp}, C_j, m_j, o_j) = 1 \\ \mathsf{Extend}(i, m, \alpha_1, ..., \alpha_{\mu-i}) = (m_1, ..., m_{\mu-i}) \\ \phi_b(i, m) = 1, \phi_a(i, m) = 0, \phi_a(\mu, m_{\mu-i}) = 1 \end{array} \right\}$$

We note that by Lemma 16, in case (c) occurs, there is a $(Q + \mu)$-query polynomial time algorithm $\mathcal{A}_1$ that generates (C, m, o) and transcript prefix $\mathbf{y}$, and an independent $(Q + \mu)$-query adversary $\mathcal{A}_2^{\mathcal{H}^*}$ which generates the witness (tr, M) such that $((C, m, o), (\mathsf{tr}, M)) \in \mathcal{R}_{\mathsf{ext}}^{\mathcal{H}_{\mathbf{y}}^*}(\mathsf{pp}, \mu - i)$ for some i. $\mathcal{A}_1$ represents the algorithm that ran the partial tree generation that created all labels on the left k-ary subtree of ω and also the prefix $\mathbf{y}$ labeling the trunk (i.e., from root to ω) of subtree S_ν, and then also ran the tree extraction algorithm (Definition 10) on this left k-ary subtree of ω. Note that conditioned on event (c), ω is the first node and index i for which this tree extraction succeeds in producing an opening (m_ω, o_ω) such that $\phi_b(i, m_\omega) = 1$ but $\phi_a(i, m_\omega) = 0$. By Lemma 16 the root-to-leaf path that includes the prefix $\mathbf{y}$ and $\mathsf{lpath}(\nu)$ matches the output of some $\mathcal{A}_*^{\mathcal{H}^*}(x)$ with partially fresh random oracle $\mathcal{H}^*$, and there is also a subtree generation algorithm that is a $(Q + \mu)$-query algorithm which generated the labels on subtree S_ν. $\mathcal{A}_2^{\mathcal{H}^*}$ represents the combination of these two algorithms and also the subtree extractor that opens the commitments on these labels. Conditioned on (c), the openings of the commitment labels within this subtree all satisfy predicate ϕ_a and the opened messages of the commitment labels $\mathsf{lpath}(\nu)_0$ along the leftmost path from ν match the output of $\mathsf{Extend}(i, m, \mathsf{lpath}(v)_1)$ where $\mathsf{lpath}(\nu)_1$ are the challenge labels along this path.

Let ρ_i denote the predicate such that $\rho_i(m, m', \alpha_1, ..., \alpha_{\mu-i}) = 1$ iff $\phi_b(i, m) = 1$,

$\phi_a(i,m) = 0$, and $\phi_a(\mu,m') = 1$. By the definition of $(k^{(\mu)}, \delta, \mathsf{com}, \phi)$-almost-special-soundness, for uniform random $\beta_1, ..., \beta_{\mu-i}$:

$$\mathbb{P}_{\beta_1,...,\beta_{\mu-i}}[\rho(m, m_{\mu-i}, \beta_1, ..., \beta_{\mu-i}) = 1 : \mathsf{Extend}(i, m, \beta_1, ..., \beta_{\mu-i}) = (m_1, ..., m_{\mu-i})] \leq \delta$$

Thus, since $Q+\mu$ is polynomial in λ, by Lemma 18 the relation $\mathcal{R}^{\mathcal{H}^*}_{\mathsf{ext}}(\mathsf{pp}, \mu - i)$ is $(Q+\mu, \delta + \mathsf{negl}(\lambda))$-hard in the RO model for $Q = \mathsf{poly}(\lambda)$. This shows that for any particular index in the transcript tree, the event of type (c) occurs with probability at most $\delta + \mathsf{negl}(\lambda)$ when running the extractor with polynomial time provers making a polynomial number of queries to the RO. This experiment may effectively occur times over the course of the tree generation algorithm, but we can loosely union bound the probability that event (c) ever occurs by the runtime of the tree generation algorithm. Similarly, we can eliminate event (b) as occurring with $\mathsf{negl}(\lambda)$ by the computational binding property of the commitment scheme, which does not require an additional union bound.

Finally, letting $\epsilon(x, \mathsf{st})$ denote the probability over the parameters and random oracle that the verifier accepts the proof π output by $P^*(\mathsf{st})$ for public input x, we conclude that:

$$\mathbb{P}\left[\begin{array}{c} (x,w) \in \mathcal{R} \textit{ and} \\ \mathcal{V}^{\mathcal{H}}(\mathsf{pp}, x, \pi) = 1 \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \pi \leftarrow P^*(\mathsf{st}) \\ w \leftarrow \mathcal{E}^{P^*(\mathsf{st})}(\mathsf{pp}, x) \end{array}\right] = \mathbb{P}\left[(x,w) \in \mathcal{R} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \pi \leftarrow P^*(\mathsf{st}) \\ w \leftarrow \mathcal{E}^{P^*(\mathsf{st})}(\mathsf{pp}, x) \end{array}\right]$$

$$\geq \sum_{x,\mathsf{st}} \left(\frac{\epsilon(x,\mathsf{st}) - (Q+1)\kappa}{1-\kappa} - 2\lambda(k^\mu + Q \cdot (k^\mu - 1)) \cdot \delta - \mathsf{negl}(\lambda)\right) \cdot \mathbb{P}\left[\mathcal{A}(\mathsf{pp}) = (x, \mathsf{st}) : \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)\right]$$

$$\geq \mathbb{P}\left[\mathcal{V}^{\mathcal{H}}(\mathsf{pp}, x, \pi) = 1 : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \pi \leftarrow P^*(\mathsf{st}) \end{array}\right] - \frac{(Q+1)\kappa}{1-\kappa} - 2\lambda(k^\mu + Q \cdot (k^\mu - 1)) \cdot \delta - \mathsf{negl}(\lambda)$$

The first equality holds because $\mathcal{E}^{\mathcal{P}^*(\mathsf{st})}(\mathsf{pp}, x)$ aborts in its first step if the deterministic $\mathcal{P}^*(\mathsf{st})$ outputs π such that $\mathcal{V}^{\mathcal{H}}(\mathsf{pp}, x, \pi) \neq 1$. □

2.10 Proof of Theorem 4 (Polynomial IOP Compilation)

The fact that the compilation preserves HVZK is straightforward. We prove this part first and then move on to proving witness-extended emulation.

HVZK

Proof. Let S_{Eval} denote the HVZK simulator for Eval and S_{IOP} denote the HVZK simulator for the original polynomial IOP. We construct an HVZK simulator S for the compiled interactive argument as follows. S begins by running S_{IOP} on the input x, which produces a series of query/response pairs to arbitrarily labeled oracles that are "sent" from the IOP prover to the verifier. S simulates the view of the honest verifier in the compiled interactive proof by replacing each distinctly labeled oracle with a fresh Γ commitment to 0, *i.e.*, the zero polynomial over $\mathbb{F}_p$. By the hiding property of Γ this has negligible distance δ_0 from the commitment sent in the real protocol. (It places this commitment at the location in the transcript where the commitment to this oracle would be sent in the compiled protocol). For each query/response pair (z, y) to an oracle, S runs S_{Eval} to simulate the view of an honest-verifier in the Eval protocol opening a hiding polynomial commitment to the value y at the point z. Let P denote an upper bound on the total number of oracles sent and Q denote an upper bound on the total number of queries to IOP oracles. If the simulation of S_{IOP} has statistical distance δ_1 from the real IOP verifier's view, and each simulated Eval subprotocol has statistical distance δ_2 to the real Eval verifier's view, then the output of S has statistical distance at most $P\delta_0 + \delta_1 + Q\delta_2$ from $\mathsf{View}_{\langle P(x,w),V(x)\rangle}$. For $P, Q < \mathsf{poly}(\lambda)$ and $\delta_0, \delta_1, \delta_2 < \mathsf{negl}(\lambda)$ this statistical distance is negligible in λ. □

Witness-extended emulation (knowledge)

Proof. Without loss of generality, assume the original IOP makes at least one query to each oracle sent. An oracle which is never queried can be omitted from the IOP.

We denote by $\mathcal{V}$ the IP verifier for the compiled IP, and $\mathcal{V}_O$ the verifier for the original IOP. Given a record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st})$ for an IP prover P^* that produces accepting transcripts with non-negligible probability, we build an emulator E for the compiled IP. E begins by constructing an IOP adversary P'_O, which succeeds also with non-negligible probability on input x. Every successful interaction of P'_O with $\mathcal{V}_O$ on input x corresponds to a successful transcript of P^* with V on x. In showing how E builds P'_O we also show how E can obtain this corresponding transcript. E will make use of the emulator E_{Eval} for the commitment scheme Γ.

Finally, E can use the IOP knowledge extractor $E_{\mathsf{IOP}}^{P'_O}(x)$ in order to output a witness for x along with the corresponding transcript.

Constructing P'_O (IOP adversary) P'_O runs as follows on initial state st_0 and input x. It internally simulates the interaction of P^* and V, using the record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st})$. It begins by running this for the first round on state st_0. For every message that P^* sends in this first round, P'_O continues simulation until there is an Eval on this commitment. (There is guaranteed to be at least one Eval on each commitment, independent of the randomness). Therefore, denoting by E_{Eval} the extractor for the Eval subprotocol between P^* and $\mathcal{V}$ on a given commitment and evaluation point, the record oracle can be used to simulate E_{Eval}'s record oracle.

For each message m that P^* sends to V at the beginning of the first round, P'_O interprets m as a commitment, and attempts to extract from it a polynomial by running the PPT emulator E_{Eval}, simulating its record oracle as just described. *If it fails in any extraction attempt it aborts.*

If P'_O succeeds in all these extractions, then it uses these extracted polynomials as its first round proof oracles that it gives to $\mathcal{V}_O$. Upon receiving the first public-coin challenge from the IOP verifier, P' uses the query function to derive the corresponding queries to each of these proof oracles. Before answering, it rewinds P^* and $\mathcal{V}$ back to the point immediately after P sent its first messages, and now substitutes random challenge from $\mathcal{V}_O$ in order to simulate P^* and V on these same queries. It checks that P^*'s answers are consistent with the answers it can compute on its own from the extracted polynomials. *If any answers are inconsistent, P'_O aborts.* Otherwise, it sends the answers to $\mathcal{V}_O$.

At the end of this first round (assuming P' has not yet aborted), P'_O has stored an updated state st' for P^* based on this simulation. It proceeds to the next round and repeats the same process, using the record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}')$. Finally, if P' makes it through all rounds without aborting, then it has a final state st_V for $\mathcal{V}_O$ based on its internal simulation of P^* and V up through the end of the last round. Finally, $\mathcal{V}_O(\mathsf{st}_V)$ outputs Accept or Reject.

Analysis of P'_O success probability We claim that if $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}_0)$ outputs an accepting transcript tr with non-negligible probability, then P'_O succeeds with non-negligible probability.

Observe that for any accepting tr between P^* and V, if P'_O happens to follow the same exact sequence of query/responses without ever aborting then it succeeds because $\mathcal{V}_O$ and $\mathcal{V}$ run the same decision algorithm on the final state of query/response pairs. Thus, it remains only to take a closer look at what events cause P'_O to abort, and bound the fraction of accepting tr for which this occurs.

As indicated in bold above, there are two kinds of events that cause P'_O to abort:

- It fails to extract from a "commitment" message m sent by P^*
- After successfully extracting a polynomial f from a commitment, P^* answer queries to f in a way that is inconsistent with f.

The second type of event contradicts the evaluation binding property of Γ, therefore it occurs with negligible probability.

To analyze the first type of event, let us define "bad commitments" for a parameter D. We define this as a property of a message m (purportedly a commitment) sent in a transcript state st.

Bounding probability of commitment extraction failure The pair (m, st) is a "bad commitment" if there is less than a $1/D$ probability that extending the transcript between P^* and $\mathcal{V}$, starting from state st, will contain a successful execution of Eval on m. This probability is over the randomness of the public-coins of $\mathcal{V}$ in the extended transcript.

Let $A(\mathsf{tr})$ denote the event that a transcript tr sampled from $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}_0)$ is accepting. Let $B(\mathsf{tr})$ denote the event that tr contains a "bad commitment" (i.e. some message m sent in state st such that $\mathsf{Bad}(m, \mathsf{st}) = 1$). The conditional probability of event $A(\mathsf{tr})$ conditioned on event $B(\mathsf{tr})$ is less than $1/D$. To see this, fix (m, st) with $\mathsf{Bad}(m, \mathsf{st}) = 1$ and consider "sampling" a random tr that contains m at state st. This is done by first choosing randomly from all partial transcripts that result in (m, st) via brute force, and then running the transcript normally from state st on random public-coins. No matter how (m, st) is chosen, the probability that this process produces an accepting transcript is by definition less than $1/D$. (The second part of the transcript following (m, st) contains at least one execution of Eval on m by hypothesis, and by the definition of $B(m, \mathsf{st}) = 1$ this execution is accepting with probability less than $1/D$).

Assume that $P(A(\mathsf{tr})) \geq 1/\mathsf{poly}(\lambda)$. Applying Bayes' law,

$$P[B(\mathsf{tr})|A(\mathsf{tr})) \leq \frac{P[A(\mathsf{tr})|B(\mathsf{tr})]}{P(A(\mathsf{tr}))} \leq \mathsf{poly}(\lambda)/D \ .$$

In other words, at least a $1 - \mathsf{poly}(\lambda)/D$ fraction of accepting transcripts do not contain "bad commitments". Furthermore, so long as a commitment m is not "bad", we can invoke the witness-emulation property of Eval to say that the PPT E_Γ emulator extracts a witness polynomial from each m with overwhelming probability.

Setting $D = 2\mathsf{poly}(\lambda)$ we get that on at least a $1/2$ fraction of accepting transcripts, P'_Os simulation also succeeds (i.e. successfully extracts from each prover commitment message) with probability at least $1/2$. This means that P'_O has a non-negligible success probability conditioned on the event that tr is an accepting transcript.

In conclusion, if tr is accepting with non-negligible probability, then there is a non-negligible probability that P'_O succeeds. $\square$

Chapter 3

Multilinear Schwartz-Zippel modulo N

This chapter presents a new lemma on the zeros of multilinear polynomials modulo a composite integer N, which was necessary for the security analysis of the DARK polynomial commitment scheme in Chapter 2. Recall that in this protocol, if the input polynomial is $f = \frac{g}{N}$ for an integer polynomial g, then the final prover message is equivalent to $\tilde{f}(\alpha_1, \ldots, \alpha_\mu) = \frac{\tilde{g}(\alpha_1,\ldots,\alpha_\mu)}{N}$ where $\tilde{f}$ and $\tilde{g}$ are multi-linear polynomials with the same coefficients as f and g respectively and $\alpha_1, \ldots, \alpha_\mu$ are the verifier's random challenges. This final message is only valid if it is an integer, or equivalently, $\tilde{g}(\alpha_1, \ldots, \alpha_\mu) \equiv 0 \bmod N$. A malicious prover could break the DARK protocol if it were possible to choose $\tilde{g}$ and a large N co-prime to $\tilde{g}$ for which this condition holds true over the verifier's challenges. Thus, the security of the protocol was dependent on bounding the probability of this event for sufficiently large N. It turns out that for λ-bit challenges and sufficiently large $N \in 2^{\Omega(\lambda)}$, this probability decreases exponentially in λ.

Specifically, we derive a tight upper bound on the probability over $\mathbf{x} = (x_1, \ldots, x_\mu) \in \mathbb{Z}^\mu$ uniformly distributed in $[0, m)^\mu$ that $f(\mathbf{x}) = 0 \bmod N$ for any μ-linear polynomial $f \in \mathbb{Z}[X_1, \ldots, X_\mu]$ co-prime to N. We show that for $N = p_1^{r_1}, ..., p_\ell^{r_\ell}$ this probability is bounded by $\frac{\mu}{m} + \prod_{i=1}^{\ell} I_{\frac{1}{p_i}}(r_i, \mu)$ where I is the regularized beta function. Furthermore, we provide an inverse result that for any target parameter λ bounds the minimum size of N for which the probability that $f(\mathbf{x}) \equiv 0 \bmod N$ is at most $2^{-\lambda} + \frac{\mu}{m}$. For $\mu = 1$ this is simply $N \geq 2^{\lambda}$. For $\mu \geq 2$, $\log_2(N) \geq 8\mu^2 + \log_2(2\mu) \cdot \lambda$ the probability that $f(\mathbf{x}) \equiv 0 \bmod N$ is bounded by $2^{-\lambda} + \frac{\mu}{m}$. We also present a computational method that derives tighter bounds for specific values of μ and λ. For example, our analysis shows that for $\mu = 20$, $\lambda = 120$ (values typical in cryptography applications), and $\log_2(N) \geq 416$ the probability is bounded by $2^{-120} + \frac{20}{m}$. We provide a table of computational bounds for a large set of μ and λ values.

3.1 The DeMillo-Lipton-Schwartz–Zippel lemma

The famous DeMillo-Lipton-Schwartz–Zippel (DLSZ) lemma [74, 171, 148] states that for any field $\mathbb{F}$, non-empty finite subset $S \subseteq \mathbb{F}$, and non-zero μ-variate polynomial f over $\mathbb{F}$ of total degree d, the number of zeros of f contained in S^{μ} is bounded by $d \cdot |S|^{\mu-1}$ (or equivalently, the probability that $f(\mathbf{x}) = 0$ for $\mathbf{x}$ sampled uniformly from S^n is bounded by $\frac{d}{|S|}$). For $\mu = 1$ this simply follows from the Fundamental Theorem of Algebra, but for multivariate polynomials, the number of zeros over the whole field could be unbounded. The computational significance of this lemma is that sampling an element from S only takes $n \cdot \log_2(|S|)$ random bits but the probability of randomly sampling a zero of f from S^n is exponentially small in $|S|$. One of its original motivations was an efficient randomized algorithm for polynomial identity testing, but it has since found widespread application in computer science [103].

The classical lemma applies more broadly to integral domains, but not to arbitrary commutative rings. As a simple counterexample, over the ring of integers modulo $N = 2p$ the polynomial $f(X) = pX \bmod N$ vanishes on half of the points in $[0, N)$. The lemma has been extended to commutative rings by restricting the set S to special subsets in which the difference of any two elements is not a zero divisor [34]. For the case of $\mathbb{Z}_N$, this means that the difference of any two elements in S must be co-prime to N. Our present work explores the setting where S is the contiguous interval $[0, m)$ and thus does not satisfy this special condition. Instead, we will restrict the polynomial f to be co-prime with N, thus ruling out polynomials of the form $f(X) = u \cdot g(X)$ where u is a zero-divisor as in the counterexample above.

As a warmup, it is easy to see that any univariate linear polynomial $f(X) = c \cdot X + b$ co-prime to N has at most one root modulo N. If there were two such roots $x_1 \not\equiv x_2 \bmod N$ then $c(x_1 - x_2) \equiv 0 \mod N$ implies c is a zero divisor (i.e., $gcd(c, N) \neq 1$). Furthermore, $c \cdot x_1 \equiv -b \bmod N$ implies $-b = c \cdot x_1 + q \cdot N$ for some $q \in \mathbb{Z}$, and thus, $gcd(c, N)$ also divides b. This would contradict the co-primality of f and N. So for x uniformly distributed in $S = [0, m)$ the probability of $f(x) \equiv 0 \bmod N$ in this case is indeed at most $\frac{1}{|S|}$. Unfortunately, this does not appear to generalize nicely to polynomials of arbitrary degree. As an example, for $N = 2^\lambda$ the polynomial $f(X) = X^\lambda \bmod N$ vanishes on half of the points in $[0, N)$.

On the other hand, we are able to generalize the lemma in a meaningful way to multivariate *linear* polynomials (i.e, at most degree 1 in each variable). It turns out that the probability of sampling a zero from $S^\mu = [0, m)^\mu$ of a μ-linear polynomial can be tightly bounded by $\frac{\mu}{|S|} + \epsilon$, where the error term ϵ is bounded by a product of regularized beta functions. We also formulate an inverse lemma showing that for all sufficiently large N this error is negligibly small. In particular, for $\log N \geq 8\mu^2 + (1 + \log \mu)\lambda$ the error term is at most $2^{-\lambda}$, showing that the error decays exponentially. Our technique for deriving this threshold lower bound $t(\lambda, \mu)$ on N for a target λ formulates $t(\lambda, \mu)$ as the objective function of a knapsack problem. We derive an analytical solution by deriving bounds on the regularized beta function. We also apply a knapsack approximation algorithm to find tighter values of $t(\lambda, \mu)$ for specific values of μ and λ.

3.2 Technical overview

The regular DLSZ is relatively simple to prove. Consider the special case of a multilinear polynomial over a field. As a base case, a univariate linear polynomial has at most one root over the field. For the induction step, express $f(X_1, ..., X_{n+1}) = g(X_1, ..., X_n) + X_{n+1}h(X_1, ..., X_n)$ for random variables $X_1, \dots, X_n$. The probability that $h(x_1, ..., x_n) =$

0 over random x_i sampled from S is at most $n/|S|$ by the inductive hypothesis, and if $h(x_1, ..., x_n) = w \neq 0$ and $g(x_1, \dots, x_n) = u$, then $u + X_{n+1}w$ has at most one root (base case). By the union bound, the overall probability is at most $n/|S| + 1/|S| = (n+1)/|S|$.

This simple proof does not work for multilinear polynomials modulo a composite integer. The base case is the same for f coprime to N, which has at most one root. However, in the induction step, it isn't enough that $h(x_1, ..., x_n) \neq 0$ as it still may be a zero divisor, in which case the polynomial $u + X_{n+1}w$ is not necessarily coprime to N and the base case no longer applies. The number of roots depends on $\gcd(u + X_{n+1}w, N)$.

We prove this using a modified inductive proof over n. Our analysis takes into account the distribution of $\gcd(u + X_{n+1}w, N)$. For each prime divisor p_i of N, the highest power of p_i that divides $u + X_{n+1}w$ follows a geometric distribution. Using an inductive analysis, we are able to show that the probability $f(x_1, \dots, x_n) \equiv 0 \bmod p^r$ is bounded by the probability that $\sum_{i=1}^{n} Z_i \geq r$ for *i.i.d.* geometric variables with success parameter $1 - \frac{1}{p}$. This probability is equal to a $I_{\frac{1}{p}}(n, r)$ where I is the regularized beta function. Furthermore, by the Chinese Remainder Theorem this probability is independent for each prime factor of N, and thus, the overall probability can be bounded by a product of regularized beta functions.

Inverse lemma While our main theorem gives a tight bound on the probability for particular values of N, μ, and m, cryptographic applications require finding concrete parameters such that the probability is exponentially small in a security parameter λ. This is easy for the standard DLSZ, as it simply states that when sampling from sets of size $d \cdot 2^\lambda$ the probability is at most $\frac{d}{2^\lambda}$. Our variant does not have such a simple form so we need to explicitly find an inverse formulation. Concretely, we want to find a value N^* such that for all $N \geq N^*$ the probability that $f(X_1, \dots, X_n) \equiv 0 \bmod N$ is bounded by $2^{-\lambda}$.

To do this we first derive simple and useful bounds for the regularized beta function.

Bounds on regularized beta function We show the following three useful facts about $I_{\frac{1}{p}}(n, r)$

- $I_{\frac{1}{p}}(r, \mu) \leq \left(\frac{n}{p}\right)^r$ for $p \geq 2\mu$

- $I_{\frac{1}{p}}(r,\mu) \leq \frac{r^n}{p^r}$ for $r \geq 2\mu$
- $\log(I_{1/p}(r-1,\mu)) - \log(I_{1/p}(r,\mu))$ is non-increasing in r for any $p > \mu$ and for $r = 1$ in p.

Knapsack Formulation We then formulate finding N^* as an optimization problem. N^* is the maximum value of N such that the probability of $f(\mathbf{x}) \equiv 0 \bmod N$ is greater than $2^{-\lambda}$. For any N let $S(N)$ denote the set of pairs (p, r) where p is a prime divisor of N with multiplicity r. Taking the logarithm of both the objective and the constraint yields a knapsack-like constraint maximization [roblem where the objective is $\log(N^*) = \sum_{(p_i,r_i)\in S(N^*)} r_i \cdot \log(p_i)$ and the constraint is $\sum_{(p_i,r_i)\in S(N^*)} -\log(I_{\frac{1}{p_i}}(r,\mu)) \leq \lambda$. Using the bounds on $I_{\frac{1}{p_i}}$ and several transformations of the problem we show that any optimal solution to this problem must be bounded by $t = 8\mu^2 + \log_2(2\mu)\lambda$, which in turn implies that $N^* \leq 2^t$.

Tighter computational solution We further show that a simple greedy knapsack algorithm computes an upper bound to the knapsack problem. The algorithm uses the fact that $\frac{\log(p)}{\log(I_{1/p}(r-1,\mu))-\log(I_{1/p}(r,\mu))}$ the so-called marginal density of each item is non increasing over certain regions. Adding the densest items to the knapsack computes an upper bound to the objective. We run the algorithm on a large number of values for μ and λ and report the result.

3.3 Main theorem statement (LCSZ)

Theorem 8 (Multilinear Composite Schwartz-Zippel (LCSZ))**.** *Let $N = \prod_{i=1}^{\ell} p_i^{r_i}$ for distinct primes $p_1, ..., p_\ell$. Let f be any μ-linear integer polynomial co-prime to N. For any integer $m > 1$ and $\mathbf{x}$ sampled uniformly from $[0, m)^\mu$:*

$$\mathbb{P}_{\mathbf{x}\leftarrow[0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] \leq \frac{\mu}{m} + \prod_{i=1}^{\ell} I_{\frac{1}{p_i}}(r_i,\mu)$$

where $I_{\frac{1}{p}}(r,\mu) = (1-\frac{1}{p})^\mu \sum_{j=r}^{\infty} \binom{\mu+r-1}{r} \left(\frac{1}{p}\right)^j$ *is the regularized beta function.*

Remark 6. *The regularized beta function characterizes the tail distribution of the sum of independent geometric random variables. If* $Y = \sum_{i=1}^{\mu} Z_i$ *where each* Z_i *is an independent geometric random variable with parameter* ϵ *then* $P[Y \geq r] = I_{1-\epsilon}(r,\mu)$. Y *is a negative binomial variable with parameters* ϵ, μ.

Remark 7. *For* $m \gg \mu$ *the theorem is nearly tight for all* N. *Setting* $f(\mathbf{x}) = \prod_{i=1}^{\mu} x_i$ *and* $m = N$ *gives* $P_{\mathbf{x} \leftarrow [0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] = \mathbb{P}_{\mathbf{x} \leftarrow [0,N)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] = \prod_{i=1}^{\ell} I_{\frac{1}{p_i}}(r_i,\mu)$

Remark 8. $1 - e^{-\mu/p_i} \leq I_{\frac{1}{p_i}}(1,\mu) = 1 - (1-\frac{1}{p_i})^\mu \leq \frac{\mu}{p_i}$. *Hence, for square-free* N *the probability in theorem 8 is upper bounded by* $\frac{\mu}{m} + \frac{\mu^\ell}{N}$, *but for* $\ell > 1$ *this is a loose upper bound unless* $\mu \ll p_i$ *for all* $p_i | N$. *For* $\ell = 1$ *(i.e., prime* N*), theorem 8 coincides with the Schwartz-Zippel lemma.*

Remark 9. $I_{\frac{1}{p_i}}(r_i, 1) = \left(\frac{1}{p_i}\right)^{r_i}$. *Hence, for* $\mu = 1$, *the bound in theorem 8 is* $\frac{1}{N} + \frac{\mu}{m}$.

Proof. We begin by introducing some notations.

- For a polynomial $f \in \mathbb{Z}[X_1, ..., X_\mu]$ let $\vec{f}$ denote the coefficients of f and let $\mathsf{cont}(f)$ denote the greatest integer divisor of f, i.e. the *content*.

- $\vec{\beta} = (\beta_1, ..., \beta_\mu) \in [0,m)^\mu$ is a random variable distributed uniformly over $[0,m)^\mu$. For any $i \geq 1$, let $\vec{\beta_i} = (\beta_1, ..., \beta_i)$, let $f_0 = f$, and let $f_i(\vec{\beta_i}) := f(\beta_1, \ldots, \beta_i, X_{i+1}, \ldots, X_\mu)$.

- Given the random variable $\vec{\beta}$ distributed uniformly over $[0,m)^\mu$, define for each $j \in [1,\ell]$ and $i \in [1,\mu]$ the random variable $Y_{j,i}$ (as a function of $\vec{\beta}$) representing the multiplicity of p_j in $\mathsf{cont}(f_i(\vec{\beta_i}))$. Naturally, we set $Y_{j,0} = 0$ for all j because $\forall_j f_0 \neq 0 \bmod p_j$. (**Note:** $Y_{j,i}$ are *not* independent). For any $i \in [\mu]$ the event that $\forall_j Y_{j,i} \geq r_j$ is equivalent to the event that $f_i(\vec{\beta_i}) = 0 \bmod N$ and the event that $\forall_j Y_{j,i} = r_i$ is equivalent to $\mathsf{cont}(f_i(\vec{\beta_i})) = N$. The event $\forall_j Y_{j,\mu} \geq r_j$ is thus equivalent to $f(\vec{\beta}) = 0 \bmod N$.

- Let $\{Z_{j,i}\}$ for $i \in [\mu]$ and $j \in [\ell]$ be a set of independent random variables, where $Z_{j,i}$ is geometric with parameter $1 - \frac{1}{p_j}$.

From the CCDF (complementary CDF) of geometric random variables we have that $P[Z_{j,i} \geq k] = (\frac{1}{p_j})^k$. Setting $Z_j := \sum_{i=1}^{\mu} Z_{j,i}$, from the CCDF of the negative binomial distribution (i.e., tail distribution of the sum of independent geometric random variables) it follows that $\forall_j P[Z_j \geq r] = I_{\frac{1}{p_j}}(r, \mu)$.

Next, we establish an important subclaim:

Claim 1. *For any $i \geq 2$ and $\vec{k}, \vec{k}' \in \mathbb{N}^\ell$ where $\forall_j k_j \geq 0$:*

$$P[\forall_j Y_{j,i} \geq k_j + k'_j | \forall_j Y_{j,i-1} = k'_j] \leq \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$$

Furthermore, for all $i \geq 1$, $P[\forall_j Y_{j,i} \geq Y_{j,i-1} + k_j] \leq \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$.

Proof. Since the order of the variables does not matter, w.l.o.g. assume that $\forall_{j \in [1,\ell']} k_j \geq 1$ and $\forall_{j > \ell'} k_j = 0$. Let $N^* = \prod_{j=1}^{\ell'} p_j^{k_j}$ and $N' = \prod_{j=1}^{\ell'} p_j^{k'_j}$. For any $i \geq 2$ and any $\mathbf{x} \in [0,m)^{i-1}$, in the event that $\vec{\beta}_{i-1} = \mathbf{x}$ and $\forall j\ Y_{j,i-1} = k'_j$, then by definition $f_{i-1}(\mathbf{x}) \equiv 0 \bmod N'$ and $\forall_j f_{i-1}(\mathbf{x})/N' \not\equiv 0 \bmod p_j$. In case $i = 1$, we have that $\forall_j Y_{j,0} = 0$, $N' = 1$, and $\forall_j f_0 = f \neq 0 \bmod p_j$. Thus, for all $i \geq 1$, conditioned on the events that $\vec{\beta}_{i-1} = \mathbf{x}$ and $\forall j\ Y_{j,i-1} = k'_j$, there exist multilinear $\mu - i$ variate polynomials h_i, g_i such that $f_{i-1}(\mathbf{x})/N' = h_i(X_{i+1}, \ldots, X_\mu) + X_i \cdot g_i(X_{i+1}, \ldots, X_\mu)$ where for all j at least one of h_i or g_i is nonzero modulo p_j.

Furthermore, for any $i \geq 1$, conditioned on the events $\forall j\ Y_{j,i-1} = k'_j$ and $\vec{\beta}_{i-1} = \mathbf{x}$, the event that $\forall_j Y_{j,i} \geq k_j + k'_j$ is equivalent to the event that $h_i + \beta_i g_i \equiv 0 \bmod N^*$.

For each index $t \in [1, 2^{\mu-i}]$ let $h_i[t]$ and $g_i[t]$ denote the tth coefficients of h_i and g_i respectively (i.e., the coefficients on the tth monomial in a canonical ordering of the $2^{\mu-i}$ monomials over the $\mu - i$ variables $X_{i+1}, \ldots, X_\mu$). Given that for every $j \in [\ell']$ the polynomial $h_i + X_i \cdot g_i$ is non-zero modulo p_j, for each $j \in [\ell']$ there exists at least one index t_j for which the univariate linear polynomial $h_i[t_j] + X_i g_i[t_j]$ is non-zero modulo p_j. We now have that:

$$P[\forall_j Y_{j,i} \geq k_j + k'_j | \vec{\beta}_{i-1} = \mathbf{x} \wedge \forall_j Y_{j,i-1} = k'_j] = P[\forall_j \ h_i + \beta_i \cdot g_i \equiv 0 \bmod p_j^{k_j}] \tag{3.1}$$
$$\leq P[\forall_j \ h_i[t_j] + \beta_i \cdot g_i[t_j] \equiv 0 \bmod p_j^{k_j}] \tag{3.2}$$

For each t_j there is *at most* one solution to the equation $h_i[t_j] + X_i \cdot g_i[t_j] \equiv 0 \bmod p_j$. Consequently, by CRT, there is at most one integer solution $x^* \in [0, N^*)$ to the system of equations $\forall_j h_i[t_j] + X_i \cdot g_i[t_j] \equiv \bmod p_j$[1]. If x^* was uniform modN^* then the probability would be $\frac{1}{N^*}$. However, we must also consider the case that x^* is not uniform mod N^*.

Let E denote the event that the system of equations $\forall_j \ h_i[t_j] + \beta_i \cdot g_i[t_j] \equiv 0 \bmod p_j^{k_j}$ from line (3.2) is satisfied. There is at most one integer equivalence class modulo N^* satisfying this system of equations and the random variable β_i is uniformly distributed over $[0, m)$. Let U denote the event that $\beta_i \in [0, m - m \bmod N^*)$ and let $\bar{U}$ denote the event that $\beta_i \in [m - m \bmod N^*, m)$. Conditioned on U, β_i is uniformly distributed modulo N^*, and thus $P[E|U] \leq 1/N^*$. Conditioned on $\bar{U}$, β_i is uniformly distributed over the set $[m - m \bmod N^*, m)$. As this set is a contiguous interval of less than N^* integers it contains at most one solution to the systems of equations, and thus, $P[E|\bar{U}] \leq 1/(m \bmod N^*)$. The probability of $\bar{U}$ is exactly $\frac{m \bmod N^*}{m}$. Therefore:

$$P[E] = P[E|U] \cdot P[U] + P[E|\bar{U}] \cdot P[\bar{U}] \tag{3.3}$$
$$\leq \frac{1}{N^*} \cdot (1 - \frac{m \bmod N^*}{m}) + \frac{1}{m \bmod N^*} \cdot \frac{m \bmod N^*}{m} \tag{3.4}$$
$$\leq \frac{1}{N^*} + \frac{1}{m} \tag{3.5}$$

We also have that:

$$P[\forall_{j \in [\ell]} Z_{j,i} \geq k_j] = \prod_{j=1}^{\ell'} P[Z_{j,i} \geq k_j] = \prod_{j=1}^{\ell'} \left(\frac{1}{p_j}\right)^{k_j} = \frac{1}{N^*}$$

[1]This is where the proof falls apart for higher degree polynomials. For example if the polynomial is quadratic in each variable then there could be up to 2^ℓ solutions where ℓ is the number of prime factors.

Thus, putting it all together, for any $i \geq 1$ and any $\mathbf{x} \in [0,m)^{i-1}$:

$$P[\forall_j Y_{j,i} \geq k_j + k'_j | \vec{\beta}_{i-1} = \mathbf{x} \wedge \forall_j Y_{j,i-1} = k'_j] \leq \frac{1}{m} + \frac{1}{N^*} = \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$$

Since the probability bound is independent of $\mathbf{x}$, this implies:

$$P[\forall_j Y_{j,i} \geq k_j + k'_j | \forall_j Y_{j,i-1} = k'_j] \leq \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$$

Consequently, for any $i \geq 2$ and $\vec{k} \in \mathbb{N}^\ell$ where $\forall_j k_j > 0$:

$$P[\forall_j Y_{j,i} \geq Y_{j,i-1} + k_j] \leq \max_{\vec{k'}} P[\forall_j Y_{j,i} \geq k_j + k'_j | \forall_j Y_{j,i-1} = k'_j] \leq \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$$

Similarly, for $i = 1$ and $\vec{k} \in \mathbb{N}^\ell$ where $\forall_j k_j > 0$:

$$P[\forall_j Y_{j,i} \geq k_j] \leq \frac{1}{m} + P[\forall_j Z_{j,i} \geq k_j]$$

□

We now prove the full theorem by induction over $i \in [1, \mu]$. Specifically, we will prove the following inductive hypothesis for $i \in [1, \mu]$:

$$P[f_i(\vec{\beta}_i) = 0 \bmod N] = P[\forall_j Y_{j,i} \geq r_j] \leq \frac{i}{m} + \prod_{j=1}^{\ell} P[\sum_{k=1}^{i} Z_{j,k} \geq r_j]$$

Setting $i = \mu$ this is equivalent to the theorem statement.

Base Case: The base case follows directly from the subclaim for the case $i = 1$.

$$P[\forall_j Y_{j,1} \geq r_j] \leq \frac{1}{m} + P[\forall_j Z_{j,1} \geq r_j] = \frac{1}{m} + \prod_{j=1}^{\ell} P[Z_{j,1} \geq r_j]$$

Induction Step: Assume the inductive hypothesis holds for some $1 \leq i < \mu$, i.e.:

$$P[\forall_j Y_{j,i} \geq r_j] \leq \frac{i}{m} + \prod_{j=1}^{\ell} P[\sum_{k=1}^{i} Z_{j,k} \geq r_j]$$

We will show this implies the hypothesis holds for $i+1$:

$$P[\forall_j Y_{j,i+1} \geq r_j] = \sum_{\vec{k}\in\mathbb{N}^\ell} P[\forall_j Y_{j,i+1} - Y_{j,i} \geq r_j - k_j | \forall_j Y_{j,i} = k_j] \cdot P[\forall_j Y_{j,i} = k_j]$$

$$\leq \sum_{\vec{k}} (P[\forall_j Z_{j,i+1} \geq r_j - k_j] + \frac{1}{m}) \cdot P[\forall_j Y_{j,i} = k_j] \quad \textbf{(by subclaim)}$$

$$= \frac{1}{m} + \sum_{\vec{k}} P[\forall_j Z_{j,i+1} \geq r_j - k_j] \cdot P[\forall_j Y_{j,i} = k_j]$$

$$= \frac{1}{m} + \sum_{\vec{\Delta}\in\mathbb{Z}^\ell:\forall_j \Delta_j \leq r_j} P[\forall_j Z_{j,i+1} \geq \Delta_j] \cdot P[\forall_j Y_{j,i} = r_j - \Delta_j] \quad \textbf{(change of variables)}$$

$$= \frac{1}{m} + \sum_{\vec{\Delta}\in\mathbb{Z}^\ell:\forall_j \Delta_j \leq r_j} \sum_{\vec{k}\in\mathbb{N}^\ell} P[\forall_j Z_{j,i+1} = \Delta_j + k_j] \cdot P[\forall_j Y_{j,i} = r_j - \Delta_j]$$

$$= \frac{1}{m} + \sum_{\vec{\Delta}'\in\mathbb{N}^\ell} P[\forall_j Z_{j,i+1} = \Delta'_j] \cdot \sum_{\vec{k}\in\mathbb{N}^\ell} P[\forall_j Y_{j,i} = r_j - (\Delta'_j - k_j)] \quad \textbf{(c.o.v.)}$$

$$= \frac{1}{m} + \sum_{\vec{\Delta}'\in\mathbb{N}^\ell} P[\forall_j Z_{j,i+1} = \Delta'_j] \cdot P[\forall_j Y_{j,i} \geq r_j - \Delta'_j]$$

$$\leq \frac{1}{m} + \sum_{\vec{\Delta}'} P[\forall_j Z_{j,i+1} = \Delta'_j] \cdot (P[\forall_j \sum_{i'=1}^{i} Z_{j,i'} \geq r_j - \Delta'_j] + \frac{i}{m}) \quad \textbf{(inductive hyp.)}$$

$$= \frac{i+1}{m} + \sum_{\vec{\Delta}'} P[\forall_j Z_{j,i+1} = \Delta'_j] \cdot P[\forall_j \sum_{k=1}^{i} Z_{j,k} \geq r_j - \Delta'_j]$$

$$= \frac{i+1}{m} + P[\forall_j \sum_{k=1}^{i} Z_{j,k} \geq r_j - Z_{j,i+1}] \quad \textbf{(independence of variables)}$$

$$= \frac{i+1}{m} + \prod_{j=1}^{\ell} P[\sum_{k=1}^{i+1} Z_{j,k} \geq k_j] \quad \textbf{(independence of variables)}$$

$\square$

3.4 Bounds on the regularized beta function

The regularized incomplete beta function is defined for $k, \mu \in \mathbb{N}$ as:

$$I_\epsilon(k,\mu) = (1-\epsilon)^\mu \sum_{j=k}^{\infty} \binom{\mu+j-1}{j} \epsilon^j$$

Special values are $I_\epsilon(k,1) = \epsilon^k$, which matches the tail distribution of a geometric variable with parameter $1-\epsilon$, and $I_\epsilon(1,\mu) = 1-(1-\epsilon)^\mu$, which is the probability that at least one of μ geometric variables of parameter $1-\epsilon$ is positive.

Lemma 19. *$I_\epsilon(k,\mu) \leq (\epsilon\mu)^k$ for all $\mu, k \in \mathbb{N}$ and $\epsilon \in (0,1)$ where $\epsilon\mu \leq 1/2$.*

Proof. For $k = 0$ the statement holds because $I_\epsilon(0,\mu) = 1$. For $\mu = 1$ we have $I_\epsilon(k,1) = \epsilon^k$. It remains to prove the inequality for $\mu \geq 2$ and $k \geq 1$. We will use the following ordinary generating function identity for binomial coefficients:

$$\sum_{j=0}^{\infty} \binom{a+j}{a} x^j = \frac{1}{(1-x)^{a+1}}$$

This allows us to write $I_\epsilon(k,\mu)$ as:

$$I_\epsilon(k,\mu) = (1-\epsilon)^\mu \cdot \left(\sum_{j=0}^{\infty} \binom{\mu+j-1}{j} \epsilon^j - \sum_{j=0}^{k-1} \binom{\mu+j-1}{j} \epsilon^j \right)$$

$$= (1-\epsilon)^\mu \cdot \left(\frac{1}{(1-\epsilon)^\mu} - \sum_{j=0}^{k-1} \binom{\mu+j-1}{j} \epsilon^j\right) = 1 - (1-\epsilon)^\mu \sum_{j=0}^{k-1} \binom{\mu+j-1}{j} \epsilon^j$$

Using the geometric series identity $(\epsilon\mu)^k = 1 - (1-\epsilon\mu) \cdot \sum_{j=0}^{k-1} (\epsilon\mu)^j$, we obtain:

$$(\epsilon\mu)^k - I_\epsilon(k,\mu) = \sum_{j=0}^{k-1} (1-\epsilon)^\mu \cdot \binom{\mu+j-1}{j} \epsilon^j - (1-\epsilon\mu)(\epsilon\mu)^j$$

$$= \sum_{j=0}^{k-1} \left((1-\epsilon)^\mu \cdot \binom{\mu+j-1}{j} - (1-\epsilon\mu)\mu^j\right)\epsilon^j$$

Let $\Phi_{\epsilon,\mu}(k) = (\epsilon\mu)^k - I_\epsilon(k,\mu)$ so that the goal is to show $\Phi_{\epsilon,\mu}(k) \geq 0$ for $k, \mu \in \mathbb{N}$ where $\mu \geq 2$ and $\epsilon\mu \leq 1/2$. Observe that:

$$\Phi_{\epsilon,\mu}(1) = \epsilon\mu - 1 + (1-\epsilon)^\mu = (1-\epsilon)^\mu - (1-\epsilon\mu) \geq 0$$

$$\lim_{k\to\infty} \Phi_{\epsilon,\mu}(k) = 0$$

Thus, it suffices to show that $\Phi_{\epsilon,\mu}(k)$ is non-increasing on the interval $k \in [2,\infty)$ when $\epsilon\mu \leq 1/2$ as this implies that $\Phi_{\epsilon,\mu}(k) \geq 0$ for all $k \geq 1$, $\mu \geq 2$, and $\epsilon\mu \leq 1/2$. Moreover, we can easily show this by showing that for all $\mu, j \geq 2$ and $\epsilon\mu \leq 1/2$:

$$(1-\epsilon)^\mu \cdot \binom{\mu+j-1}{j} \leq (1-\epsilon\mu)\cdot\mu^j$$

Letting $R(\mu, j) = \frac{\binom{\mu+j-1}{j}}{\mu^j}$, observe that:

$$R(\mu,j) = \frac{\binom{\mu+j-1}{j}}{\mu^j} = \frac{\prod_{i=0}^{j-1}(\mu+i)}{j!\ \mu^j} = \prod_{i=0}^{j-1}\frac{\mu+i}{\mu\cdot(i+1)}$$

Since $\mu+i \leq \mu\cdot(i+1)$ for all $\mu \geq 2$ and $i \geq 0$, it follows that $R(\mu,j) \leq R(\mu,2) = \frac{1}{2}\cdot(1+\frac{1}{\mu})$ for $j \geq 2$. Furthermore, for $\epsilon \in (0, 1/\mu)$:

$$\frac{d}{d\epsilon}\frac{(1-\epsilon)^\mu}{1-\epsilon\mu} = \frac{(\mu-1)\epsilon\mu(1-\epsilon)^{\mu-1}}{(1-\epsilon\mu)^2} \geq 0$$

Thus, for $\epsilon \in (0, \frac{1}{2\mu}]$ and $\mu \geq 2$:

$$\frac{(1-\epsilon)^\mu}{(1-\epsilon\mu)}\cdot R(\mu,j) \leq \frac{(1-\frac{1}{2\mu})^\mu}{1/2}\cdot R(\mu,2) \leq \frac{(1+\frac{1}{\mu})}{\sqrt{e}} \leq e^{\frac{1}{\mu}-1/2} \leq 1$$

This completes the proof. □

Lemma 20. *$I_\epsilon(k,\mu) \leq \epsilon^k \cdot k^\mu$ for $k \geq 2\mu$ and $\epsilon \leq 1/2$.*

This is tighter than Bound 1 for larger k, i.e. when $k^\mu < \mu^k$.

Proof. Similar to the analysis in Bound 1, define $\Psi_{\epsilon,\mu}(k) = \epsilon^k k^\mu - I_\epsilon(k,\mu)$ so that:

$$\Psi_{\epsilon,\mu}(k) = \epsilon^k k^\mu - 1 + (1-\epsilon)^\mu \sum_{j=0}^{k-1} \binom{\mu+j-1}{j} \epsilon^j$$

Bound 2 holds iff $\Psi_{\epsilon,\mu}(k) \geq 0$ for all $k \geq 2\mu$ and $\epsilon \leq 1/2$. For $\mu = 1$ we have $I_\epsilon(k,1) = \epsilon^k$ so

$$\Psi_{\epsilon,1}(k) = \epsilon^k \cdot k - \epsilon^k \geq 0$$

Furthermore, $\lim_{k\to\infty} \Psi_{\epsilon,\mu}(k) = 0$. Thus, it suffices to show that $\Psi_{\epsilon,\mu}$ is non-increasing on the interval $[2\mu, \infty)$ for $\mu \geq 2$ and $\epsilon \leq 1/2$.

Observe that:

$$\Psi_{\epsilon,\mu}(k+1) - \Psi_{\epsilon,\mu}(k) = \epsilon^{k+1}(k+1)^\mu - \epsilon^k k^\mu + (1-\epsilon)^\mu \binom{\mu+k-1}{k} \epsilon^k$$

$$= \epsilon^k \left[\epsilon(k+1)^\mu - k^\mu + (1-\epsilon)^\mu \binom{\mu+k-1}{k} \right]$$

Defining:

$$\Delta_\epsilon(k,\mu) := (1-\epsilon)^\mu \frac{\binom{\mu+k-1}{k}}{k^\mu} + \epsilon(1+\frac{1}{k})^\mu$$

$\Psi_{\epsilon,\mu}(k)$ is non-increasing on $k \in [2\mu, \infty]$ iff $\Delta_\epsilon(k,\mu) \leq 1$ for all $k \geq 2\mu$. We will prove this for $\mu \geq 2$ and $\epsilon \leq 1/2$. Using the inequality $(1+\frac{1}{k})^\mu \leq \sqrt{e}$ for $k \geq 2\mu$:

$$\Delta_\epsilon(2\mu,\mu) \leq (1-\epsilon)^\mu \frac{\binom{3n-1}{2\mu}}{(2\mu)^\mu} + \epsilon\sqrt{e}$$

The right hand side is decreasing as $\mu \to \infty$ and thus for $\mu \geq 2$ and $\epsilon \leq 1/2$:

$$\Delta_\epsilon(2\mu,\mu) \leq (1-\epsilon)^2 \cdot \frac{\binom{5}{4}}{4^2} + \epsilon\sqrt{e} = (1-\epsilon)^2 \cdot \frac{5}{16} + \epsilon\sqrt{e} \ < 1$$

To see why this is less than 1 for $\epsilon \leq 1/2$, note that $\frac{d}{d\epsilon}(1-\epsilon)^2 \cdot c_1 + \epsilon \cdot c_2 = 2c_1\epsilon + c_2 - 2c_1$

is positive when $\epsilon \geq 0$ and $c_2 \geq 2c_1$, and $\sqrt{e} > \frac{5}{8}$. Thus, on the interval $\epsilon \in [0, \frac{1}{2}]$, $(1-\epsilon)^2 \frac{5}{16} + \epsilon\sqrt{e} \leq \frac{1}{4} \cdot \frac{5}{16} + \frac{1}{2}\sqrt{e} = 0.902...$

Finally, since $\Delta_\epsilon(k, \mu)$ is decreasing as $k \to \infty$:

$$\forall_{k \geq 2\mu, \mu \geq 2, \epsilon \leq 1/2} \ \Delta_\epsilon(k, \mu) \leq \Delta_\epsilon(2\mu, \mu) < 1$$

□

Corollary 2. *For any prime p and any positive integer μ*

$$P[\sum_{i=1}^{\mu} X_i \geq r] = I_{\frac{1}{p}}(r, \mu) \leq \begin{cases} \frac{r^\mu}{p^r} \ \textit{if } r \geq 2\mu \\ (\frac{\mu}{p})^r \ \textit{if } p \geq 2\mu \\ 1 \ \textit{otherwise} \end{cases}$$

, where X_i are independent geometric variables with parameter $(\frac{1}{p})$ and $P[X_i \geq r] = \left(\frac{1}{p}\right)^r$

3.5 Inverse LCSZ

theorem 8 (LCSZ) bounds the probability $\mathbb{P}_{\mathbf{x} \leftarrow [0,m)^\mu}[f(\mathbf{x} \equiv 0 \bmod N]$ for given values of μ, N, and m, which has the form $\frac{\mu}{m} + \delta_{N,\mu}$. In the case that N is prime, $\delta_{N,\mu} = \frac{\mu}{N}$, which agrees with the standard Schwartz-Zippel lemma applied to μ-linear polynomials. The term $\delta_{N,\mu}$ for composite N, which is dependent on both μ and the factorization of N, has a complicated closed form expression in terms of a product of regularized beta functions.

This section analyzes the inverse: for a given $\mu, \lambda \in \mathbb{N}$ what size threshold $t(\lambda, \mu) \in \mathbb{N}$ is sufficient such that $\delta_{N,\mu} \leq 2^{-\lambda}$ for all $N \geq t(\lambda, \mu)$? In other words:

$$t(\lambda, \mu) := \sup\{N \in \mathbb{N} : \prod_{(p,r) \in S(N)} I_{\frac{1}{p}}(r, \mu) \geq 2^{-\lambda}\} \qquad (t(\lambda, \mu) \text{ def})$$

For $\mu = 1$, since $I_{1/p}(r, 1) = \frac{1}{p^r}$ and $\prod_{(p,r) \in S(N)} I_{\frac{1}{p}}(r, \mu) = \frac{1}{N}$, it is easy to see that $t(\lambda, \mu) = 2^\lambda$. For $\mu \geq 2$, the value of $t(\lambda, \mu)$ (or even an upper bound) is not nearly as easy to derive. For the rest of this section we will focus on this $\mu \geq 2$ case. We will analytically derive an upper bound to $t(\lambda, \mu)$, showing that $\log t(\lambda, \mu) \in O(\mu^{2+\epsilon} + \frac{\lambda}{\epsilon})$ for any $\epsilon \geq \log_\mu(2)$.

Theorem 9 (Inverse LCSZ)**.** *For all $\mu \geq 2$, $\epsilon \geq \log_\mu(2)$, and all N such that*

$$\log N \geq 4\mu^{2+\epsilon} + (1 + \frac{1}{\epsilon}) \cdot \lambda$$

we have that for any μ-linear polynomial f that is coprime with N

$$\mathbb{P}_{x \leftarrow [0,m)^\mu}[f(x) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}$$

By setting $\epsilon = \log_\mu(2)$ we get:

Corollary 3. *For all N such that*

$$\log N \geq 8\mu^2 + \log_2(2\mu) \cdot \lambda$$

we have that for any n-linear polynomial f that is coprime with N

$$P_{x \leftarrow [0,m)^\mu}[f(x) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}$$

3.5.1 Proof of Inverse LCSZ (theorem 9)

By theorem 8 (CSZ) we have that for $N = \prod_i p_i^{r_i}$:

$$\mathbb{P}_{\mathbf{x} \leftarrow [0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] \leq \prod_i I_{\frac{1}{p_i}}(r_i, \mu) + \frac{\mu}{m}.$$

For $\mu = 1$ and $\log_2(N) \geq \lambda$, theorem 8 shows that $\mathbb{P}_{x \leftarrow [0,m)}[f(x) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{1}{m}$. This is derived by substituting $I_{1/p}(r, 1) = \frac{1}{p^r}$, which gives $\mathbb{P}_{x \leftarrow [0,m)}[f(x) \equiv 0 \bmod N] \leq \frac{1}{N} + \frac{1}{m}$. The case $\mu \geq 2$ is more complicated. This is the focus of the rest of the proof..

For a given $N \in \mathbb{N}$, let $S(N)$ denote the set of pairs (p, r) where p is a prime divisor of N and r is its multiplicity, i.e. $N = \prod_{(p,r) \in S} p^r$. Define:

$$t(\lambda,\mu) := \sup\{N \in \mathbb{N} : \prod_{(p,r)\in S(N)} I_{\frac{1}{p}}(r,\mu) \geq 2^{-\lambda}\} \qquad (t(\lambda,\mu)\text{ def})$$

It follows from theorem 8 (CSZ) that if $N \geq t(\lambda,\mu)$ then

$$\mathbb{P}_{\mathbf{x}\leftarrow[0,m)^{\mu}}[f(\mathbf{x}) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}.$$

Assuming $t(\lambda,\mu) < \infty$, we obtain the following constrained maximization problem:

$$\log_2 t(\lambda,\mu) := \max_{N\in\mathbb{N}} \log_2 N \text{ subject to } \sum_{(p,r)\in S(N)} -\log_2 I_{\frac{1}{p}}(r,\mu) \leq \lambda. \qquad \text{(Constrained Maximization 1)}$$

In order to derive an upper bound on $\log_2 t(\lambda,\mu)$, we construct a sequence of modified maximization problems, each of which is an upper bound to the prior. The last in this sequence is a knapsack problem for which we analytically derive an upper bound.

Definition 18. *Let* $\mathsf{val}(p,r) := r\log_2 p$ *and let* $\mathsf{weight}_\mu(p,r) = -\log_2 I_{\frac{1}{p}}(r,\mu)$. *Additionally, for all* $\epsilon \geq \log_\mu(2)$, *let:*

$$\mathsf{weight}_{\mu,\epsilon}(p,r) := \begin{cases} r\cdot(\log_2(p)-\log_2(\mu)) \text{ if } p \geq \mu^{1+\epsilon} \\ r\cdot\log_2(p) - \mu\cdot\log_2(r) \text{ if } p < \mu^{1+\epsilon} \wedge r > 2(1+\epsilon)\frac{\mu\ln\mu}{\ln p} \\ 0 \text{ otherwise} \end{cases}$$

Claim 2. *For any prime* p *and* $r\in\mathbb{N}$, *if* $\epsilon \geq \log_\mu(2)$ *then* $\mathsf{weight}_\mu(p,r) \leq \mathsf{weight}_{\mu,\epsilon}(p,r)$.

Proof. If $\epsilon \geq \log_\mu(2)$ then $\mu^{1+\epsilon} \geq 2\mu$ and the claim follows from Corollary 2 (to lemma 19 and lemma 20). □

Claim 3. *For any prime* p, $\mathsf{weight}_{\mu,\epsilon}(p,r)$ *is non-decreasing over* $r \geq 1$ *and increasing for* $r > 2(1+\epsilon)\frac{\mu\log\mu}{\log p}$.

Proof. We first show that the function $\mathsf{weight}_{\mu,\epsilon}(p,r)$ is non-decreasing in r in each of the three cases, which comprise three subdivisions of the plane $\mathsf{Primes} \times \mathbb{N}$, which we denote S_A, S_B, and S_C respectively. S_A contains all (p,r) where $p \geq \mu^{1+\epsilon}$, in which case $\frac{d}{dr}\mathsf{weight}_{\mu,\epsilon}(p,r) = \log_2(p) - \log_2(\mu) > 0$. S_B contains all (p,r) where $r > 2(1+\epsilon)\frac{\mu \ln \mu}{\ln p}$ and $p < \mu^{1+\epsilon}$, in which case $\frac{d}{dr}\mathsf{weight}_{\mu,\epsilon}(p,r) = \log_2(p) - \frac{\mu}{r \ln 2}$, and $\log_2(p) - \frac{\mu}{r\ln 2} \geq \log_2 p - \frac{1}{2\ln 2} > 0$. The weight function is constant at 0 for all remaining pairs, which comprise set S_C.

It remains to show that $\mathsf{weight}_{\mu,\epsilon}(p,r)$ increases in r across the boundary between S_B and S_C, for which it suffices to show that the weight is positive for all $(p,r) \in S_B$. Suppose, towards contradiction, that $r \log_2 p \leq \mu \log_2 r$ and $r > 2(1+\epsilon)\frac{\mu \log_2 \mu}{\log_2 p}$. This would imply that both $\frac{r}{\log_2 r} \leq \frac{\mu}{\log_2 p}$ and $\frac{r}{2(1+\epsilon)\log \mu} > \frac{\mu}{\log_2 p}$, which implies that $\log_2 r > 2(1+\epsilon)\log \mu$. Since $\frac{r}{\log_2 r}$ is monotonic increasing in r, this in turn implies that $\frac{r}{\log_2 r} > \frac{\mu^2}{4 \log_2 \mu} \geq \mu$ for all $\mu \geq 1$. Finally, the implication that $\frac{r}{\log_2 r} > \mu$ contradicts the assumption that $r \log_2 p \leq \mu \log_2 r$.

□

The first modified maximization problem is:

$$\max_{N \in \mathbb{N}} \sum_{(p,r)\in S(N)} \mathsf{val}(p,r) \text{ subject to } \sum_{(p,r)\in S(N)} \mathsf{weight}_{\mu,\epsilon}(p,r) \leq \lambda$$

(Constrained Maximization 2)

Claim 4. *eq.* (Constrained Maximization 2) *is an upper bound to eq.* (Constrained Maximization 1) *for any μ and $\epsilon \geq \log_\mu(2)$:*

Proof. For any $N \in \mathbb{N}$, by definition $\sum_{(p,r)\in S(N)} \mathsf{val}(p,r) = \log_2 N$. Thus the only difference between the two maximization problems are the constraints. Furthermore, if $\epsilon \geq \log_\mu(2)$ then, by Claim 2, eq. (Constrained Maximization 2) is simply a relaxation of the constraints in eq. (Constrained Maximization 1). □

Let Primes denote the infinite set of prime numbers.

Definition 19 (Marginal Value/Weight)**.** *Using $\mathsf{val}(p,r)$ and $\mathsf{weight}(p,r)$ as defined in Definition 18, $r \geq 1$ and $p \in \mathsf{Primes}$, let $\Delta\mathsf{val}(p,r) := \mathsf{val}(p,r) - \mathsf{val}(p,r-1) = \log_2 p$ and*

$\Delta\mathsf{val}(p,0) := \mathsf{val}(p,0) = 0$. *Similarly, for* $r \geq 1$ *let* $\Delta\mathsf{weight}_{\mu,\epsilon}(p,r) := \mathsf{weight}_{\mu,\epsilon}(p,r) - \mathsf{weight}_{\mu,\epsilon}(p,r-1)$ *and* $\Delta\mathsf{weight}_{\mu,\epsilon}(p,0) := \mathsf{weight}_{\mu,\epsilon}(p,0) = 1$.

By Claim 3, $\Delta\mathsf{weight}_{\mu,\epsilon}(p,r)$ is non-negative for all prime p and $r \in \mathbb{N}$, and positive for $r > 2(1+\epsilon)\frac{\mu \log \mu}{\log p}$. The following knapsack problem gives an upper bound to eq. (Constrained Maximization 2) :

$$\max_{S \subseteq \mathsf{Primes} \times \mathbb{N}} \sum_{(p,r)\in S} \Delta\mathsf{val}(p,r) \text{ subject to } \sum_{(p,r)\in S} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \leq \lambda. \quad \text{(Knapsack Problem)}$$

Claim 5. *eq.* (Knapsack Problem) *is an upper bound to eq.* (Constrained Maximization 2)*.*

Proof. Le S^* denote argmax of eq. (Knapsack Problem) with $v^* = \sum_{(p,r)\in S^*} \Delta\mathsf{val}(p,r)$ and suppose (towards contradiction) that there exists $N \in \mathbb{N}$ such that $\sum_{(p,r)\in S(N)} \mathsf{weight}_{\mu,\epsilon}(p,r) \leq \lambda$ and $\sum_{(p,r)\in S(N)} \mathsf{val}(p,r) > v^*$. Consider the set S', which includes $S(N)$ and adds for each $(p,r) \in S(N)$ the pairs (p,j) for each $j \leq r$, i.e:

$$S' = \bigcup_{(p,r)\in S(N)} \bigcup_{j=0}^{r} \{(p,j)\}$$

Observe that:

$$\sum_{(p,r)\in S'} \Delta\mathsf{val}(p,r) = \sum_{(p,r)\in S(N)} \sum_{j=0}^{r} \Delta\mathsf{val}(p,j) = \sum_{(p,r)\in S(N)} \mathsf{val}(p,r) > v^*$$

$$\sum_{(p,r)\in S'} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) = \sum_{(p,r)\in S(N)} \sum_{j=0}^{r} \Delta\mathsf{weight}_{\mu,\epsilon}(p,j) = \sum_{(p,r)\in S(N)} \mathsf{weight}_{\mu,\epsilon}(p,r) \leq \lambda$$

This is a contradiction to the assumption that v^* is the solution to eq. (Knapsack Problem).

□

Finally, we prove a series of claims that will enable us to derive an upper bound on eq. (Knapsack Problem). First, we define:

Definition 20 (Density). $\mathsf{density}_{\mu,\epsilon}(p,r) = \frac{\Delta\mathsf{val}(p,r)}{\Delta\mathsf{weight}_{\mu,\epsilon}(p,r)}$ *where* $\Delta\mathsf{val}(p,r)$ *and* $\Delta\mathsf{weight}(p,r)$ *are defined in Definition 19.*

Claim 6. *For all* $S \subseteq \mathit{Primes} \times \mathbb{N}$*:*

$$\sum_{(p,r)\in S} \Delta\mathsf{val}(p,r) \leq \sum_{(p,r)\in S} \Delta\mathsf{weight}(p,r) \cdot \max_{(p,r)\in S}\{\mathsf{density}_{\mu,\epsilon}(p,r)\}$$

Proof.

$$\begin{aligned}\sum_{(p,r)\in S} \Delta\mathsf{weight}(p,r) \cdot \max_{(p,r)\in S}\{\mathsf{density}_{\mu,\epsilon}(p,r)\} &\geq \sum_{(p,r)\in S} \mathsf{weight}_{\mu}(p,r) \cdot \mathsf{density}_{\mu,\epsilon}(p,r) \\ &= \sum_{(p,r)\in S} \mathsf{val}(p,r)\end{aligned}$$

□

Claim 7. *If* $\mu \geq 2$, $\epsilon \geq \log_{\mu}(2)$, $r \geq 1$, *and* $p \geq \mu^{1+\epsilon}$ *then* $\mathsf{density}_{\mu,\epsilon}(p,r) \leq 1 + \frac{1}{\epsilon}$

Proof. If $\epsilon \geq \log_{\mu}(2)$ and $p \geq \mu^{1+\epsilon}$ then $p \geq 2\mu$, and thus:

$$\mathsf{density}_{\mu,\epsilon}(p,r) = \frac{\Delta\mathsf{val}(p,r)}{\Delta\mathsf{weight}_{\mu,\epsilon}(p,r)} = \frac{\log p}{\log_2(p) - \log_2(\mu)}$$

Furthermore, $p \geq \mu^{1+\epsilon}$ implies that:

$$\log_2(p) - \log_2(\mu) \geq \log_2(p) - \frac{1}{1+\epsilon}\log_2(p) = \log_2(p) \cdot \frac{\epsilon}{1+\epsilon}$$

Thus $\mathsf{density}_{\mu,\epsilon}(p,r) \leq \frac{\log_2(p)}{\log_2(p)\cdot\frac{\epsilon}{1+\epsilon}} = 1 + \frac{1}{\epsilon}$ □

Claim 8. *If* $\mu \geq 2$, $\epsilon \geq \log_{\mu}(2)$, $r > 2(1+\epsilon)\cdot\frac{\mu\cdot\ln\mu}{\ln p}$, *and* $p < \mu^{1+\epsilon}$ *then* $\mathsf{density}_{\mu,\epsilon}(p,r) \leq 1+\frac{1}{\epsilon}$

Proof. Since $p < \mu^{1+\epsilon}$, the conditions on r imply $r > 2(1+\epsilon)\frac{\mu\ln\mu}{\ln p} > 2\mu$ and thus:

$$\mathsf{density}_{\mu,\epsilon}(p,r) = \frac{\Delta\mathsf{val}(p,r)}{\Delta\mathsf{weight}_{\mu,\epsilon}(p,r)} = \frac{\ln p}{\ln p + \mu\ln\frac{r-1}{r}} = \frac{1}{1 - \frac{\mu}{\ln p}\ln\frac{r}{r-1}}$$

$\mathsf{density}_{\mu,\epsilon}(p,r)$ is non-negative because $\Delta\mathsf{val}(p,r)$ is non-negative and $\Delta\mathsf{weight}(p,r)$ is positive over $r > 2(1+\epsilon)\frac{\mu\log\mu}{\log p}$. Thus, for p and r satisfying these conditions, it must be the case that $0 \le \frac{\mu}{\ln p}\ln\frac{r}{r-1} < 1$. Furthermore, this term is decreasing (approaching zero) as r increases, which shows that for r and p subject to these conditions $\mathsf{density}_{\mu,\epsilon}(p,r)$ is also decreasing in r. Combining this with the fact that $\ln\frac{r}{r-1} = \ln(1+\frac{1}{r-1}) \le \frac{1}{r-1}$:

$$\mathsf{density}_{\mu,\epsilon}(p,r) \le \mathsf{density}_{\mu,\epsilon}(p, 2(1+\epsilon)\frac{\mu\ln\mu}{\ln p}+1) \le \frac{1}{1-\frac{\mu}{\ln p}\frac{\ln p}{2(1+\epsilon)\mu\ln\mu}} = \frac{1}{1-\frac{1}{2(1+\epsilon)\ln\mu}} \le 1+\frac{1}{\epsilon}$$

□

Claim 9. *For $\alpha \in \mathbb{R}$ let $\mathsf{Primes}(\alpha)$ denote the set of prime numbers strictly less than α. For $\mu \in \mathbb{N}$ and $\epsilon \in (0,1)$ define:*

$$B_{\mu,\epsilon} := \{(p,r) : p \in \mathsf{Primes}(\mu^{1+\epsilon}), r \le 2(1+\epsilon)\frac{\mu\ln\mu}{\ln p}\}$$

Then :

$$\sum_{(p,r)\in B_{\mu,\epsilon}} \Delta\mathsf{val}(p,r) \le 4\mu^{2+\epsilon}$$

Proof. Let $\pi(X)$ denote the prime counting function. We use the fact that $\pi(x) \le 1.3\cdot\frac{x}{\ln(x)}$ for all $x > 1$ [144].

$$\sum_{(p,r)\in B_{\mu,\epsilon}} \Delta\mathsf{val}(p,r) = \sum_{p\in\mathsf{Primes}(\mu^{1+\epsilon})} \sum_{r=0}^{\lfloor 2(1+\epsilon)\frac{\mu\ln\mu}{\ln p}\rfloor} \log p = \sum_{\mathsf{Primes}(\mu^{1+\epsilon})} 2(1+\epsilon)\mu\log_2\mu$$
$$\le 1.3\cdot\frac{\mu^{1+\epsilon}}{\ln(\mu^{1+\epsilon})}\cdot\frac{2\cdot(1+\epsilon)\mu\ln(\mu)}{\ln(2)} \le 4\mu^{2+\epsilon}$$

□

Putting together these claims, we obtain the bound:

Claim 10. *For all $\mu \geq 2$, $\epsilon \geq \log_\mu(2)$ and $S \subseteq \mathsf{Primes} \times \mathbb{N}$:*

$$\sum_{(p,r)\in S} \Delta\mathsf{val}(p,r) \leq 4n^{2+\epsilon} + \sum_{(p,r)\in S} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \cdot (1+\frac{1}{\epsilon})$$

Proof. Partition S into disjoint sets S_1 and S_2 such that S_2 contains all the pairs $(p,r) \in S$ for which either $p \geq \mu^{1+\epsilon}$ or $r > 2(1+\epsilon)\frac{\mu \ln \mu}{\ln p}$, and S_1 contains the remaining pairs. $S_1 \subseteq B_{\mu,\epsilon}$ from Claim 9 and thus $\sum_{(p,r)\in S_1} \Delta\mathsf{val}(p,r) \leq 4\mu^{2+\epsilon}$. Furthermore, by Claim 7, if $(p,r) \in S_2$ then $\mathsf{density}_{\mu,\epsilon}(p,r) \leq 1 + \frac{1}{\epsilon}$ and by Claim 6:

$$\sum_{(p,r)\in S_2} \Delta\mathsf{val}(p,r) \leq \sum_{(p,r)\in S_2} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \cdot (1+\frac{1}{\epsilon})$$

Putting everything together:

$$\sum_{(p,r)\in S} \Delta\mathsf{val}(p,r) = \sum_{(p,r)\in S_1} \Delta\mathsf{val}(p,r) + \sum_{(p,r)\in S_2} \Delta\mathsf{val}(p,r) \leq 4n^{2+\epsilon} + \sum_{(p,r)\in S} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \cdot (1+\frac{1}{\epsilon})$$

□

Finally, we can conclude from Claim 10 that for any $S \in \mathsf{Primes} \times \mathbb{N}$, $\mu \geq 2$, and $\epsilon \geq \log_\mu(2)$, if $\sum_{(p,r)\in S} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \leq \lambda$, i.e., if S satisfies the constraints of eq. (Knapsack Problem) then:

$$\sum_{(p,r)\in S} \Delta\mathsf{weight}_{\mu,\epsilon}(p,r) \leq 4n^{2+\epsilon} + \lambda \cdot (1+\frac{1}{\epsilon})$$

The right hand side of the equation is therefore an upper bound for the solution to eq. (Knapsack Problem), and consequently (by Claim 12 and Claim 5), an upper bound for the solution $t(\lambda,\mu)$ to eq. (Constrained Maximization 1) when $\mu \geq 2$. In conclusion, for any $N \in \mathbb{N}$, $\mu \geq 2$, and $\epsilon \geq \log_\mu(2)$, if $\log_2 N \geq 4n^{2+\epsilon} + \lambda \cdot (1+\frac{1}{\epsilon})$ then $\log_2 N \geq t(\lambda,\mu)$ and:

$$\mathbb{P}_{\mathbf{x}\leftarrow[0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}$$

3.5.2 Computational Inverse LCSZ

Theorem 9 provides an analytical upper bound on $t(\lambda, \mu)$ for any $\mu, \lambda \in \mathbb{N}$. However, the analytical bound does not appear to be tight for $\mu \geq 2$ (for $\mu = 1$ it is tight). This next section provides an algorithm to derive an upper bound on $t(\lambda, \mu)$ for any specific values of λ, μ. The algorithm gives tighter bounds than theorem 9 for a table of tested values (Table 3.1). This is useful in practice, e.g. for deriving concrete cryptographic security parameters in cryptographic protocols that rely on LCSZ.

As shown in the prior section, $t(\lambda, \mu)$ is upper bounded by a solution to a knapsack problem, eq. (Knapsack Problem), over the infinite set of items $\mathsf{Primes} \times \mathbb{N}$. There is a simple well-known greedy algorithm that returns an upper bound to the optimal value for the knapsack problem over a *finite* set of items. This algorithm greedily adds items to the knapsack in order of decreasing density until the knapsack overflows the weight bound, and returns the total value of the added items at this point. Over an infinite set of items, it is not generally possible to sort by decreasing density. However, by leveraging monotonicity properties of the density function in our case, we are able to adapt the greedy approximation algorithm to work for eq. (Knapsack Problem). In particular, we are able to enumerate over pairs in $\mathsf{Primes} \times \mathbb{N}$ in an order of non-increasing density.

Claim 11. *Let A and B be any pair of discrete strictly ordered sets which contain minimum elements $a_0 \in A$ and $b_0 \in B$. Let $a+$ denote the next element of A after a and likewise $b+$ the next element of B after b. If $f : A \times B \to \mathbb{R}^+$ is monotonically non-increasing over pairs (a, b_0) as $a \in A$ increases, and for any fixed a, monotonically non-increasing over pairs (a, b) as $b \in B$ increases, then the following algorithm enumerates the pairs $(a, b) \in A \times B$ in order of decreasing $f(a, b)$. The algorithm initializes the set $C = \{(a_0, b_0)\}$ and also a variable max_A to keep track of the highest order element in A seen so far. At each iteration, it removes a pair $(a, b) \in C$ of lowest $f(a, b)$ value and appends (a, b) to the output enumeration list. Before proceeding to the next iteration, it adds $(a, b+)$ to C, and if $a = \mathsf{max}_A$ then it also adds $(a+, b_0)$ to C and updates $\mathsf{max}_A := a+$.*

Proof. Suppose, towards contradiction, that the algorithm appends (a, b) to the output list, and there exists at least one pair (a', b') not yet in the list at this iteration such that $f(a', b') > f(a, b)$. If $b' \neq b_0$ and (a', b_0) appeared in the output list already, then each (a', b^*) for $b^* \in [b_0, b']$ would also have been added to C and removed before (a, b) because $f(a', b^*) \geq f(a', b') > f(a, b)$. This would be a contradiction, so it remains to consider the case that $b' = b_0$, i..e. that (a', b_0) did not appear in the list and $f(a', b_0) > f(a, b)$.

First, this implies that $a' > a$. Otherwise, If $a' \leq a$, then (a', b_0) would have been added to C before (a, b) and thus removed before (a, b). Second, $b \neq b_0$ and $f(a, b) < f(a, b_0)$, as otherwise this would imply that $f(a', b_0) > f(a, b) \geq f(a, b_0)$, contradicting the monotonicity property. Thus (a, b_0) must already appear in the output list because it is added to C before (a, b) and removed before (a, b). Furthermore, for all $a^* \in [a, a']$, $f(a^*, b_0) \geq f(a', b_0) > f(a, b)$, thus each such pair (a^*, b_0) would have been added and removed before (a, b). This is a contradiction. $\square$

We use the enumeration algorithm of Claim 11 to implement the greedy algorithm that obtains an upper bound to a generic knapsack problem over the infinite set of items $\mathsf{Primes} \times \mathbb{N}$:

$$\max_{S \subseteq \mathsf{Primes} \times \mathbb{N}} \sum_{(p,r) \in S} \mathsf{val}(p, r) \text{ subject to } \sum_{(p,r) \in S} \mathsf{weight}(p, r) \leq \lambda. \tag{Generic Knapsack Problem}$$

for any $\mathsf{val}, \mathsf{weight} : \mathsf{Primes} \times \mathbb{N} \to \mathbb{R}^+$ where $\mathsf{density}(p, r) = \frac{\mathsf{val}(p,r)}{\mathsf{weight}(p,r)}$ is monotonic non-increasing over r for any fixed p, and also over p for fixed $r = 1$. This is presented below as algorithm 3.1.

Moreover, we will show that the density function defined in terms of $\mathsf{val}(p, r) = \log p$, $\mathsf{weight}(p, 1) = -\log I_{1/p}(1, \mu)$, and $\mathsf{weight}(p, r) = \log I_{1/p}(r-1,) - \log I_{1/p}(r, \mu)$ for $r > 1$ satisfies this monotonicity property. The density function in eq. (Knapsack Problem) also has this monotonicity property, but we are able to obtain a tighter bound on eq. (Constrained Maximization 1) by defining regularized beta function directly instead of the simpler form upper bounds on the regularized beta function that were more useful for deriving the analytical result in theorem 9.

```
Input $\mu \in \mathbb{N}, \lambda \in \mathbb{N}$

1. Initialize a max heap $H$ that stores tuples $(p, r, d) \in \mathbb{P} \times \mathbb{Z} \times \mathbb{R}$ and sorts them by the
   third value.
2. Initialize $w \leftarrow 0$ and $v \leftarrow 0$.
3. Push $(\mathsf{density}(2,1), 2, 1)$ onto the heap and set $\mathsf{pmax} = 2$.
4. While $w < \lambda$
5.  (a) Pop $(p, r, d)$ from $H$.
    (b) Push $(p, r+1, (\mathsf{density}(p, r+1))$ onto $H$
    (c) Set $v \leftarrow v + \mathsf{val}(p, r)$
    (d) Set $w \leftarrow w + \mathsf{weight}(p, r)$
    (e) If $p = \mathsf{pmax}$ then set $\mathsf{pmax} \leftarrow \mathsf{next\_prime}(p)$ and push $(\mathsf{density}(\mathsf{pmax}, 1), \mathsf{pmax}, 1)$
        onto $H$
6. Output $v$
```

Algorithm 3.1: Greedy algorithm that returns an upper bound to eq. (Constrained Maximization 2)

Claim 12. *eq.* (Generic Knapsack Problem) *with* $\mathsf{density}(p,r) = \frac{\mathsf{val}(p,r)}{\mathsf{weight}(p,r)}$, $\mathsf{val}(p,r) = \log p$, *and* $\mathsf{weight}(p,r) = \log I_{1/p}(r-1,\mu) - \log I_{1/p}(r,\mu)$ *is an upper bound to eq.* (Constrained Maximization 1).

Proof. The analysis is the same as in Claim 5, replacing $\Delta\mathsf{weight}_{\mu,\epsilon}$ with the weights defined here. □

Claim 13. *For* $p \in \mathbb{P}, r \in \mathbb{N}$ *and* $\mu \geq 2 \in \mathbb{N}$ *let* $\mathsf{density}(p,r) = \frac{\mathsf{val}(p,r)}{\mathsf{weight}(p,r)}$ *where* $\mathsf{val}(p,r) = \log p$, *and* $\mathsf{weight}(p,r) = \log I_{1/p}(r-1,\mu) - \log I_{1/p}(r,\mu)$ *for* $r \geq 1$. *Then* $\mathsf{density}(p,r)$ *is decreasing in* r *and* $\mathsf{density}(p,1)$ *is non-increasing in* p.

Proof. **Part 1:** $\mathsf{density}(p,r)$ **is decreasing in** r

$\mathsf{density}(p,r) = \frac{\log p}{\mathsf{weight}(p,r)}$ is decreasing in r iff $\mathsf{weight}(p,r)$ is increasing in r.

$$\mathsf{weight}(p,r) = -\log I_{1/p}(r,\mu) + \log I_{1/p}(r-1,\mu) = \log \frac{I_{1/p}(r-1,\mu)}{I_{1/p}(r,\mu)}$$

is decreasing in r if $\frac{I_{1/p}(r-1,\mu)}{I_{1/p}(r,\mu)}$ is decreasing in r. This is the case if for all r

$$\frac{I_{1/p}(r,\mu)}{I_{1/p}(r+1,\mu)} - \frac{I_{1/p}(r-1,\mu)}{I_{1/p}(r,\mu)} > 0$$

. Which is equivalent to showing that

$$I_{1/p}(r,\mu) \cdot I_{1/p}(r,\mu) - I_{1/p}(r-1,\mu) \cdot I_{1/p}(r-1,\mu) > 0 \tag{3.6}$$

The regularized beta function $I_x(a,b)$ is log convave for all $b > 1$ as shown in [104]. This implies that for $b > 1$ and all $\alpha, \beta > 0$ $I_x(a+\alpha,b) \cdot I_x(a+\beta,b) - I_x(a,b) \cdot I_x(a+\alpha+\beta,b) > 0$. Setting $a = r-1, b = \mu, \alpha = 1, \beta = 1$ this shows that eq. (3.6) holds and weight is incresing in r, and $\mathsf{density}$ is decreasing in r for all $\mu > 1$.

Part 2: $\mathsf{density}(p,1)$ is non-increasing in p.

$$\mathsf{density}(p,1) = \frac{\log p}{\mathsf{weight}(p,1)} = \frac{\ln p}{-\ln(1-(1-p^{-1})^{\mu})}$$

The derivative $\frac{d}{dp}\mathsf{density}(p,1)$ is non-negative iff:

$$-p^{-1} \cdot \ln(1-(1-p^{-1})^{\mu}) - \frac{\ln p \cdot \mu(1-p^{-1})^{\mu-1} \cdot p^{-2}}{1-(1-p^{-1})^{\mu}} \geq 0$$

Equivalently:

$$-\ln(1-(1-p^{-1})^{\mu}) - \frac{\ln p \cdot \mu(1-p^{-1})^{\mu-1}p^{-1}}{1-(1-p^{-1})^{\mu}} \geq 0$$

Set $x = 1 - \frac{1}{p}$, which increases over the range $(1/2, 1)$ as p increases in the range $[2, \infty)$. The requisite inequality for $x \in (1/2, 1)$ becomes:

$$-\ln(1-x^{\mu}) + \frac{\ln(1-x) \cdot \mu \cdot x^{\mu-1}(1-x)}{1-x^{\mu}} \geq 0$$

which, rearranging terms, holds true iff for $x \in (1/2, 1)$:

$$\frac{\ln(1-x^\mu)(1-x^\mu)}{\ln(1-x)(1-x)x^{\mu-1}} \leq \mu$$

In fact we can show this inequality holds true over all $x \in (0,1)$. Using the inequalities $\ln(1-x^\mu) \leq -x^\mu$ and $-\ln(1-x) \geq x$ we obtain:

$$\frac{\ln(1-x^\mu)(1-x^\mu)}{\ln(1-x)(1-x)x^{\mu-1}} \leq \frac{x^\mu(1-x^\mu)}{x(1-x)x^{\mu-1}} = \frac{1-x^\mu}{1-x} \leq \mu$$

□

Theorem 10 (Computational bound)**.** *Let k be the output of algorithm algorithm 3.1 on input λ, μ. Then for all $m \in \mathbb{N}$, all $N \geq 2^k$ and all μ-linear polynomials f, coprime with N, $\log_2 N \geq t(\lambda,\mu)$ and*

$$P_{\mathbf{x}\leftarrow[0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}$$

Proof. algorithm 3.1 is a greedy enumeration algorithm over pairs (p,r) following the enumeration strategy in claim 11. By claim 13, $\mathsf{density}$ satisfies the monotonicity conditions required for claim 11 and thus the enumeration algorithm enumerates pairs in order of non-increasing density. Thus, the algorithm outputs an upper bound to eq. (Generic Knapsack Problem) with density $\mathsf{density}(p,r) = \frac{\mathsf{val}(p,r)}{\mathsf{weight}(p,r)}$, $\mathsf{val}(p,r) = \log p$, and $\mathsf{weight}(p,r) = \log I_{1/p}(r-1,\mu) - \log I_{1/p}(r,\mu)$. By claim 12 this is an upper bound to eq. (Constrained Maximization 1) which in turn by theorem 8 gives a bound on $\log_2 t(\lambda,\mu)$. □

3.5.3 Computational Results

Using algorithm 3.1 we computed analytical bounds for all $\mu \in (1,50)$ for different values of λ. The precise bound for $\mu = 20$ and $\lambda = 120$ is:

$$\begin{aligned} 2^v = {} & 2^{36} \cdot 3^{20} \cdot 5^{11} \cdot 7^{8} \cdot 11^{5} \cdot 13^{5} \cdot 17^{4} \cdot 19^{3} \cdot 23^{3} \cdot 29^{2} \cdot 31^{2} \cdot 37^{2} \cdot 41^{2} \cdot 43^{2} \cdot 47^{2} \cdot 53^{2} \cdot 59 \cdot 61 \cdot 67 \\ & \cdot 71 \cdot 73 \cdot 79 \cdot 83 \cdot 89 \cdot 97 \cdot 101 \cdot 103 \cdot 107 \cdot 109 \cdot 113 \cdot 127 \cdot 131 \cdot 137 \cdot 139 \cdot 149 \cdot 151 \cdot 157 \cdot 163 \end{aligned}$$

Other results are presented in their logarithmic form in table 3.1. The results are significantly tighter than the analytical theorem 9. For $n = 20$ and $\lambda = 120$ the analytical theorem gives a value for $\log_2(N)$ of 3839 vs the computational which is 416. We also provide the open-source Python implementation of the algorithm on Github[2].

[2]`https://github.com/bbuenz/Composite-Schwartz-Zippel`

μ	$\lambda = 40$	$\lambda = 100$	$\lambda = 120$	$\lambda = 240$
1	40	100	120	240
2	57	130	156	290
3	67	148	175	328
4	79	169	197	359
5	86	187	212	386
6	97	200	234	415
7	107	214	244	435
8	113	227	260	459
9	122	237	277	483
10	133	252	289	500
11	139	263	301	523
12	148	276	315	540
13	152	291	331	565
14	160	304	344	576
15	168	314	354	600
16	178	323	366	616
17	186	335	381	634
18	193	347	391	653
19	198	356	407	664
20	207	368	416	679
21	216	378	429	695
22	222	389	437	718
23	228	402	448	732
24	233	411	464	749
25	241	420	472	758
26	248	432	481	772
27	256	438	492	792
28	264	452	506	806
29	275	460	516	820
30	278	469	527	831
50	419	662	736	1105

Table 3.1: Computationally determined values of $t(\lambda, \mu)$ such that for all $N \geq t(\lambda, \mu)$, $\mathbb{P}_{\mathbf{x} \leftarrow [0,m)^\mu}[f(\mathbf{x}) \equiv 0 \bmod N] \leq 2^{-\lambda} + \frac{\mu}{m}$ for different μ and different λ

Chapter 4

Authenticated datastructures from GUOs

This chapter presents new authenticated datastructures (ADS), which similar to the SNARKs from Chapter 2 (and polynomial commitment scheme DARK), are based on integer commitments in groups of unknown order. An *accumulator* [32] provides algorithms to produce a succinct binding commitment to a set of values and also to produce succinct membership and/or non-membership proofs for any value in the set, which can be publicly verified against the commitment. A *vector commitment* (VC) scheme [60] provides similar functionality, but for a vector of values, and contains an algorithm to produce a succinct proof for the value at any given position in the vector. Vector commitments are also implied by polynomial commitments [105], discussed in Chapter 2. Subvector commitments [109] are VCs where a multiple vector positions can be authenticated in a single succinct proof (i.e., sublinear in the size of the subset). Authenticated datastructures are *dynamic* if the commitment and authentication proofs can be updated efficiently as the datastructure is updated, and in particular, more efficiently than trivially recomputing the commitment from scratch for the updated datastructure. Otherwise we call it *static*. A *universal* accumulator is dynamic and supports both membership and non-membership proofs.

Authenticated data structures enable users to retrieve authenticated items from a remotely stored data structure. Beyond the applications to blockchain scalability discussed in the introduction (Section 1.1), they have been used and/or suggested for many applications within a similar realm, including accountable certificate management [50, 128], group signatures and anonymous credentials [57], computations on authenticated data [5], anonymous e-cash [146, 123], privacy-preserving data outsourcing [154], updatable signatures [137, 58], and decentralized bulletin boards [84, 88].

Authenticated datastructures are also useful for constructing SNARKs. We saw in Chapter 2 that polynomial commitments are used to build SNARKs from polynomial interactive oracle proofs (PIOPs), but vector commitments are sufficient for constructing SNARKs from PCPs and interactive oracle proofs (IOPs). As a review from Chapter 2, the earliest SNARK construction of Micali [122] showed how *probabilistically checkable proofs* (PCPs) can be used to construct succinct non-interactive arguments. In this construction the prover commits to a long PCP using a Merkle tree and then uses a random oracle to generate a few random query positions. The prover then verifiably opens the proof at the queried positions by providing Merkle inclusion paths. This technique has been generalized to the broader class of *interactive oracle proofs* (IOPs) [29]. In an IOP the prover sends multiple proof oracles to a verifier. The verifier uses these oracles to query a small subsets of the proof, and afterwards accepts or rejects the proof. If the proof oracle is instantiated with a Merkle tree commitment and the verifier is public coin, then an IOP can be compiled into a non-interactive proof secure in the random oracle model [29]. In particular, this compiler is used to build succinct non-interactive (zero-knowledge) proof of knowledge with a quasilinear prover and polylogarithmic verifier. Recent practical instantiations of proof systems from IOPs include Ligero [10], STARKs [22], and Aurora [27].

Merkle trees have two significant drawbacks when used to compile IOPs to SNARKs: first, position openings are not constant size, and second, the openings of several positions cannot be compressed into a single constant size proof (i.e. it is not a subvector commitment). A vector commitment with these properties would have dramatic benefits for reducing the size of the non-interactive proof compiled from an IOP.

4.1 Summary of results

Dynamic accumulators can be constructed from the strong RSA assumption in groups of unknown order (such as an RSA group or the class group) [18, 57, 112, 117], from bilinear maps [73, 56, 127], and from Merkle hash trees [121, 55]. These accumulators (with the exception of Merkle trees) naturally support batching of membership proofs, but not batching of non-membership proofs. Vector commitments based on similar techniques [114, 60, 113] have constant size openings, but large setup parameters.

Our main technical contributions are a set of batching and aggregation techniques for accumulators based on the strong RSA assumption in groups of unknown order. We show that using these batching techniques it is possible to construct a VC with constant size subvector openings and constant size public parameters. Previously, it was only known how to construct a VC with constant size subvector openings and public parameters *linear* in the length of the vector.

This has immediate implications for IOP compilers. The Merkle-tree IOP compiler outputs a non-interactive proof that is $O(\lambda q \log n)$ larger (additive blowup) than the original IOP communication, where q is the number of oracle queries, n is the maximum length[1] of the IOP proof oracles, and λ is the Merkle tree security parameter. When replacing the Merkle-tree in the IOP compiler with our new VC, we achieve only $O(r\lambda)$ blowup in proof size, independent of q and n, but dependent on the number of IOP rounds r. In the special case of a PCP there is a single round (i.e. $r = 1$). A similar result was recently demonstrated by Lai and Malvota [110] using the vector commitments of Catalano and Fiore (CF) [60], but the construction requires the verifier to access public parameters linear in n. It was not previously known how to achieve this with constant size public parameters. In fact, by observing that CF commitments admit succinct subvector openings, Lai and Malavolta constructed the smallest known transparent SNARKs (of constant size) using theoretical PCPs [107, 122], consisting of just 2 elements in a hidden order group and 240 additional

[1]In each round of an IOP, the prover prepares a message and sends the verifier a "proof oracle", which gives the verifier random read access to the prover's message. The "length" of the proof oracle is the length of this message.

bits of the PCP proof for 80-bit statistical security. Instantiating the group with class groups and targeting 100-bit security yields a proof of ≈ 540 bytes. However, the verifier must either use linear storage or perform linear work for each proof verification to generate the public proof parameters. In similar vein, we can use our new VCs to build the same non-interactive argument system, but with sublinear size parameters (in fact constant size). Under the same parameters our proofs are slightly larger, consisting of 5 group elements, a 128-bit integer, and the 240 bits of the PCP proof ($\approx 1.3KB$).

Targeting 100-bit security with class groups, replacing Merkle trees with our VCs would incur only 1 KB per round of a typical IOP. In Aurora [27], it was reported that Merkle proofs take up 154 KB of the 222 KB proof for a circuit of size 2^{20}. Our VCs would reduce the size of the proof to less than 100 KB, a 54% reduction. For STARKs, a recent benchmark indicates that the Merkle paths make up over 400 KB of the 600 KB proof for a circuit of 2^{52} gates [22]. With our VCs, under the same parameters the membership proofs would take up roughly 22 KB, reducing the overall proof size to approximately 222 KB, nearly a 63% reduction.

Furthermore, replacing Merkle trees with our new VCs maintains good performance for proof verification. Roughly, each Merkle path verification of a k-bit block is substituted with k modular multiplications of λ-bit integers. The performance comparison is thus $\log n$ hashes vs k multiplications, which is even an improvement for $k < \log n$. In the benchmarked STARK example, Merkle path verification comprises roughly 80% of the verification time.

4.2 Technical overview

Batching and aggregation. We use the term *batching* to describe a single action applied to n items instead of one action per item. For example a verifier can batch verify n proofs faster than n times verifying a single membership proof. *Aggregation* is a batching technique that is used when non-interactively combining n items to a single item. For example, a prover can aggregate n membership proofs to a single constant size proof.

Succinct proofs for hidden order groups. Wesolowski [163] recently introduced a constant sized and efficient to verify proof that a triple (u, w, t) satisfies $w = u^{2^t}$, where u and w are elements in a group $\mathbb{G}$ of unknown order. The proof extends to exponents that are not a power of two and still provides significant efficiency gains over direct verification by computation.

We expand on this technique to provide a new proof of knowledge of an exponent, which we call a *PoKE* proof. It is a proof that a computationally bounded prover knows the discrete logarithm between two elements in a group of unknown order. The proof is succinct in that the proof size and verification time is independent of the size of the discrete-log and has good soundness. We also generalize the technique to pre-images of homomorphisms from $\mathbb{Z}^q$ to $\mathbb{G}$ of unknown order. We prove security in the generic group model, where an adversarial prover operates over a generic group. Nevertheless, our extractor is classical and does not get to see the adversary's queries to the generic group oracles. We also rely on a succinct unstructured common reference string (CRS). Using the generic group model for extraction and relying on a CRS is necessary to bypass certain impossibility results for proofs of knowledge in groups of unknown order [16, 157].

We also extend the protocol to obtain a (honest verifier zero-knowledge) Σ-Protocol of DLOG in $\mathbb{G}$. This protocol is the first succinct Σ-protocol of this kind.

Distributed accumulator with batching. Next, we extend current RSA-based accumulators [57, 112] to create a universal accumulator for a distributed/decentralized setting where no single trusted accumulator manager exists and where updates are processed in batches. Despite this we show how membership and non-membership proofs can be efficiently aggregated. Moreover, items can efficiently be removed from the accumulator without a trapdoor or even knowledge of the accumulated set. Since the trapdoor is not required for our construction we can extend Lipmaa's [117] work on accumulators in groups of unknown order without a trusted setup by adding dynamic additions and deletions to the accumulator's functionality. As noted in Chapter 2, class groups of imaginary quadratic order are a candidate group of unknown order without a trusted setup [49].

Batching non-membership proofs. We next show how our techniques can be amplified to create a succinct and efficiently verifiable batch membership and batch non-membership proofs. We then use these batch proofs to create the first vector commitment construction with constant sized batch openings (recently called subvector commitments [109]) and $O(1)$ setup. This improves on previous work [60, 113] which required superlinear setup time and linear public parameter size. It also improves on Merkle tree constructions which have logarithmic sized non-batchable openings. The efficient setup also allows us to create sparse vector commitments which can be used as a key-value map commitment.

Soundness lower bounds in hidden order groups. Certain families of sigma protocols for a relation in a generic group of unknown order can achieve at most soundness 1/2 per challenge [16, 157]. Yet, our work gives sigma protocols in a generic group of unknown order that have negligible soundness error. This does not contradict the known impossibility result because our protocols involve a CRS, whereas the family of sigma protocols to which the 1/2 soundness lower bound applies do not have a CRS. Our results are significant as we show that it suffices to have a CRS containing two fresh random generic group generators to circumvent the soundness lower bound.

Note that we only prove how to extract a witness from a successful prover that is restricted to the generic group model. Proving extraction from an arbitrary prover under a falsifiable assumption is preferable and remains an open problem.

4.3 Preliminaries

Notation.

- $a \parallel b$ is the concatenation of two lists a, b
- $\vec{a}$ is a vector of elements and a_i is the ith component
- $[\ell]$ denotes the set of integers $\{0, 1, \ldots, \ell - 1\}$.
- $\mathsf{negl}(\lambda)$ is a negligible function of the security parameter λ

- $\mathsf{Primes}(\lambda)$ is the set of integer primes less than 2^λ
- $x \xleftarrow{\$} S$ denotes sampling a uniformly random element $x \in S$.
  $x \xleftarrow{\$} \mathcal{A}(\cdot)$ denotes the random variable that is the output of a randomized algorithm $\mathcal{A}$.
- $GGen(\lambda)$ is a randomized algorithm that generates a group of unknown order in a range $[a, b]$ such that a, b, and $a - b$ are all integers exponential in λ.

4.3.1 Assumptions

As a review from Chapter 2 (Section 2.8, Definition 3), the adaptive root assumption, introduced in [164], is as follows.

We say that the **adaptive root assumption** holds for $GGen$ if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $w \in \mathbb{G}$ and some st. Then, a random prime ℓ in $\mathsf{Primes}(\lambda)$ is chosen and $\mathcal{A}_1(\ell, \mathsf{st})$ outputs $w^{1/\ell} \in \mathbb{G}$. More precisely, for all efficient $(\mathcal{A}_0, \mathcal{A}_1)$:

$$\mathsf{Adv}^{\mathrm{ar}}_{(\mathcal{A}_0,\mathcal{A}_1)}(\lambda) := \Pr\left[u^\ell = w \neq 1 \ : \ \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(\mathbb{G}) \\ \ell \xleftarrow{\$} \mathsf{Primes}(\lambda) \\ u \xleftarrow{\$} \mathcal{A}_1(\ell, \mathsf{st}) \end{array} \right] \leq \mathsf{negl}(\lambda)\,.$$

The adaptive root assumption implies that the adversary can't compute the order of any non trivial element. For any element with known order the adversary can compute arbitrary roots that are co-prime to the order. This immediately allows the adversary to win the adaptive root game. For the group Z_N this means that we need to exclude $\{-1, 1\}$

We will also need the strong RSA assumption (Chapter 2, Assumption 2) for general groups of unknown order. The adaptive root and strong RSA assumptions are incomparable. The former states that it is hard to take a random root of a chosen group element, while the latter says that it is hard to take a chosen root of a random group element. In groups of unknown order that do not require a trusted setup the adversary A additionally gets access to *GGen*'s random coins. We review the definition here for convenience.

Definition 21 (Strong RSA assumption). *GGen satisfies the strong RSA assumption if for all efficient $\mathcal{A}$:*

$$\Pr\left[u^\ell = g \text{ and } \ell \text{ is an odd prime} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda),\ g \xleftarrow{\$} \mathbb{G}, \\ (u,\ell) \in \mathbb{G} \times \mathbb{Z} \xleftarrow{\$} \mathcal{A}(\mathbb{G}, g) \end{array}\right] \le \mathsf{negl}(\lambda)\,.$$

4.3.2 Generic group model for groups of unknown order

We will use the generic group model for groups of unknown order as defined by Damgard and Koprowski [72]. The group is parameterized by two integer public parameters A, B. The order of the group is sampled uniformly from $[A, B]$. The group $\mathbb{G}$ is defined by a random injective function $\sigma : \mathbb{Z}_{|\mathbb{G}|} \to \{0,1\}^\ell$, for some ℓ where $2^\ell \gg |\mathbb{G}|$. The group elements are $\sigma(0), \sigma(1), \ldots, \sigma(|\mathbb{G}| - 1)$. A *generic group algorithm* $\mathcal{A}$ is a probabilistic algorithm. Let $\mathcal{L}$ be a list that is initialized with the encodings given to $\mathcal{A}$ as input. The algorithm can query two generic group oracles:

- $\mathcal{O}_1$ samples a random $r \in \mathbb{Z}_{|\mathbb{G}|}$ and returns $\sigma(r)$, which is appended to the list of encodings $\mathcal{L}$.
- When $\mathcal{L}$ has size q, the second oracle $\mathcal{O}_2(i, j, \pm)$ takes two indices $i, j \in \{1, \ldots, q\}$ and a sign bit, and returns $\sigma(x_i \pm x_j)$, which is appended to $\mathcal{L}$.

Note that unlike Shoup's generic group model [153], the algorithm is not given $|\mathbb{G}|$, the order of the group $\mathbb{G}$.

4.3.3 Argument systems

An argument system for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is a triple of randomized polynomial time algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$, where Setup takes an (implicit) security parameter λ and outputs a common reference string (crs) pp. If the setup algorithm uses only public randomness we say that the setup is transparent and that the crs is unstructured. The prover $\mathcal{P}$ takes as input a statement $x \in \mathcal{X}$, a witness $w \in \mathcal{W}$, and the crs pp. The verifier $\mathcal{V}$ takes as input pp and x and after interaction with $\mathcal{P}$ outputs 0 or 1. We denote the transcript between the prover and verifier by $\langle \mathcal{V}(\mathsf{pp}, x), \mathcal{P}(\mathsf{pp}, x, w)\rangle$ and write $\langle \mathcal{V}(\mathsf{pp}, x), \mathcal{P}(\mathsf{pp}, x, w)\rangle = 1$ to indicate that the verifier accepted the transcript. If $\mathcal{V}$ uses only public randomness we say that the protocol is public coin.

Definition 22 (Completeness)**.** *We say that an argument system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is **complete** if for all* $(x, w) \in \mathcal{R}$*:*

$$\Pr\big[\,\langle \mathcal{V}(\mathsf{pp}, x), \mathcal{P}(\mathsf{pp}, x, w)\rangle = 1 : \mathsf{pp} \overset{\$}{\leftarrow} \mathsf{SetupSetup}(\lambda)\big] = 1.$$

We now define soundness and knowledge extraction for our protocols. The adversary is modeled as two algorithms $\mathcal{A}_0$ and $\mathcal{A}_1$, where $\mathcal{A}_0$ outputs the instance $x \in \mathcal{X}$ *after* Setup is run, and $\mathcal{A}_1$ runs the interactive protocol with the verifier using a st output by A_0. In slight deviation from the soundness definition used in statistically sound proof systems, we do not universally quantify over the instance x (i.e. we do not require security to hold for all input instances x). This is due to the fact that in the computationally-sound setting the instance itself may encode a trapdoor of the crs pp (e.g. the order of a group of unknown order), which can enable the adversary to fool a verifier. Requiring that an efficient adversary outputs the instance x prevents this. In our soundness definition the adversary $\mathcal{A}_1$ succeeds if he can make the verifier accept when no witness for x exists. For the stronger *argument of knowledge* definition we require that an extractor with access to $\mathcal{A}_1$'s internal state can extract a valid witness whenever $\mathcal{A}_1$ is convincing. We model this by enabling the extractor to rewind $\mathcal{A}_1$ and reinitialize the verifier's randomness.

Definition 23 (Arguments (of Knowledge))**.** *We say that an argument system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is sound if for all poly-time adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\begin{array}{ll} \langle \mathcal{V}(\mathsf{pp}, x), \mathcal{A}_1(\mathsf{pp}, x, \mathsf{st})\rangle = 1 & \mathsf{pp} \overset{\$}{\leftarrow} \mathsf{Setup}(1^\lambda) \\ \text{and } \nexists w \quad (x, w) \in \mathcal{R} : & (x, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}) \end{array}\right] = \mathsf{negl}(\lambda)\,.$$

Additionally, the argument system is an argument of knowledge if for all poly-time adversaries $\mathcal{A}_1$ there exists a poly-time extractor Ext *such that for all poly-time adversaries $\mathcal{A}_0$:*

$$\Pr\left[\begin{array}{ll} \langle \mathcal{V}(\mathsf{pp}, x), \mathcal{A}_1(\mathsf{pp}, x, \mathsf{st})\rangle = 1 & \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ \text{and } (x, w') \notin \mathcal{R} : & (x, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}) \\ & w' \xleftarrow{\$} \mathsf{Ext}(\mathsf{pp}, x, \mathsf{st}) \end{array}\right] = \mathsf{negl}(\lambda)\,.$$

Any argument of knowledge is also sound. In some cases we may further restrict $\mathcal{A}$ in the security analysis, in which case we would say the system is an argument of knowledge for a restricted class of adversaries. For example, in this work we construct argument systems for relations that depend on a group $\mathbb{G}$ of unknown order. In the analysis we replace $\mathbb{G}$ with a generic group and restrict $\mathcal{A}$ to a generic group algorithm that interacts with the oracles for this group. For simplicity, although slightly imprecise, we say the protocol is an *argument of knowledge in the generic group model.* Groth [96] recently proposed a SNARK system for arbitrary relations that is an argument of knowledge in the generic group model in a slightly different sense, where the generic group is used as part of the construction rather than the relation and the adversary is a generic group algorithm with respect to this group generated by the setup.

Definition 24 (Non interactive arguments)**.** *A* ***non-interactive argument system*** *is an argument system where the interaction between $\mathcal{P}$ and $\mathcal{V}$ consists of only a single round. We then write the prover $\mathcal{P}$ as $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{pp}, x, w)$ and the verifier as $\{0,1\} \leftarrow \mathsf{Vf}(\mathsf{pp}, x, \pi)$.*

The Fiat-Shamir heuristic [78] and its generalization to multi-round protocols [29] can be used to transform public coin argument systems to non-interactive systems.

4.4 Succinct proofs for hidden order groups

In this section we present several new succinct proofs in groups of unknown order. The proofs build on a proof of exponentiation recently proposed by Wesolowski [163] in the context of verifiable delay functions [37]. We show that the Wesolowski proof is a *succinct* proof of knowledge of a discrete-log in a group of unknown order. We then derive a *succinct* zero-knowledge argument of knowledge for a discrete-log relation, and more generally for knowledge of the inverse of a homomorphism $h : \mathbb{Z}^n \to \mathbb{G}$, where $\mathbb{G}$ is a group of unknown order. Using the Fiat-Shamir heuristic, the non-interactive version of this protocol is a special purpose SNARK for the pre-image of a homomorphism.

4.4.1 A succinct proof of exponentiation

Let $\mathbb{G}$ be a group of unknown order. Let $[\ell] := \{0, 1, \ldots, \ell - 1\}$ and let $\mathsf{Primes}(\lambda)$ denote the set of odd prime numbers in $[0, 2^\lambda]$. We begin by reviewing Wesolowski's (non-ZK) proof of exponentiation [163] in the group $\mathbb{G}$. Here both the prover and verifier are given (u, w, x) and the prover wants to convince the verifier that $w = u^x$ holds in $\mathbb{G}$. That is, the protocol is an argument system for the relation

$$\mathcal{R}_{\mathsf{PoE}} = \left\{ \big((u, w \in \mathbb{G}, x \in \mathbb{Z});\ \perp \big)\ :\ w = u^x \in \mathbb{G} \right\}.$$

The verifier's work should be much less than computing u^x by itself. Note that $x \in \mathbb{Z}$ can be much larger than $|\mathbb{G}|$, which is where the protocol is most useful. The protocol works as follows:

<u>Protocol PoE (Proof of exponentiation) for $\mathcal{R}_{\mathsf{PoE}}$ [163]</u>

Params: $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$; Inputs: $u, w \in \mathbb{G},\ x \in \mathbb{Z}$; Claim: $u^x = w$

1. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to prover.
2. Prover computes the quotient $q = \lfloor x/\ell \rfloor \in \mathbb{Z}$ and residue $r \in [\ell]$ such that $x = q\ell + r$.

   Prover sends $Q \leftarrow u^q \in \mathbb{G}$ to the Verifier.
3. Verifier computes $r \leftarrow (x \bmod \ell) \in [\ell]$ and accepts if $Q^\ell u^r = w$ holds in $\mathbb{G}$.

The protocol above is a minor generalization of the protocol from [163] in that we allow an arbitrary exponent $x \in \mathbb{Z}$, where as in [163] the exponent was restricted to be a power of two. This does not change the soundness property captured in the following theorem, whose proof is given in [163, Prop. 2] (see also [40, Thm. 2]) and relies on the adaptive root assumption for *GGen*.

Theorem 11 (Soundness PoE [163])**.** *Protocol PoE is an argument system for Relation $\mathcal{R}_{\mathsf{PoE}}$ with negligible soundness error, assuming the adaptive root assumption holds for GGen.*

For the protocol to be useful the verifier must be able to compute $r = x \bmod \ell$ faster than computing $u^x \in \mathbb{G}$. The original protocol presented by Wesolowski assumed that $x = 2^T$ is a power of two, so that computing $x \bmod \ell$ requires only $\log(T)$ multiplications in $\mathbb{Z}_\ell$ whereas computing u^x requires T group operations.

For a general exponent $x \in \mathbb{Z}$, computing $x \bmod \ell$ takes $O((\log x)/\lambda)$ multiplications in $\mathbb{Z}_\ell$. In contrast, computing $u^x \in \mathbb{G}$ takes $O(\log x)$ group operations in $\mathbb{G}$. Hence, for the current groups of unknown order, computing u^x takes λ^3 times as long as computing $x \bmod \ell$. Concretely, when ℓ is a 128 bit integer, a multiplication in $\mathbb{Z}_\ell$ is approximately 5000 time faster than a group operation in a 2048-bit RSA group. Hence, the verifier's work is much less than computing $w = u^x$ in $\mathbb{G}$ on its own.

Note that the adaptive root assumption is not only a sufficient security requirement but also necessary. In particular it is important that no known order elements are in the group $\mathbb{G}$. Assume for example that $-1 \in \mathbb{G}$ such that $(-1)^2 = 1 \in \mathbb{G}$. If $g^x = y$ then an adversary can succeed in $\mathsf{PoE}(g, -y, x)$ by setting $Q' \leftarrow -1 \cdot g^{\lfloor x/\ell \rfloor}$. It is, therefore, important to not directly use the multiplicative RSA group $\mathbb{G} := (\mathbb{Z}/N)^*$ but rather $\mathbb{G}^+ := \mathbb{G}/\{-1, 1\}$ as described in [40].

The PoE protocol can be generalized to a relation involving any homomorphism $\phi : \mathbb{Z}^n \to \mathbb{G}$ for which the adaptive root assumption holds in $\mathbb{G}$. The details of this generalization are discussed in Section 4.9.1.

4.4.2 A succinct proof of knowledge of a discrete-log

We next show how the protocol PoE can be adapted to provide an argument of knowledge of discrete-log, namely an argument of knowledge for the relation:

$$\mathcal{R}_{\mathsf{PoKE}} = \big\{\big((u, w \in \mathbb{G});\ x \in \mathbb{Z}\big)\ :\ w = u^x \in \mathbb{G}\big\}.$$

The goal is to construct a protocol that has communication complexity that is much lower than simply sending x to the verifier. As a stepping stone we first provide an argument of knowledge for a modified PoKE relation, where the base $u \in \mathbb{G}$ is fixed and encoded in a CRS. Concretely let CRS consist of the unknown-order group $\mathbb{G}$ and the generator g. We construct an argument of knowledge for the following relation:

$$\mathcal{R}_{\mathsf{PoKE}^*} = \big\{\big(w \in \mathbb{G};\ x \in \mathbb{Z}\big)\ :\ w = g^x \in \mathbb{G}\big\}.$$

The argument modifies the $\mathsf{PoE\ Protocol}$ in that x is not given to the verifier, and the remainder $r \in [\ell]$ is sent from the prover to the verifier:

$\mathsf{Protocol\ PoKE}^*$ (Proof of knowledge of exponent) for Relation $\mathcal{R}_{\mathsf{PoKE}^*}$

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda),\ g \in \mathbb{G}$; Inputs: $w \in \mathbb{G}$; Witness: $x \in \mathbb{Z}$; Claim: $g^x = w$

1. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.
2. Prover computes the quotient $q \in \mathbb{Z}$ and residue $r \in [\ell]$ such that $x = q\ell + r$. Prover sends the pair $(Q \leftarrow g^q,\ r)$ to the Verifier.
3. Verifier accepts if $r \in [\ell]$ and $Q^\ell g^r = w$ holds in $\mathbb{G}$.

Here the verifier does not have the witness x, but the prover additionally sends $r := (x \bmod \ell)$ along with Q in its response to the verifier's challenge. Note that the verifier no longer computes r on its own, but instead relies on the value from the prover. We will demonstrate an extractor that extracts the witness $x \in \mathbb{Z}$ from a successful prover, and prove that this extractor succeeds with overwhelming probability against a generic group prover. In fact, in the next section we will present a generalization of $\mathsf{Protocol\ PoKE}^*$ to group representations in terms of bases $\{g_i\}_{i=1}^n$ included in the CRS, i.e. a proof of knowledge of an integer vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\prod_i g_i^{x_i} = w$. We will prove that this protocol is an argument of knowledge against a generic group adversary. The security of $\mathsf{Protocol\ PoKE}^*$ above follows as a special case. Hence, the following theorem is a special case of Theorem 17 below.

Theorem 12. $\mathsf{Protocol\ PoKE}^*$ *is an argument of knowledge for relation* $\mathcal{R}_{\mathsf{PoKE}^*}$ *in the generic group model.*

An attack. $\mathsf{Protocol\ PoKE}^*$ requires the discrete logarithm base g to be encoded in the CRS. When this protocol is applied to a base freely chosen by the adversary it becomes insecure. In other words, $\mathsf{Protocol\ PoKE}^*$ is not a secure protocol for the relation $\mathcal{R}_{\mathsf{PoKE}}$.

To describe the attack, let g be a generator of $\mathbb{G}$ and let $u = g^x$ and $w = g^y$ where $y \neq 1$ and x does not divide y. Suppose that the adversary knows both x and y but not the discrete log of w base u. Computing an integer discrete logarithm of w base u is still difficult in a generic group (as follows from Lemma 23), however an efficient adversary can nonetheless succeed in fooling the verifier as follows. Since the challenge ℓ is co-prime with x with overwhelming probability, the adversary can compute $q, r \in \mathbb{Z}$ such that $q\ell + rx = y$. The adversary sends $(Q = g^q,\ r)$ to the verifier, and the verifier checks that indeed $Q^\ell u^r = w$. Hence, the verifier accepts despite the adversary not knowing the discrete log of w base u.

This does not qualify as an "attack" when $x = 1$, or more generally when x divides y, since then the adversary does know the discrete logarithm y/x such that $u^{y/x} = w$.

Extending PoKE for general bases. To obtain a protocol for the relation $\mathcal{R}_{\mathsf{PoKE}}$ we start by modifying protocol PoKE^* so that the prover first sends $z = g^x$, for a fixed base g, and then executes two PoKE^* style protocols, one base g and one base u, in parallel, showing that the discrete logarithm of w base u equals the one of z base g. We show that the resulting protocol is a secure argument of knowledge (in the generic group model) for the relation $\mathcal{R}_{\mathsf{PoKE}}$. The transcript of this modified protocol now consists of two group elements instead of one.

Protocol PoKE (Proof of knowledge of exponent)

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda),\ g \in \mathbb{G}$; Inputs: $u, w \in \mathbb{G}$; Witness: $x \in \mathbb{Z}$;

Claim: $u^x = w$

1. Prover sends $z = g^x \in \mathbb{G}$ to the verifier.
2. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.
3. Prover finds the quotient $q \in \mathbb{Z}$ and residue $r \in [\ell]$ such that $x = q\ell + r$. Prover sends $Q = u^q$ and $Q' = g^q$ and r to the Verifier.
4. Verifier accepts if $r \in [\ell]$, $Q^\ell u^r = w$, and $Q'^\ell g^r = z$.

The intuition for the security proof is as follows. The extractor first uses the same extractor for Protocol PoKE* (specified in Theorem 17) to extract the discrete logarithm x of z base g. It then suffices to argue that this extracted discrete logarithm x is a *correct* discrete logarithm of w base u. We use the adaptive root assumption to argue that the extracted x is a correct discrete logarithm of w base u.

We can optimize the protocol to bring down the proof size back to a single group element. We do so in the protocol PoKE2 below by adding one round of interaction. The additional round has no effect on proof size after making the protocol non-interactive using Fiat-Shamir.

Protocol PoKE2 (Proof of knowledge of exponent)

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$; Inputs: $u, w \in \mathbb{G}$; Witness: $x \in \mathbb{Z}$; Claim: $u^x = w$

1. Verifier sends $g \overset{\$}{\leftarrow} \mathbb{G}$ to the Prover.
2. Prover sends $z \leftarrow g^x \in \mathbb{G}$ to the verifier.
3. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$ and $\alpha \overset{\$}{\leftarrow} [0, 2^\lambda)$.
4. Prover finds the quotient $q \in \mathbb{Z}$ and residue $r \in [\ell]$ such that $x = q\ell + r$. Prover sends $Q = u^q g^{\alpha q}$ and r to the Verifier.
5. Verifier accepts if $r \in [\ell]$ and $Q^\ell u^r g^{\alpha r} = w z^\alpha$.

The intuition for the security proof is the same as for Protocol PoKE, but we additionally show that (in the generic group model) a similar extraction argument holds when the prover instead sends $Q \leftarrow u^q g^q$ and r such that $Q^\ell u^r g^r = wz$. The extraction argument uses the fact that with overwhelming probability the generic adversary did not obtain g from any of its group oracle queries prior to forming w and therefore the adversary's representation of w does not contain g as a base with a non-zero exponent. The extractor is able to obtain an exponent x such that $(gu)^x = wz$. This alone does not yet imply that $u^x = w$, however if the prover sends Q, r such that $Q^\ell u^r g^{\alpha r} = wz^\alpha$, then the extractor obtains a fixed x such that $(g^\alpha u)^x = wz^\alpha$ with high probability over the random choice of α. This implies that either $u^x = w$ or w/u^x is an element of low order, which breaks the adaptive root assumption. We summarize this in the following theorem. See Section 4.11 for the proof.

Theorem 13 (PoKE Argument of Knowledge)**.** *Protocol PoKE and Protocol PoKE2 are arguments of knowledge for relation $\mathcal{R}_{\mathsf{PoKE}}$ in the generic group model.*

The PoKE argument of knowledge can be extended to an argument of knowledge for the pre-image of a homomorphism $\phi : \mathbb{Z}^n \rightarrow \mathbb{G}$. This is included in Section 4.9.2.

We can also construct a (honest-verifier) zero-knowledge version of the PoKE argument of knowledge protocol using a method similar to the classic Schnorr Σ-protocol for hidden order groups. This is covered in Section 4.9.4.

4.4.3 Aggregating Knowledge of Co-prime Roots

Unlike exponents, providing a root of an element in a hidden order group is already succinct (it is simply a group element). There is a simple aggregation technique for providing a succinct proof of knowledge for multiple *co-prime* roots $x_1, ..., x_n$ simultaneously. This is useful for aggregating PoKE proofs.

When the roots are all for the same element α then the witness is trivially a root α^{1/x^*} where $x^* = x_1 \cdots x_n$. From this witness one can publicly extract the x_ith root of α for each i. We show a method where the elements need not be the same, i.e. the witness is a list of elements $w_1, ..., w_n$ for public elements $\alpha_1, ..., \alpha_n$ and public integers $x_1, ..., x_n$ such that $w_i^{x_i} = \alpha_i$ for each i and $gcd(x_i, x_j) = 1 \forall i, j \in [1, n], i \neq j$. The size of the proof is still a single element.

Concretely the PoKCR protocol is a proof for the relation:

$$\mathcal{R}_{\mathsf{PoKCR}} = \left\{ \left(\alpha \in \mathbb{G}^n;\ \mathbf{x} \in \mathbb{Z}^n \right)\ :\ w = \phi(\mathbf{x}) \in \mathbb{G} \right\}.$$

The proof is the product of witnesses, $w \leftarrow w_1 \cdots w_n$. From this product and the public x_i's and α_i's it is possible to extract an x_ith root of each α_i. (This is not necessarily the same as w_i as roots are not unique). Moreover, the verification algorithm does not need to run this extraction procedure in full, it only needs to check that $w^{x^*} = \prod_i \alpha_i^{x^*/x_i}$. This equation can be verifier with $O(n \log n)$ group exponentiations with exponents of size at most $\max_i |x_i|$ using the optimized recursive **MultiExp** algorithm shown below.

<u>Protocol PoKCR for Relation $\mathcal{R}_{\mathsf{PoKCR}}$</u>

Input: $\mathbb{G}$, $\alpha_1, ..., \alpha_n \in \mathbb{G}$, $x_1, ..., x_n \in \mathbb{Z}$ s.t. $gcd(x_1, ..., x_n) = 1$;

Witness: $\mathbf{w} \in \mathbb{G}^n$ s.t. $w_i^{x_i} = \alpha_i$

1. Prover sends $w \leftarrow \prod_{i=1}^n w_i$ to the Verifier.
2. Verifier computes $x^* \leftarrow \prod_{i=1}^n x_i$, and $y \leftarrow \prod_{i=1}^n \alpha_i^{x^*/x_i}$ using $\mathbf{MultiExp}(n, \vec{\alpha}, \vec{x})$. Verifier accepts if $w^{x^*} = y$.

```
MultiExp(n, $\vec{\alpha}$, $\vec{x}$):
    1. if $n = 1$ return $\alpha$
    2. $\vec{\alpha_L} \leftarrow (\alpha_1, ..., \alpha_{n/2})$; $\vec{\alpha_R} \leftarrow (\alpha_{n/2+1}, ..., \alpha_n)$
    3. $\vec{x_L} \leftarrow (x_1, ..., x_{n/2})$; $\vec{x_R} \leftarrow (x_{n/2+1}, ..., x_n)$
    4. $x_L^* \leftarrow x_1 \cdots x_{n/2}$; $x_R^* \leftarrow x_{n/2+1} \cdots x_n$
    5. $L \leftarrow$ MultiExp$(n/2, \vec{\alpha_L}, \vec{x_L})$; $R \leftarrow$ MultiExp$(n/2, \vec{\alpha_R}, \vec{x_R})$
    6. return $L^{x_R^*} \cdot R^{x_L^*}$
```

Lemma 21. *Protocol PoKCR is an argument of knowledge for Relation $\mathcal{R}_{\mathsf{PoKCR}}$.*

Proof. We show that given any w such that $w^{x^*} = y = \prod_{i=1}^n \alpha_i^{x^*/x_i}$ it is possible to compute directly an x_ith root of α_i for all i. For each i and $j \neq i$ let $z_{ij} = x^*/(x_i x_j)$. For each

i, let $A_j = \prod_{i \neq j} \alpha_i^{z_{ij}}$, then we can express $y = A_j^{x_i} \alpha_i^{x^*/x_i}$. This shows that the element $u = w^{(x^*/x_i)} A_j^{-1}$ is an x_ith root of $\alpha_i^{x^*/x_i}$. Since $gcd(x^*/x_i, x_i) = 1$, there exist Bezout coefficients a, b such that $a(x^*/x_i) + bx_i = 1$. Finally, $u^a \alpha_i^b$ is an x_ith root of α_i as $(u^a \alpha_i^b)^{x_i} = \alpha_i^{(ax^*/x_i)+bx_i} = \alpha_i$. $\square$

Non-interactive proofs All of the protocols can be made non-interactive using the standard Fiat-Shamir transform. In the Fiat-Shamir transform, the prover non-interactively builds a simulated transcript of the protocol by replacing each of the verifier's challenges with a hash of the protocol transcript preceding the challenge using a collision-resistant hash function H as a heuristic substitute for a random oracle. In our protocols, the verifier's challenges are sampled from $\mathsf{Primes}(\lambda)$ and $\mathbb{G}$. Therefore, the non-interactive version must involve a canonical mapping of the output seed σ of the random oracle to a random prime or element of $\mathbb{G}$. Furthermore, it is important that hashing to an element $g \in \mathbb{G}$ does not reveal the discrete log relation between g and any another element (i.e. $g \leftarrow u^\sigma$ is not secure). The simplest way to map σ to a prime in $\mathsf{Primes}(\lambda)$ is find the smallest integer i such that the first λ bits of $H(\sigma, i)$ is prime. More efficient methods are described in Section 4.8. It is these non-interactive, succinct, and efficiently verifiable proofs that are most useful for the applications discussed later in this paper. Section 4.12 summarizes the non-interactive proofs that will be used later.

Aggregating PoKE proofs Several non-interactive PoE/PoKE/PoKE2 proofs can be aggregated using the PoKCR protocol. The value Q sent to the verifier in this proof is the ℓth root of yg^{-r}. As long as the primes sampled in each proof instance are distinct then these proofs (specifically the values Q_i) are a witness for an instance of PoKCR. Since the primes are generated by hashing the inputs to the proof they need not be included in the proof.

4.5 Trapdoorless Universal Accumulator

In this section we describe a number of new techniques for manipulating accumulators built from the strong RSA assumption in a group of unknown order. We show how to efficiently

λ: Security Parameter
t: A discrete time counter
A_t: Accumulator value at time t
S_t: The set of elements currently accumulated
w_x^t, u_x^t: Membership and non-membership proofs
pp: Public parameters implicitly available to all methods
upmsg: Information used to update proofs
$\mathsf{Setup}(\lambda, z) \to \mathsf{pp}, A_0$ Generate the public parameters
$\mathbf{Add}(A_t, x) \to \{A_{t+1}, \mathbf{upmsg}\}$ Update the accumulator
$\mathbf{Del}(A_t, x) \to \{A_{t+1}, \mathbf{upmsg}\}$ Delete a value from the accumulator
$\mathbf{MemWitCreate}(A_t, S, x) \to w_x^t$ Create an membership proof
$\mathbf{NonMemWitCreate}(A_t, S, x) \to u_x^t$ Create a non-membership proof
$\mathbf{MemWitUp}(A_t, w_x^t, x, \mathbf{upmsg}) \to w_x^{t+1}$ Update an membership proof
$\mathbf{NonMemWitUp}(A_t, w_x^t, x, \mathbf{upmsg}) \to u_x^{t+1}$ Update a non-membership proof
$\mathbf{VerMem}(A_t, x, w_x^t) \to \{0, 1\}$ Verify membership proof
$\mathbf{VerNonMem}(A_t, x, u_x^t) \to \{0, 1\}$ Verify non-membership proof

Figure 4.1: A trapdoorless universal accumulator.

remove elements from the accumulator, how to use the proof techniques from Section 4.4 to give short membership proofs for multiple elements, and how to non-interactively aggregate inclusion and exclusion proofs. All our techniques are geared towards the setting where there is no trusted setup. We begin by defining what an accumulator is and what it means for an accumulator to be secure.

Our presentation of a trapdoorless universal accumulator mostly follows the definitions and naming conventions of [15]. Figure 4.1 summarizes the accumulator syntax and list of associated operations. One notable difference in our syntax is the presence of a common reference string pp generated by the Setup algorithm in place of private/public keys.

The security definition we follow [117] formulates an *undeniability* property for accumulators. For background on how this definition relates to others that have been proposed see [15], which gives generic transformations between different accumulators with different properties and at different security levels.

The following definition states that an accumulator is secure if an adversary cannot construct an accumulator, an element x and a valid membership witness w_x^t and a non-membership witness u_x^t where w_x^t shows that x is in the accumulator and u_x^t shows that it is not. Lipmaa [117] also defines undeniability without a trusted setup. In that definition the adversary has access to the random coins used by Setup.

Definition 25 (Accumulator Security (Undeniability)).

$$\Pr\left[\begin{array}{l} pp, A_0 \in \mathbb{G} \xleftarrow{\$} \mathit{Setup}(\lambda) \\ (A, x, w_x, u_x) \xleftarrow{\$} \mathcal{A}(pp, A_0) \\ \textbf{\textit{VerMem}}(A, x, w_x^t) \wedge \textbf{\textit{VerNonMem}}(A, x, u_x^t) \end{array}\right] = \mathsf{negl}(\lambda)$$

4.5.1 Accumulator construction

Several sub-procedures that are used heavily in the construction are summarized below. **Bezout**(x,y) refers to a sub-procedure that outputs Bezout coefficients $a, b \in \mathbb{Z}$ for a pair of co-prime integers x, y (i.e. satisfying the relation $ax + by = 1$). **ShamirTrick** uses Bezout coefficient's to compute an (xy)-th root of a group element g from an x-th root of g and a yth root of g. **RootFactor** is a procedure that given an element $y = g^x$ and the factorization of the exponent $x = x_1 \cdots x_n$ computes an x_i-th root of y for all $i = 1, \ldots, n$ in total time $O(n \log(n))$. Naively this procedure would take time $O(n^2)$. It is related to the **MultiExp** algorithm described earlier and was originally described by [147].

```
ShamirTrick($w_1, w_2, x, y$): [152]
    1. if $w_1^x \neq w_2^y$ return $\perp$
    2. $a, b \leftarrow$ Bezout($x, y$)
    3. return $w_1^b w_2^a$

H_prime($x$):
    1. $y \leftarrow \mathsf{H}(x)$
    2. while $y$ is not odd prime:
    3.     $y \leftarrow \mathsf{H}(y)$
    4. return $y$

RootFactor($g, x_1, \ldots, x_n$):
    1. if $n = 1$ return g
    2. $n' \leftarrow \lfloor \frac{n}{2} \rfloor$
    3. $g_L \leftarrow g^{\prod_{j=1}^{n'} x_j}$
    4. $g_R \leftarrow g^{\prod_{j=n'+1}^{n} x_j}$
    5. $L \leftarrow$ RootFactor($g_R, x_1, \ldots, x_{n'}$)
    6. $R \leftarrow$ RootFactor($g_L, x_{n'+1}, \ldots, x_n$)
    7. return $L \parallel R$
```

Groups of unknown order The accumulator requires a procedure $GGen(\lambda)$ which samples a group of unknown order in which the strong root assumption (Definition 21) holds.

One can use the quotient group $(\mathbb{Z}/N)^*/\{-1,1\}$, where N is an RSA modulus, which may require a trusted setup to generate the modulus N. Alternatively, one can use a class group which eliminates the trusted setup. Note that the adaptive root assumption requires that these groups have no known elements of low order, and hence the group $(\mathbb{Z}/N)^*$ is not suitable because $(-1) \in (\mathbb{Z}/N)^*$ has order two [40]. Given an element of order two it is possible to convince a PoE-verifier that $g^x = -y$ when in fact $g^x = y$.

The basic RSA accumulator. We review he classic RSA accumulator [57, 117] below, omitting all the procedures that require trapdoor information. All accumulated values are odd primes. If the strong RSA assumption (Definition 21) holds in $\mathbb{G}$, then the accumulator satisfies the undeniability definition [117].

The core procedures for the basic dynamic accumulator are the following:

- Setup generates a group of unknown order and initializes the group with a generator of that group.
- **Add** takes the current accumulator A_t, an element from the odd primes domain, and computes $A_{t+1} = A_t$.
- **Del** does not have such a trapdoor and therefore needs to reconstruct the set from scratch. The **RootFactor** algorithm can be used for pre-computation. Storing 2^k elements and doing $n \cdot k$ work, the online removal will only take $(1 - \frac{1}{2}^k) \cdot n$ steps.
- A membership witness is simply the accumulator without the aggregated item.
- A membership non-witness, proposed by [112], uses the fact that for any $x \notin S$, $\gcd(x, \prod_{s \in S} s) = 1$. The Bezout coefficients $(a, b) \leftarrow \mathbf{Bezout}(x, \prod_{s \in S} s)$ are therefore a valid membership witness. The actual witness is the pair (a, g^b) which is short because $|a| \approx |x|$.
- Membership and non-membership witnesses can be efficiently updated as in [112]

```
Setup(λ):
    1. $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$
    2. $g \xleftarrow{\$} \mathbb{G}$
    3. return $\mathbb{G}, g$

Add($A_t, S, x$):
    1. if $x \in S$ : return $A_t$
    2. else :
    3.     $S \leftarrow S \cup \{x\}$
    4.     upmsg $\leftarrow x$
    5.     return $A_t^x$, upmsg

Del($A_t, S, x$):
    1. if : $x \notin S$ : return $A_t$
    2. else :
    3.     $S \leftarrow S \setminus \{x\}$
    4.     $A_{t+1} \leftarrow g^{\prod_{s \in S} s}$
    5.     upmsg $\leftarrow \{x, A_t, A_{t+1}\}$
    6.     return $A_{t+1}$, upmsg

MemWitCreate($A, S, x$) :
    1. $w_x^t \leftarrow g^{\prod_{s \in S, s \neq x} s}$
    2. return $w_x^t$

NonMemWitCreate($A, S, x$) :
    1. $s^* \leftarrow \prod_{s \in S} s$
    2. $a, b \leftarrow$ Bezout($s^*, x$)
    3. $B \leftarrow g^b$
    4. return $u_x^t \leftarrow \{a, B\}$

VerMem($A, w_x, x$) :
    1. return 1 if $(w_x)^x = A$

VerNonMem($A, u_x, x$) :
    1. $\{a, B\} \leftarrow u_x$
    2. return 1 if $A^a B^x = g$
```

Theorem 14 (Security accumulator [117]). *Assume that the strong RSA assumption (Definition 21) holds in $\mathbb{G}$. Then the accumulator satisfies undeniability (Definition 25) and is therefore secure.*

Proof. We construct an $\mathcal{A}_{RSA}$ that given an $\mathcal{A}_{Acc}$ for the accumulator breaks the strong RSA assumption. $\mathcal{A}_{RSA}$ receives a group $\mathbb{G} \leftarrow GGen(\lambda)$ and a challenge $g \xleftarrow{\$} \mathbb{G}$. We now run $\mathcal{A}_{Acc}$ on input $\mathbb{G}$ and $A_0 = g$. $\mathcal{A}_{Acc}$ returns a tuple (A, x, w_x, u_x) such that **VerMem**$(A, x, w_x) = 1$ and **VerNonMem**$(A, x, u_x) = 1$. $\mathcal{A}_{RSA}$ parses $(a, B) = u_x$ and computes $B \cdot (w_x)^a$ as the xth root of g. x is an odd prime by definition and $(B \cdot w_x^a)^x = B^x \cdot A^b = g$. This contradicts the strong RSA assumption and thus shows that the accumulator construction satisfies undeniability. □

Updating membership witnesses [57, 112] Updating membership witnesses when an item is added simply consists of adding the item to the witness which itself is an accumulator. The membership witness is an xth root of the accumulator A_t. After removal of $\hat{x}$, A_{t+1} is an $\hat{x}$th root of A_t. We can use the **ShamirTrick** to compute an $x \cdot \hat{x}$th root of A_t which corresponds to the updated witness. Updating the non-membership witnesses is done by computing the Bezout coefficients between x and the newly added/deleted item $\hat{x}$ and then updating non-membership witness such that it represents the Bezout coefficient's between x and the product of the accumulated elements. For a complete description and correctness proof see [112].

4.5.2 Batching and aggregation of accumulator witnesses

Aggregating membership witnesses Aggregating membership witnesses for many elements into a single membership witness for the set is straightforward using **ShamirTrick**. However, verification of this membership witness is linear in the number of group operations. Note that the individual membership witnesses can still be extracted from the aggregated witness as $w_x = w_{xy}^y$. Security, therefore, still holds for an accumulator construction with aggregated membership witnesses. The succinct proof of exponentiation ($\mathsf{NI\text{-}PoE}$) enables us to produce a single membership witness that can be verified in constant time. The verification **VerAggMemWit** simply checks the proof of exponentiation.

Aggregating existing membership witnesses for elements in several distinct accumulators (that use the same setup parameters) can be done as well. The algorithm **MemWitX** simply multiplies together the witnesses w_x for an element $x \in A_1$ and w_y for $y \in A_2$ to create an inclusion proof w_{xy} for x *and* y. The verification checks $w_{xy}^{x \cdot y} = A_1^y A_2^x$. If x and y are co-prime then we can directly recover w_x and w_y from the proof w_{xy}. In particular $w_x = \textbf{ShamirTrick}(A_1^y, A_1, w_{xy}^y A_2^{-1}, y, x)$ and $w_y = \textbf{ShamirTrick}(A_2^x, A_2, w_{xy}^x A_1^{-1}, x, y)$.

[1]The condition that $gcd(x, y) = 1$ is minor as we can simply use a different set of primes as the domains for each accumulator. Equivalently we can utilize different collision resistant hash functions with prime domain for each accumulator. The concrete security assumption would be that it is difficult to find two values a, b such that both hash functions map to the same prime. We utilize this aggregation technique in our IOP application (Section 4.7).

```
AggMemWit$(A, w_x, w_y, x, y)$ :
    1. $w_{x\cdot y} \leftarrow$ ShamirTrick$(A, w_x, w_y, x, y)$
    2. return $w_{x\cdot y}$, NI-PoE$(w_{x\cdot y}, x \cdot y, A)$
MemWitCreate*$(A, \{x_1, \ldots, x_n\})$ :
    1. $x^* = \prod_{i=1}^{n} x_i$
    2. $w_{x^*} \leftarrow$ MemWitCreate$(A, x^*)$
    3. return $w_{x^*}$, NI-PoE$(x, w_{x^*}, A)$
VerMem*$(A, \{x_1, \ldots, x_n\}, w = \{w_x, \pi\})$:
    1. return NI-PoE.verify$(\prod_{i=1}^{n} x_i, w, A, \pi)$

MemWitX$(A_1, A_2, w_x, w_y, x, y)$ :
    1. return $w_{xy} \leftarrow w_x \cdot w_y$
VerMemWitX$(A_1, A_2, w_{xy}, x, y)$ :
    1. if $gcd(x, y) \neq 1$
    2.     return $\perp$
    3. else
    4.     return $w_{xy}^{x\cdot y} \leftarrow A_1^y A_2^x$
```

Distributed accumulator updates In the decentralized/distributed setting, the accumulator is managed by a distributed network of participants who only store the accumulator state and a subset of the accumulator elements along with their membership witnesses. These participants broadcast their own updates and listen for updates from other participants, updating their local state and membership witnesses appropriately when needed.

We observe that the basic accumulator functions do not require a trapdoor or knowledge of the entire state, summarized in Figure 4.2. In particular, deleting an item requires knowledge of the item's current membership witness (the accumulator state after deletion is this witness). Moreover, operations can be performed in batches as follows:

The techniques are summarized as follows:

- **BatchAdd** An NI-PoE proof can be used to improve the amortized verification efficiency of a batch of updates that add elements $x_1, ..., x_m$ at once and update the accumulator to $A_{t+1} \leftarrow A_t^{x^*}$. A network participant would check that $x^* = \prod_i x_i$ and verify the proof rather than compute the m exponentiations.

- **BatchDel** Deleting elements in a batch uses the **AggMemWit** function to a compute the aggregate membership witness from the individual membership witnesses of each element. This is the new state of the accumulator. A NI-PoE proof improves the verification efficiency of this batch update.

- **CreateAllMemWit** It is possible for users to update membership and non-membership witnesses [112]. The updates do not require knowledge of the accumulated set S but do require that every accumulator update is processed. Since this is cumbersome some users may rely on service providers for maintaining the witness. The service provider may store the entire state or just the users witnesses. Creating all users witnesses naively requires $O(n^2)$ operations. Using the **RootFactor** algorithm this time can be reduced to $O(n \log(n))$ operations or amortized $O(\log(n))$ operations per witness.

- **CreateManyNonMemWit** Similarly to **CreateAllMemWit** it is possible to create m non-membership witness using $O(\max(n, m) + m \log(m))$ operations. This stands in contrast to the naive algorithm that would take $O(m \cdot n)$ operations. The algorithm is in Figure 4.5.2.

```
$\mathbf{Add}(A_t, x)$:
    1. return $A_t^x$
$\mathbf{BatchAdd}(A_t, \{x_1, \ldots, x_m\})$:
    1. $x^* \leftarrow \prod_{i=1}^m x_i$
    2. $A_{t+1} \leftarrow A_t^{x^*}$
    3. return $A_{t+1}, \mathsf{NI\text{-}PoE}(x^*, A_t, A_{t+1})$
$\mathbf{DelWMem}(A_t, w_x^t, x)$:
    1. if $\mathbf{VerMem}(A_t, w_x^t, x) = 1$
    2. return $w_x^t$
$\mathbf{BatchDel}(A_t, (x_1, w_{x_1}^t) \ldots, (x_m, w_{x_m}^t))$:
    1. $A_{t+1} \leftarrow w_{x_1}^t$
    2. $x^* \leftarrow x_1$
    3. for $i \leftarrow 2, i \leq m$
    4.     $A_{t+1} \leftarrow \mathbf{ShamirTrick}(A_{t+1}, w_{x_i}^t, x, x_i)$
    5.     $x^* \leftarrow x^* \cdot x_i$
    6. return $A_{t+1}, \mathsf{NI\text{-}PoE}(x^*, A_{t+1}, A_t)$
$\mathbf{CreateAllMemWit}(S)$ :
    1. return $\mathbf{RootFactor}(g, S)$
```

Figure 4.2: Distributed and stateless accumulator functions.

Batching non-membership witnesses A non-membership witness u_x for x in an accumulator with state A for a set S is $u_x = \{a, g^b\}$ such that $as^* + bx = 1$ for $s^* \leftarrow \prod_{s \in S} s$. The verification checks $A^a g^{bx} = g$. Since $gcd(s^*, x) = 1$ and $gcd(s^*, y) = 1$ if and only if $gcd(s^*, xy) = 1$, to batch non-membership witnesses we could simply construct a non-membership witness for $x \cdot y$. A prover computes $a', b' \leftarrow \mathbf{Bezout}(s^*, xy)$ and sets $u_{xy} \leftarrow a', g^{b'}$. This is still secure as a non-membership witness for both x and y because we can easily extract a non-membership witness for x as well as for y from the combined witness (a', B') by setting $u_x = (a', (B')^y)$ and $u_y = (a', (B')^x)$.

```
**CreateManyNonMemWit**$(A, S, \{x_1, \ldots, x_m\})$:
  1. $x^* = \prod_{i=1}^{m} x_i$
  2. $\{a, B\} =$ **NonMemWitCreate**$(A, S, x^*)$
  3. **return BreakUpNonMemWit**$(A, \{a, B\}, \{x_1, \ldots, x_m\})$
**BreakUpNonMemWit**$(A, \{a, B\}, \{x_1, \ldots, x_m\})$:
  1. **if** $m = 1$ **return** $\{a, B\}$
  2. $x_L = \prod_{i=1}^{m/2} x_i$
  3. $x_R = \prod_{i=\lfloor m/2 \rfloor+1}^{m} x_i$
  4. $B_L = B^{x_R} A^{\left\lfloor \frac{a}{x_L} \right\rfloor}, a_L = a \bmod x_L$
  5. $B_R = B^{x_L} A^{\left\lfloor \frac{a}{x_R} \right\rfloor}, a_R = a \bmod x_R$
  6. $u_L =$ **BreakUpNonMemWit**$(A, \{a_L, B_L\}, \{x_1, \ldots, x_{\lfloor m/2 \rfloor}\})$
  7. $u_R =$ **BreakUpNonMemWit**$(A, \{a_R, B_R\}, \{x_{\lfloor m/2 \rfloor+1}, \ldots, x_m\})$
  8. **return** $u_L || u_R$
```

Figure 4.3: Algorithm for creating multiple non membership witnesses

Unfortunately, $|a'| \approx |xy|$ so the size of this batched non-membership witness is linear in the number of elements included in the batch. A natural idea is to set $u_{xy} = (V, B) \leftarrow (A^{a'}, g^{b'}) \in \mathbb{G}^2$ instead of $(a', B) \in \mathbb{Z} \times \mathbb{G}$ as the former has constant size. The verification would check that $VB^{xy} = g$. This idea doesn't quite work as an adversary can simply set $V = gB^{-xy}$ without knowing a discrete logarithm between A and V. Our solution is to use the NI-PoKE2 protocol to ensure that V was created honestly. Intuitively, soundness is achieved because the knowledge extractor for the NI-PoKE2 can extract a' such that (a', B) is a standard non-membership witness for xy.

The new membership witness is $V, B, \pi \leftarrow$ NI-PoKE(A,v;b). The size of this witness is independent of the size of the statement. We can further improve the verification by adding a proof of exponentiation that the verification equation holds: NI-PoE$(x \cdot y, B, g \cdot V^{-1})$. Lastly, recall from Section 4.4 that the two independent NI-PoKE2 and NI-PoE proofs can be aggregated into a single group element.

We present the non-membership protocol bellow as **NonMemWitCreate***. The verification algorithm **VerNonMem*** simply verifies the NI-PoKE2 and NI-PoE.

```
**NonMemWitCreate***$(A, s^*, x^*)$ : // $A = g^{s^*}$, $s^* = \prod_{s \in S} s$, $x = \prod x_i, x_i \in \mathsf{Primes}(\lambda)$
    1. $a, b \leftarrow$ **Bezout**$(s^*, x^*)$
    2. $V \leftarrow A^a, B \leftarrow g^b$
    3. $\pi_V \leftarrow \mathsf{NI\text{-}PoKE2}(A, V; a)$ // $V = A^a$
    4. $\pi_g \leftarrow \mathsf{NI\text{-}PoE}(x^*, B, g \cdot V^{-1})$// $B^x = g \cdot V^{-1}$
    5. **return** $\{V, B, \pi_V, \pi_g\}$
**VerNonMem***$(A, u = \{V, B, \pi_V, \pi_g\}, \{x_1, \ldots, x_n\})$:
    1. **return** $\mathsf{NI\text{-}PoKE2.verify}(A, V, \pi_V) \wedge \mathsf{NI\text{-}PoE.verify}(\prod_{i=1}^{n} x_i, B, g \cdot V^{-1}, \pi_g)$
```

Soundness of batch non-membership witnesses Using the knowledge extractor for NI-PoKE2 and relying on the soundness of NI-PoE, and given an adversary who outputs a valid batch non-membership witness (V, B, π_V, π_g) for a set of odd prime elements $x_1, ..., x_k$ with respect to an accumulator state A, we can extract individual non-membership witnesses for each x_i. The knowledge extractor for NI-PoKE2 (Theorem 13) obtains a such that $V = A^a$ and the soundness of NI-PoE (Theorem 11) guarantees that $B^{x^*} \cdot V = g$ where $x^* = \prod_i x_i$. Given a and B we can compute a non-membership witness for $x_i | x^*$ as $B^{\frac{x^*}{x_i}}$,a because $(B^{\frac{x^*}{x_i}})^{x_i} A^a = B^{x^*} V = g$ Recall that we proved the existence of a knowledge extractor only for the interactive form of PoKE2 and soundness for the interactive form of PoE, relying on the generic group model. The existence of a knowledge extractor for NI-PoKE2 and soundness of NI-PoE are derived from the Fiat-Shamir heuristic.

Batch accumulator security We now formally define security for an accumulator with batch membership and non-membership witnesses. The definition naturally generalizes Definition 25. We omit a correctness definition as it follows directly from the definition of the batch witnesses. We assume that correctness holds perfectly.

Definition 26 (Batch Accumulator Security (Undeniability)).

$$\Pr\left[\begin{array}{l} \mathsf{pp}, A_0 \in \mathbb{G} \xleftarrow{\$} \mathsf{Setup}(\lambda) \\ (A, I, E, w_I, u_E) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}, A_0): \\ \textbf{\textit{VerMem*}}(A, I, w_I) \land \textbf{\textit{VerNonMem*}}(A, S, u_S) \land I \cap S \neq \emptyset \end{array}\right] = \mathsf{negl}(\lambda)$$

From the batch witnesses w_I and u_S we can extract individual accumulator witnesses for each element in I and S. Since the intersection of the two sets is not empty we have an element x and extracted witnesses w_x and u_x for that element. As in the proof of Theorem 14 this lets us compute and xth root of g which directly contradicts the strong RSA assumption. Our security proof will be in the generic group model as it implies the strong RSA assumption, the adaptive root assumption and can be used to formulate extraction for the PoKE2 protocol. Our security proof uses the interactive versions of PoKE2 and PoE protocols but extraction/soundness holds for their non-interactive variants as well.

Theorem 15. *The batch accumulator construction presented in Section 4.5.2 is secure (Definition 26) in the generic group model.*

Proof. We will prove security by showing that given an adversary that can break the accumulator security we can construct an efficient extractor that will break the strong RSA assumption (Definition 21). This, however, contradicts the generic group model in which strong RSA holds [72]. Given a strong RSA challenge $g \in \mathbb{G}$ we set A_0 the accumulator base value to g. Now assume there exists such an adversary $\mathcal{A}$ that on input $(\mathbb{G}, g)$ with non-negligible probability outputs (A, I, E, w_I, u_E) such that w_I and u_E are valid witnesses for the accumulator A and the inclusion proof elements I intersect with the exclusion proof elements E. Let $x \in I \cap S$ be in the intersection. The batch membership witness w_I is such that $w_I^{\prod_{x_i \in I} x_i} = A$ with overwhelming probability. This follows directly from the soundness of the accompanying PoE proof (Theorem 11). We can directly compute $w_x = w_I^{\prod x_i \in I, x_i \neq x}$, i.e. a membership witness for x.

The batch non-membership witness u_E consists of $B, V \in G$ as well as a PoKE2 and a PoE We now use the PoKE2 extractor to compute $a \in \mathbb{Z}, B \in \mathbb{G}$. Given that the extractor succeeds with overwhelming probability (Theorem 13) and the overwhelming soundness of PoE(Theorem 11), a, B satisfy $A^a B^{\prod_{x_i \in E} x_i} = g$. From this we can compute $B' = B^{\prod_{x_i \in E, x_i \neq x} x_i}$ such that $A^a B'^x = g$. As in the proof of Theorem 14 we can now compute $B' w_x^a$ as an xth root of g. This is because $B'^x (w_x^a)^x = B'^x A^a = g$. This, however, contradicts the strong RSA assumption. □

Aggregating non-membership witnesses We have shown how to create a constant sized batch non-membership witness for arbitrary many witnesses. However this process required knowledge of the accumulated set S. Is it possible to aggregate multiple independent non-membership witnesses into a single constant size witness? We show that we can aggregate unbatched witnesses and the apply the same batching technique to the aggregated witness. This is useful in the stateless blockchain setting where a node may want to aggregate non-membership witnesses created by independent actors. Additionally it will allow us to aggregate vector commitment openings for our novel construction presented in Section 4.6.

Given two non-membership witness $u_x = \{a_x, B_x\}$ and $u_y = \{a_y, B_y\}$ for two distinct elements x and y and accumulator A we want to create a witness for $x \cdot y$. As shown for batch non-membership witnesses a non-membership witness for $x \cdot y$ is equivalent to a witness for x and y if x and y are co-prime.

Concretely we will compute $a_{xy} \in \mathbb{Z}$ and $B_{xy} \in \mathbb{G}$ such that $A^{a_{xy}} B_{xy}^{x \cdot y} = g$. First we compute $\alpha, \beta \leftarrow \textbf{Bezout}(x, y)$. Then we set $B' \leftarrow B_x^\beta B_y^\alpha$ and set $a' \leftarrow \beta a_x y + \alpha a_y x$. Note that $B' \in \mathbb{G}$, $a' \in \mathbb{Z}$ already satisfy $A^{a'} B'^{xy} = (A^{a_x} B_x^x)^{\beta y} (A^{a_y} B_y^y)^{\alpha x} = g^{\beta y + \alpha x} = g$ but that $|a'|$ is not necessarily less than xy. To enforce this we simply reduce a' mod xy, setting $a_{xy} \leftarrow a'$ mod xy and $B_{xy} \leftarrow B' A^{\lfloor \frac{a'}{xy} \rfloor}$. The verification equation $A^{a_{xy}} B_{xy}^{xy} = A^{a'} B'^{xy} = g$ is still satisfied.

The full protocol is presented below:

```
**AggNonMemWit**$(A, x, y, u_x = (a_x \in [x], B_x \in \mathbb{G}), u_y = (a_y, B_y))$ :
    1. $\alpha, \beta \leftarrow \mathbf{Bezout}(x, y)$
    2. $B' \leftarrow B_x^{\beta} B_y^{\alpha}$
    3. $a' \leftarrow \beta a_x y + \alpha a_y x$
    4. $a_{xy} \leftarrow a' \bmod xy$
    5. $B_{xy} \leftarrow B' A^{\lfloor \frac{a'}{xy} \rfloor}$
    6. **return** $\{a_{xy}, B_{xy}\}$
```

As in Theorem 15, non-membership witnesses for x and y individually can be computed from an aggregated non-membership witness for x and y. Note that we can also use a PoKE proof and apply the non-membership batching technique presented above to make the proof constant size. The final witness can be verified using the **VerNonMem*** algorithm.

Unions and Multiset accumulators Our succinct proofs can be used to prove that an accumulator is the union of two other accumulators. This uses a succinct proof of a DDH tuple, another special case of a homomorphism preimage. Further details are given in Section 4.10.1. In the distributed accumulator setting, it is necessary to assume that no item is added twice to the accumulator. Otherwise, the distributed delete operation will fail. Alternatively, the construction can be viewed as a multi-set accumulator, where every element has a counter. Generating a valid membership witness for an element requires knowing the count of that element in the accumulator multi-set. Further details on this construction are given in Section 4.10.2.

4.6 Batchable Vector Commitments with Small Parameters

4.6.1 VC Definitions

We review briefly the formal definition of a vector commitment. We only consider static commitments that do not allow updates, but our scheme can naturally be modified to be dynamic.

Vector commitment syntax A VC is a tuple of four algorithms: $\mathsf{VC.Setup}$, $\mathsf{VC.Com}$, $\mathsf{VC.Open}$, $\mathsf{VC.Verify}$.

1. $\mathsf{VC.Setup}(\lambda, n, \mathcal{M}) \to \mathsf{pp}$ Given security parameter λ, length n of the vector, and message space of vector components $\mathcal{M}$, output public parameters pp, which are implicit inputs to all the following algorithms.

2. $\mathsf{VC.Com}(\vec{m}) \to \tau, com$ Given an input $\vec{m} = (m_1, ..., m_n)$ output a commitment com and advice τ.

3. $\mathsf{VC.Update}(com, m, i, \tau) \to \tau, com$ Given an input message m and position i output a commitment com and advice τ.

4. $\mathsf{VC.Open}(com, m, i, \tau) \to \pi$ On input $m \in \mathcal{M}$ and $i \in [1, n]$, the commitment com, and advice τ output an opening π that proves m is the ith committed element of com.

5. $\mathsf{VC.Verify}(com, m, i, \pi) \to 0/1$ On input commitment com, an index $i \in [n]$, and an opening proof π output 1 (accept) or 0 (reject).

If the vector commitment does not have an $\mathsf{VC.Update}$ functionality we call it a *static* vector commitment.

Definition 27 (Static Correctness)**.** *A static vector commitment scheme VC is correct if for all $\vec{m} \in \mathcal{M}^n$ and $i \in [1, n]$:*

$$\Pr\left[\mathsf{VC.Verify}(com, m_i, i, \pi) = 1 : \begin{array}{l} pp \leftarrow \mathsf{VC.Setup}(\lambda, n, \mathcal{M}) \\ \tau, com \leftarrow \mathsf{VC.Com}(\vec{m}) \\ \pi \leftarrow VC.Open(com, m_i, i, \tau) \end{array}\right] = 1$$

The correctness definition for dynamic vector commitments also incorporates updates. Concretely whenever $\mathsf{VC.Update}$ is invoked the underlying committed vector $\vec{m}$ is updated correctly.

Binding commitments The main security property of vector commitments (of interest in the present work) is position binding. The security game augments the standard binding commitment game

Definition 28 (Binding)**.** *A vector commitment scheme VC is position binding if for all $O(\mathsf{poly}(\lambda))$-time adversaries $\mathcal{A}$ the probability over $\mathsf{pp} \leftarrow \mathsf{VC.Setup}(\lambda, n, \mathcal{M})$ and $(com, i, m, m', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{pp})$ the probability that $\mathsf{VC.Verify}(com, m, i, \pi) = \mathsf{VC.Verify}(com, m', i, \pi') = 1$ and $m \neq m'$ is negligible in λ.*

4.6.2 VC construction

We first present a VC construction for bit vectors, i.e. using the message space $\mathcal{M} = \{0, 1\}$. We then explain how this can be easily adapted for a message space of arbitrary bit length.

Our VC construction associates a unique prime[2] integer p_i with each ith index of the bitvector $\vec{m}$ and uses an accumulator to commit to the set of all primes corresponding to indices where $m_i = 1$. The opening of the ith index to $m_i = 1$ is an inclusion proof of p_i and the opening to $m_i = 0$ is an exclusion proof of p_i. By using our accumulator from Section 4.5, the opening of each index is constant-size. Moreover, the opening of several indices can be batched into a constant-size proof by aggregating all the membership witnesses for primes on the indices opened to 1 and batching all the non-membership witnesses for primes on the indices opened to 0.

The VC for vectors on a message space of arbitrary bit length is exactly the same, interpreting the input vector as a bit vector. Opening a λ-bit component is then just a special case of batch opening several indices of a VC to a bit vector. The full details are in Figure 4.4.

Both the accumulator's CRS as well as $\mathsf{PrimeGen}$ can be represented in constant space independent of n. This means that the public parameters for the vector commitment are also constant-size and independent of n, unlike all previous vector commitments with $O(1)$ size openings [60, 113, 110]. The batch opening of several (mixed value) indices consists of

[2]Examples include $\mathsf{H}_{\mathsf{prime}}$ (described earlier), or alternatively the function that maps i to the next prime after $f(i) = 2(i+2)\cdot\log_2(i+2)^2$, which maps the integers $[0, N)$ to smaller primes than $\mathsf{H}_{\mathsf{prime}}$ (in expectation).

```
VC.Setup($\lambda$):
  • $A \leftarrow Accumulator.\mathsf{Setup}(\lambda)$
  • return $pp \leftarrow (A, n)$

VC.Com($\vec{m}, pp$) :
  • $\mathcal{P} \leftarrow \{p_i | i \in [1,n] \wedge m_i = 1\}$
  • $A.\mathbf{BatchAdd}(\mathcal{P})$
  • return $A$

VC.Update($b, b' \in \{0,1\}, i \in [1,n]$):
  • if $b = b'$ return $A$
  • elseif $b = 1$
  •     return $A.\mathbf{Add}(p_i)$
  • else
  •     return $A.\mathbf{Del}(p_i)$

VC.Open($b \in \{0,1\}, i \in [1,n]$) :
  • if $b = 1$
  •     return $A.\mathbf{MemWitCreate}(p_i)$
  • else
  •     return $A.\mathbf{NonMemWitCreate}(p_i)$

VC.Verify($A, b \in \{0,1\}, i, \pi$) :
  • if $b = 1$ :
  •     return $A.\mathbf{VerMem}(\pi, p_i)$
  • else :
  •     return $A.\mathbf{VerNonMem}(\pi, p_i)$

VC.BatchOpen($\vec{b} \in \{0,1\}^m, \vec{i} \in [1,n]^m$) :
  • Ones $\leftarrow \{j \in [1,m] : b_j = 1\}$
  • Zeros $\leftarrow \{j \in [1,m] : b_j = 0\}$
  • $p^+ \leftarrow \prod_{j \in \mathsf{Ones}} p_{\vec{i}[j]}$; $p^- \leftarrow \prod_{j \in \mathsf{Zeros}} p_{\vec{i}[j]}$
  • $\pi_I \leftarrow A.\mathbf{MemWitCreate*}(p^+)$
  • $\pi_E \leftarrow A.\mathbf{NonMemWitCreate*}(p^-)$
  • return $\{\pi_I, \pi_E\}$
VC.BatchVerify($A, \vec{b}, \vec{i}, \pi_I, \pi_E$) :
  • Ones $\leftarrow \{j \in [1,m] : b_j = 1\}$
  • Zeros $\leftarrow \{j \in [1,m] : b_j = 0\}$
  • $p^+ \leftarrow \prod_{j \in \mathsf{Ones}} p_{\vec{i}[j]}$; $p^- \leftarrow \prod_{j \in \mathsf{Zeros}} p_{\vec{i}[j]}$
  • return $A.\mathbf{VerMem}(p^+, \pi_I) \wedge$
    $A.\mathbf{VerNonMem*}(p^-, \pi_E)$
VC.BatchOpen*($\vec{b} \in \{0,1\}^m, \vec{i} \in [1,n]^m$) :
  • Ones $\leftarrow \{j \in [1,m] : b_j = 1\}$
  • Zeros $\leftarrow \{j \in [1,m] : b_j = 0\}$
  • $p^+ \leftarrow \prod_{j \in \mathsf{Ones}} p_{\vec{i}[j]}$; $p^- \leftarrow \prod_{j \in \mathsf{Zeros}} p_{\vec{i}[j]}$
  • $\pi_I \leftarrow A.\mathbf{MemWitCreate*}(p^+)$
  • $\pi_E \leftarrow A.\mathbf{NonMemWitCreate}(p^-)$
  • return $\{\pi_I, \pi_E\}$
VC.BatchVerify*($A, \vec{b}, \vec{i}, \pi_I, \pi_E$) :
  • Ones $\leftarrow \{j \in [1,m] : b_j = 1\}$
  • Zeros $\leftarrow \{j \in [1,m] : b_j = 0\}$
  • $p^+ \leftarrow \prod_{j \in \mathsf{Ones}} p_{\vec{i}[j]}$; $p^- \leftarrow \prod_{j \in \mathsf{Zeros}} p_{\vec{i}[j]}$
  • return $A.\mathbf{VerMem}(p^+, \pi_I) \wedge$
    $A.\mathbf{VerNonMem}(p^-, \pi_E)$
```

Figure 4.4: Vector commitment scheme from accumulator with batchable membership and non-membership witnesses.

2 elements in $\mathbb{G}$ for the aggregate membership-witness and an additional 5 elements in $\mathbb{G}$ for the batch non-membership witness, plus one λ-bit integer.

Aggregating Openings Just as for our accumulator construction we can aggregate vector commitment openings. The aggregation does not require knowledge of the vector contents and the running time of the aggregation is independent of the length of the vector. The opening of a bit in the vector commitment consists of an accumulator inclusion proof and an exclusion proof, both of which we can aggregate as shown in Section 4.5.2.

This aggregation protocol works for outputs of VC.Open, but unfortunately it does not extend to outputs of VC.BatchOpen. The latter contain PoKE proofs created by **VerNonMem***, which would somehow need to be aggregated as well along with their inputs. When opening only a small number of bit indices, say in a 256-bit component of the vector, VC.BatchOpen* could be used instead so that these openings can be later aggregated. While the output size of VC.BatchOpen* grows linearly in m, the number of batched indices, it still contains only three group elements and an integer whose size is proportional to the product of at most m λ-bit primes. These are the unique primes associated with indices of the vector and heuristically can be chosen to be much smaller than λ bits, i.e. closer to $\log n$ bits. After the aggregation step is completed, the aggregate non-membership witness can be further compressed with a PoKE proof.

This aggregation protocol is important for the account-based stateless blockchain application, described in more detail in Section 4.7. In this application, there is a distributed network of participants who who each hold openings for only a partial number of the vector components (e.g. every user knows the value corresponding to their own account data). A batch of transactions will typically contain many openings (of small values) produced by many different participants in the network. In this case, it makes sense for the participants to each produce an opening of the form VC.BatchOpen* so that after the individual participants post all the openings they can be aggregated into a single constant size value that is persisted in the transaction log.

Optimization The number of group elements can be reduced by utilizing a PoKCR for all of the PoE and PoKE roots involved in the membership/non-membership witness generation. It is important that all PoE and PoKE protocols use different challenges. These challenges are then guaranteed to be co-prime. This reduces the number of opening proof elements to $5 \in \mathbb{G}$ and 1 λ-bit integer.

4.6.3 Comparison

Table 4.6.3 compares the performance of our new VC scheme, the Catalano-Fiore (CF)[60] RSA-based VC scheme, and Merkle trees. The table assumes the VC input is a length n

vector of k bit elements with security parameter λ. The performance for the CF scheme include batch openings which were introduced by Lai and Malatova[110]. We note that the **MultiExp** algorithm from Section 4.4.3 also applies to the CF scheme. In particular it can improve the Setup and Open time. The comparison reflects these improvements.

Metric	This Work	Catalono-Fiore [60, 110]	Merkle Tree
		Setup	
Setup	O(1)	$O(n \cdot \log(n) \cdot \lambda)$ $\mathbb{G}$	O(1)
$\lvert\mathsf{pp}\rvert$	O(1)	O(n) $\mathbb{G}$	O(1)
$\mathsf{Com}(\mathbf{m}) \to c$	$O(n \cdot \log(n) \cdot k)$ $\mathbb{G}$	$O(n \cdot k)$ $\mathbb{G}$	$O(n)$ H
$\lvert\mathbf{c}\rvert$	1 $\mathbb{G}$	1 $\mathbb{G}$	1 $\lvert\mathsf{H}\rvert$
		Proofs	
$\mathsf{Open}(m_i, i) \to \pi$	$O(k \cdot n \log n)$ $\mathbb{G}$	$O(n \cdot (k + \lambda))$ $\mathbb{G}$.	$O(\log n)$ H
$\mathsf{Verify}(m_i, i, \pi)$	$O(\lambda)$ $\mathbb{G}$+ $k \cdot \log n$ $\mathbb{F}$	$O(k + \lambda)$ $\mathbb{G}$	$O(\log n)$ H
$\lvert\pi\rvert$	$O(1)$ $\lvert\mathbb{G}\rvert$	1 $\lvert\mathbb{G}\rvert$	$O(\log n)$ $\lvert\mathsf{H}\rvert$
$\mathsf{Open}(\mathbf{m}, \vec{i}) \to \pi_{\vec{i}}$	$O(k \cdot n \log n)$ $\mathbb{G}$	$O(n \log n \cdot (k + \lambda))$ $\mathbb{G}$	$O(n \log n)$ H
$\mathsf{Verify}(\mathbf{m}, \vec{i}, \pi_{\mathbf{i}})$	$O(\lambda)$ $\mathbb{G}$ + $O(k \cdot n \log n)$ $\mathbb{F}$	$O(nk)$ $\mathbb{G}$	$O(n \log n)$ H
$\lvert\pi_{\mathbf{i}}\rvert$	4 $\lvert\mathbb{G}\rvert$+ λ	1 $\lvert\mathbb{G}\rvert$	$O(n \log n)$ $\lvert\mathsf{H}\rvert$
Aggregatable	Yes	No	No

Table 4.1: Comparison between Merkle trees, Catalano-Fiore RSA VCs and the new VCs presented in this work. The input $\vec{m}$ is a vector of n. $\mathbb{G}$ refers to a group operation in $\mathbb{G}$, $\mathbb{F}$ to a multiplication in a field of size roughly 2^λ, and H to a hash operation. Group operations are generally far more expensive than hashes which are more expensive than multiplication in $\mathbb{F}$. $\lvert\mathbb{G}\rvert$ is the size of a hidden order group element and $\lvert\mathsf{H}\rvert$ is the size of a hash output.

4.6.4 Key-Value Map Commitment

Our vector-commitment can be used to build a commitment to a key-value map. A key-value map can be built from a sparse vector. The key-space is represented by positions in the vector and the associated value is the data at the keys position. The vector length is exponential in the key length and most positions are zero (null). Our VC commitment naturally supports sparse vectors because the complexity of the commitment is proportional to the number of bit indices that are set to 1, and otherwise independent of the vector length.

Optimization with honest updates In order to commit to arbitrary length values we can hash the value and then commit to the resulting hash. Unfortunately this still requires setting λ bits in the vector commitment which corresponds to adding λ, λ-bit primes to the underlying accumulator. This can make updating the commitment computationally quite expensive. In some settings we can do better than this. Note that the VC and the accumulator definitions (Definition 25) assume that the adversary outputs the commitment. This requirement is too strong for settings where every update follows the rules of the system, i.e. is performed by the challenger. In this case we can implement a key-value map commitment by storing in the VC which keys exist and storing in the accumulator a key, value tuple. If the key already exists then an update will update the entry in the accumulator. Otherwise it will add an entry to the accumulator and set the corresponding VC bit to 1. The construction requires only 1 bit to be stored per key in the VC and 1 entry in the accumulator. The construction also is not secure if the adversary can output an accumulator value as it could contain multiple entries for the same key. We omit a formal security definition and proof but note that security follows directly from the binding guarantees of the underlying accumulator and vector commitment constructions. The VC ensures that each key appears at most ones in the accumulator and the accumulator ensures the integrity of the committed data.

4.7 Application to IOP-based SNARKs

Merkle tree paths contribute significant overhead to both the proof size of a compiled IOP proof and its verification time. Vector commitments with smaller openings than Merkle trees, or batchable openings (i.e. subvector commitments), can help reduce this overhead [110]. Using our new VCs, the opening proof for each round of the compiled IOP is just 4 group elements in $\mathbb{G}$ and a λ-bit integer (plus one additional element for the VC commitment itself). Instantiating $\mathbb{G}$ with a class group of quadratic imaginary order and tuning security to 128-bits requires elements of size approximately 2048-bits [98]. Thus, the VC openings contribute 8320 bits to the proof size per IOP round. When applied to the “CS-proof” SNARK considered by Lai and Malavolta, which is based on a theoretical PCP that checks 3 bits per query and has 80 queries, the proof size is $5 \cdot 2048 + 128 + 3 \cdot 80 = 10608$ bits, or 1.3

KB. This is the shortest (theoretical) setup-free SNARK with sublinear public parameters to date.

Our VCs also achieve concrete improvements to practical IOPs. Targeting 100-bit security in the VC component and otherwise apples-to-apples comparisons with benchmarks for Aurora [27] and STARKS [22], we can conservatively use 2048-bit class group elements. With these parameters, our VCs reduce the size of the Aurora proofs on a 2^{20} size circuit from 222 KB to less than 100 KB, a 54% reduction, and the size of STARK proofs for a circuit of 2^{52} gates from 600 KB to approximately 222 KB, a 63% reduction. This rough estimate is based on the Merkle path length 42 and round number 21 extrapolated from the most recent STARK benchmarks for this size circuit [22].

Replacing Merkle trees with our VCs does not significantly impact the verification cost, and in some cases it may even improve verification time. Recall that verifying a batch VC proof costs approximately one λ-bit integer multiplication and a primality check per bit. Furthermore, using the optimization described in Section 4.8 eliminates the primality checks for the verifier (at a slight cost to the prover). Computing a SHA256 hash function (whether SHA256 or AES with Davies-Meyer) is comparable to the cost of a λ-bit integer multiplication. Thus, as a loose estimate, replacing each Merkle path per query with a single λ-bit multiplication would achieve a factor $\log n = 36$ reduction. In STARKS, Merkle paths are constructed over 256-bit blocks of the proof rather than bits, thus the comparison is 36 hashes vs 256 modular multiplications. The Merkle path validation accounts for 80% of the verification time.

While using our vector commitment has many benefits for IOPs, there are several sever downsides. Our vector commitment is not quantum secure as a quantum computer can find the order of the group and break the Strong-RSA assumption. Merkle trees are more plausibly quantum secure. Additionally, the prover for an IOP instantiated with our vector commitment would be significantly slower than one with a Merkle tree.

4.8 Hashing To Primes

Our constructions use a hash-function with prime domains in several places: Elements in the accumulator are mapped to primes, using a collision resistant hash function with prime domain. The vector commitment associates a unique prime with each index. All of the proofs presented in Section 4.4 use a random prime as a challenge. When the proofs are made non-interactive, using the Fiat-Shamir heuristic the challenge is generated by hashing the previous transcript.

In Section 4.5.1 we present a simple algorithm for a collision-resistant hash function $\mathsf{H}_{\mathsf{prime}}$ with prime-domain built from a collision resistant hash function H with domain $\mathbb{Z}_{2^\lambda}$. The hash function iteratively hashes a message and a counter, increasing the counter until the output is prime. If we model H as a random function with then the expected running time of $\mathsf{H}_{\mathsf{prime}}$ is $O(\lambda)$. This is because there are $O(\frac{n}{\log(n)})$ primes below n.

The problem of hashing to primes has been studied in several contexts: Cramer and Shoup [69] provide a way to generate primes with efficiently checkable certificates. Fouque and Tibouchi[83] showed how to quickly generate random primes. Seeding the random generation with a collision resistant hash function can be used to generate an efficient hash function with prime domain. Despite these improvements, the hash function actually introduces a significant overhead for verification and in this section we present several techniques how the hashing can be further sped up.

PoE,PoKE proofs We first investigate the PoE,PoKE family of protocols. In the non-interactive variant the challenge ℓ is generated by hashing the previous transcript to a prime. The protocol can be modified by having the prover provide a short nonce such that $\ell \leftarrow H(transcript||nonce)$ with $\ell \in \mathsf{Primes}(\lambda)$. In expectation the nonce is just $\log(\lambda)$ bits and with overwhelming probability it is less than $2\log(\lambda)$ bits. This modification allows the adversary to produce different challenges for the same transcript. However it does not increase an adversary's advantage. The prover can always alter the input to generate new challenges. By changing the nonce the prover can grind a polynomial number of challenges but the soundness error in all of our protocols is negligible. The change improves the verification as the verifier only needs to do a single primality check instead of λ. The change is particularly interesting if proof verification is done in a circuit model of computation, where variable time operations are difficult and costly to handle. Circuit computations have become increasingly popular for general purpose zero-knowledge proofs[89, 51, 27]. Using the adapted protocol verification becomes a constant time operation which uses only a single primality check.

Accumulator A similar improvement can be applied to accumulators. The users can provide a nonce such that $element||nonce$ is accumulated instead of just the element. This of course allows an adversary to accumulate the same element twice. In some applications this is acceptable. In other applications such as stateless blockchains it is guaranteed that no element is accumulated twice(see Section 4.7). One way to guarantee uniqueness is to commit to the current state of the accumulator for every added element. In an inclusion proof, the prover would provide the nonce as part of the proof. The verifier now only does a single primality check to ensure that $\mathsf{H}(element||nonce)$ is indeed prime. This stands in contrast to $O(\lambda)$ primality checks if $\mathsf{H}_{\mathsf{prime}}$ is used. The nonce construction prohibits efficient exclusion proofs but these are not required in some applications, such as the blockchain application.

Vector Commitments The vector commitment construction uses one prime per index to indicate whether the vector is 1 at that index or 0. The security definition for a vector commitment states that a secure vector commitment cannot be opened to two different openings at the same index. In our construction this would involve giving both an inclusion as well as an exclusion proof for a prime in an accumulator, which is impossible if the accumulator itself is secure. Using a prime for each index again requires using a collision resistant hash function with prime domain which uses $O(\lambda)$ primality checks or an injective function which runs in time $O(\log(n)^2)$, where n is the length of the vector. What if instead of accumulating a prime for each index we accumulate a random λ bit number at each index? The random number could simply be the hash of the index. Is this construction still secure? First consider the case where each index's number has a unique prime factor. This adapted construction is trivially still secure. What, however, if x_k, associated with index k, is the product of x_i and x_j. Then accumulating x_i and x_j lets an adversary also give an inclusion proof for x_k. Surprisingly, this does still not break security. While it is possible to give an inclusion proof for x_k, i.e. open the vector at index k to 1 it is suddenly impossible to give an exclusion proof for x_k, i.e. open the vector at index k to 0. The scenario only breaks the correctness property of the scheme, in that it is impossible to commit to a vector that is 1 at i and j but 0 at k. In a setting, where the vector commitment is used as a static commitment to a vector, correctness only needs to hold for the particular vector that is being committed to. In the IOP application, described in Section 4.7, the prover commits to a long proof using a vector commitment. If these correctness failures only happen for few vectors, it may still be possible to use the scheme. This is especially true because in the IOP application the proof and also the proof elements can be modified by hashing the proof elements along with a nonce. A prover would modify the nonces until he finds a proof, i.e. a vector that he can commit to. To analyze the number of correctness failures we can compute the probability that a k-bit element divides the product of n k-bit random elements. Fortunately, this question has been analyzed by Coron and Naccache[67] with respect to the Gennaro-Halevi-Rabin Signature Scheme[90]. They find that for 50 Million integers and 256-bit numbers the probability that even just a single correctness failure occurs is 1%. Furthermore we find experimentally that for 2^{20} integers and 80-bit numbers only about $8,000$ integers do not have a unique prime factor. Thus, any vector that is 1 at these $8,000$ positions can be committed to using just 80-bit integers. Our results suggest that using random integer indices instead of prime indices can be useful, if a) perfect completeness is not required b) primality checks are a major cost to the verifier.

4.9 PoE/PoKE Generalizations and Zero Knowledge

4.9.1 A succinct proof of homomorphism preimage

We observe that the protocol PoE can be generalized to a relation for any homomorphism $\phi : \mathbb{Z}^n \to \mathbb{G}$ for which the adaptive root assumption holds in $\mathbb{G}$. Specifically, Protocol PoHP below is a protocol for the relation:

$$\mathcal{R}_{\phi,\mathsf{PoHP}} = \big\{\big((w \in \mathbb{G},\ \mathbf{x} \in \mathbb{Z}^n);\ \bot\big)\ :\ w = \phi(\mathbf{x}) \in \mathbb{G}\big\}.$$

This generalization will be useful in our applications.

Protocol PoHP (Proof of homomorphism preimage) for $\mathcal{R}_{\phi,\mathsf{PoHP}}$

Params: $\mathbb{G} \xleftarrow{\$} GGen(\lambda),\ \phi : \mathbb{Z}^n \to \mathbb{G}$; Inputs: $\mathbf{x} \in \mathbb{Z}^n,\ w \in \mathbb{G}$; Claim: $\phi(\mathbf{x}) = w$

1. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
2. For $i = 1, \ldots, n$: Prover finds integers q_i and $r_i \in [\ell]$ s.t. $x_i = q_i\ell + r_i$.
   Let $\mathbf{q} \leftarrow (q_1, ..., q_n) \in \mathbb{Z}^n$ and $\mathbf{r} \leftarrow (r_1, ..., r_n) \in [\ell]^n$.
   Prover sends $Q \leftarrow \phi(\mathbf{q}) \in \mathbb{G}$ to Verifier.
3. Verifier computes $r_i = (x_i \bmod \ell) \in [\ell]$ for all $i = 1, \ldots, n$, sets $\mathbf{r} = (r_1, \ldots, r_n)$, and accepts if $Q^\ell \phi(\mathbf{r}) = w$ holds in $\mathbb{G}$.

Theorem 16 (Soundness PoHP)**.** *Protocol PoHP is an argument system for Relation $\mathcal{R}_{\phi,\mathsf{PoHP}}$ with negligible soundness error, assuming the adaptive root assumption holds for GGen.*

Proof. Suppose that $\phi(\mathbf{x}) \neq w$, but the adversary succeeds in making the verifier accept with non-negligible probability. Let $\mathbf{q}$ and $\mathbf{r}$ be as defined in step (2) of the protocol and let Q be the prover's message to the verifier. Then $[Q/\phi(\mathbf{q})]^\ell = [w/\phi(\mathbf{r})]/[\phi(\mathbf{x})/\phi(\mathbf{r})] = w/\phi(\mathbf{x}) \neq 1$. We thus obtain an algorithm to break the adaptive root assumption for the instance $\hat{w} := w/\phi(\mathbf{x})$ by interacting with the adversary, giving it the adaptive root challenge ℓ, and outputting $\hat{u} := Q/\phi(\mathbf{q}) \in \mathbb{G}$, where Q is the value output by the adversary. $\square$

4.9.2 A succinct proof of knowledge of a homomorphism preimage

The PoKE argument of knowledge can be extended to an argument of knowledge for the pre-image of a homomorphism $\phi : \mathbb{Z}^n \to \mathbb{G}$.

$$\mathcal{R}_\phi = \big\{ \big(w \in \mathbb{G};\ \mathbf{x} \in \mathbb{Z}^n \big) \ :\ w = \phi(\mathbf{x}) \in \mathbb{G} \big\}.$$

For a general homomorphism ϕ we run into the same extraction challenge that we encountered in extending Protocol PoKE* to work for general bases. The solution for Protocol PoKE was to additionally send g^x where g is either a base in the CRS or chosen randomly by the verifier and execute a parallel PoKE for $g \mapsto g^x$. We can apply exactly the same technique here on each component x_i of the witness, i.e. send g^{x_i} to the verifier and execute a parallel PoKE that $g \mapsto g^{x_i}$. This allows the extractor to obtain the witness x, and the soundness of the protocol then follows from the soundness of Protocol PoHP. However, as an optimization to reduce the communication we can instead use the group representation homomorphism $Rep : \mathbb{Z}^n \to \mathbb{G}$ defined as

$$Rep(\mathbf{x}) = \prod_{i=1}^{n} g_i^{x_i}$$

for base elements g_i defined in the CRS. The prover sends $Rep(\mathbf{x})$ in its first message, which is a single group element independent of n.

Protocol PoKHP (Proof of knowledge of homomorphism preimage)

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$, $(g_1, ..., g_n) \in \mathbb{G}^n$, $\phi : \mathbb{Z}^n \to \mathbb{G}$; Inputs: $w \in \mathbb{G}$; Witness: $\mathbf{x} \in \mathbb{Z}$; Claim: $\phi(\mathbf{x}) = w$

1. Prover sends $z = Rep(\mathbf{x}) = \prod_i g_i^{x_i} \in \mathbb{G}$ to the verifier.
2. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.
3. For each x_i, Prover computes q_i, r_i s.t. $x_i = q_i\ell + r_i$, sets $\vec{q} \leftarrow (q_1, ..., q_n) \in \mathbb{Z}^n$ and $\vec{r} \leftarrow (r_1, ..., r_n) \in [\ell]^n$. Prover sends $Q_\phi \leftarrow \phi(\vec{q}) \in \mathbb{G}$, $Q_{Rep} \leftarrow Rep(\vec{q}) \in \mathbb{G}$, and $\vec{r}$ to Verifier.

4. Verifier accepts if $\vec{r} \in [\ell]^n$, $Q_\phi^\ell \phi(\mathbf{r}) = w$, and $Q_{Rep}^\ell Rep(\mathbf{r}) = z$.

In order to analyze the security of this protocol, it is helpful to first consider a special case of Protocol PoKHP protocol for the homomorphism $Rep : \mathbb{Z}^n \to \mathbb{G}$, which is a generalization of Protocol PoKE*. In this case the prover of course does not need to separately send $Rep(\mathbf{x})$ in the first message. The protocol is as follows:

Protocol PoKRep (Proof of knowledge of representation)

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda), (g_1, ..., g_n) \in \mathbb{G}^n$; Inputs: $w \in \mathbb{G}$; Witness: $\vec{x} \in \mathbb{Z}$; Claim: $Rep(\vec{x}) = \prod_{i=1}^n g_i^{x_i} = w$

1. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.
2. For each x_i, Prover finds q_i, r_i s.t. $x_i = q_i\ell + r_i$, sets $\vec{q} \leftarrow (q_1, ..., q_n) \in \mathbb{Z}^n$ and $\vec{r} \leftarrow (r_1, ..., r_n) \in [\ell]^n$. Prover sends $Q \leftarrow Rep(\vec{q}) = \prod_i g_i^{q_i} \in \mathbb{G}$ and $\vec{r}$ to Verifier.
3. Verifier accepts if $\vec{r} \in [\ell]^n$, $Q^\ell Rep(\vec{r}) = w$.

The following theorems prove security of the two protocols above.

Theorem 17 (PoKRep Argument of Knowledge)**.** Protocol PoKRep *is an argument of knowledge for relation* $\mathcal{R}_{\mathsf{Rep}}$ *in the generic group model.*

Proof. See Section 4.11. □

Theorem 18 (PoKHP Argument of Knowledge)**.** *Protocol PoKHP is an argument of knowledge for the relation* $\mathcal{R}_\phi$ *in the generic group model.*

Proof. See Section 4.11. □

4.9.3 A succinct proof of integer exponent mod n

There are several applications of accumulators that require proving complex statements about integer values committed in an accumulator (e.g. [25, 124]). Practical succinct argument systems (SNARGs/SNARKs/STARKs) operate on statements defined as an arithmetic circuit, and the prover efficiency scales with the multiplication complexity of the statement. Since RSA accumulators are an algebraic accumulator, in constrast to Merkle trees, one would hope that the arithmetic complexity of statements involving RSA accumulator elements would be much lower than those involving Merkle tree elements. Unfortunately, this is not the case because RSA accumulator operations are in $\mathbb{Z}_N$ for composite N with unknown factorization, whereas arithmetic circuits for SNARGs are always defined over finite fields $\mathbb{Z}_p$. Finding ways to combine RSA accumulators with SNARKs more efficiently is an interesting research direction.

We present a variant of PoKE^* for a group of unknown order $\mathbb{G}$, which is an argument of knowledge that the integer discrete log x of an element $y \in \mathbb{G}$ is equivalent to $\hat{x}$ modulo a public odd prime integer n. Concretely, the new protocol $\mathsf{PoKEMon}$ is for the following relation:

$$\mathcal{R}_{\mathsf{PoKEMon}} = \big\{ \big(w \in \mathbb{G}, \hat{x} \in [n]; x \in \mathbb{Z} \big) \ : \ w = g^x \in \mathbb{G}, x \bmod n = \hat{x} \big\}.$$

As with PoKE^*, the base element g and the unknown order group $\mathbb{G}$ are fixed public parameters. $\mathsf{PoKEMon}$ modifies PoKE^* by setting the challenge to be $\ell \cdot n$ where $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.

Protocol $\mathsf{PoKEMon}$ (Proof of equality mod n) for Relation $\mathcal{R}_{\mathsf{PoKEMon}}$

Params: $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$, $g \in \mathbb{G}$; Inputs: Odd prime $n, w \in \mathbb{G}, \hat{x} \in [n]$; Witness: $x \in \mathbb{Z}$; Claim: $g^x = w$ and $x \bmod n = \hat{x}$

1. Verifier sends $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$.
2. Prover computes the quotient $q \in \mathbb{Z}$ and residue $r \in [\ell \cdot n]$ such that $x = q(\ell \cdot n) + r$. Prover sends the pair $(Q \leftarrow g^q, \ r)$ to the Verifier.
3. Verifier accepts if $r \in [\ell \cdot n]$ and $Q^{\ell \cdot n} g^r = w$ holds in $\mathbb{G}$ and $r \bmod n = \hat{x}$.

The same technique can be applied to PoKE, where the base element can be freely chosen by the prover.

We can prove security by directly reducing it to the security of the PoKE^* protocol and additionally the strong RSA assumption, which assumes it is hard to compute an ℓth root of a random group element g for odd prime ℓ.

Theorem 19 (PoKEMon Argument of Knowledge)**.** *Protocol PoKEMon is an argument of knowledge for the relation $\mathcal{R}_{\mathsf{PoKEMon}}$ if* ProtocolPoKE* *is an argument of knowledge for the relation $\mathcal{R}_{\mathsf{PoKE}^*}$ and the strong RSA assumption holds for GGen.*

Proof. We use the extractor Ext^* of the PoKE* protocol to build an extractor Ext for PoKEMon, which succeeds with overwhelming probability in extracting x such that $g^x = w$ and $x = \hat{x} \bmod n$ from any PoKEMon adversary that has a non-negligible success rate.

Ext runs a copy of Ext^* and simulates both the PoKE* challenges and a PoKE* adversary's response. When Ext receives the challenge ℓ and the PoKEMon adversary's response (Q, r), it computes $q' = \lceil r/\ell \rceil$ and $r' = r \bmod \ell$ so that $r = q'\ell + r'$ and sets $Q' \leftarrow Q^n g^{q'}$. It forwards (ℓ, Q', r') to Ext^*. If the PoKEMon adversary's response is valid then $Q^{\ell n} g^r = w$, implying that $Q'^{\ell} g^{r'} = w$. Thus, Ext simulates for Ext^* a transcript of the PoKE* protocol for a PoKE* adversary that succeeds with the same rate as the PoKEMon adversary. By hypothesis, Ext^* succeeds with overwhelming probability to output x such that $g^x = w$.

Consider any iteration in which Ext had received an accepting PoKEMon transcript (ℓ, Q, r). We claim that $x = r \bmod \ell \cdot n$ with overwhelming probability, by the strong RSA assumption.

Suppose that $x - r \neq 0 \bmod \ell \cdot n$ then given that ℓ and n are prime we have that either $\gcd(\ell, x - r) = 1$ or $\gcd(n, x - r) = 1$. Without loss of generality we assume the latter. Let $Q' = Q^{\ell}$ then $Q'^n = g^{x-r}$ and $\gcd(n, x - r) = 1$. Now using Shamir's trick we can compute an nth root of g. Let $a, b = \mathbf{Bezout}(n, x - r)$ such that $an + b(x - r) = 1$. Then $u = Q'^b g^a$ is an nth root of g as $u^n = Q'^{bn} g^{an} = g^{b\cdot(x-r)} g^{an} = g$. This shows that (u, n) breaks the strong RSA assumption for $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$.

This contradicts the hypothesis and we, therefore, have that $r = x \bmod \ell \cdot n$ which implies that $r = x \bmod n$ for an overwhelming number of accepting transcripts. And since in any accepting transcript $\hat{x} = r \bmod n$ we have that $x = \hat{x} \bmod n$ with overwhelming probability.

□

4.9.4 A succinct zero-knowledge proof of discrete-log

The PoKE protocol for succinctly proving knowledge of an exponent can further be made zero-knowledge using a method similar to the classic Schnorr Σ-protocol for hidden order groups. The Schnorr protocol for hidden order groups has the same structure as the standard Schnorr protocol for proving knowledge of a discrete logarithm x such that $u^x = w$ in a group of known order. Here, the prover first samples a blinding factor $k \in [-B, B]$ and sends $A = u^k$, obtains a challenge c, and returns $s = k + cx$. The verifier checks that $u^z = aw^c$. In hidden order groups, k must be sampled from a range of integers $[-B, B]$ such that $|\mathbb{G}|/B$ is negligible.

The classical Schnorr protocol for hidden order groups is an honest verifier statistical zero-knowledge (HVSZK) protocol and has soundness error of only 1/2 against a classical adversary [16]. Only for a small subclass of homomorphisms better soundness can be proven [17]. Unfortunately, [16] proved that the soundness limitation is fundamental and cannot be improved against a classical adversary, and therefore requires many rounds of repetition. However, we are able to show that we can prove much tighter soundness if the adversary is restricted to operating in a generic group.

Definition 29 (Zero Knowledge)**.** *We say an argument system* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *for* $\mathcal{R}$ *has **statistical zero-knowledge** if there exists a poly-time simulator* Sim *such that for* $(x, w) \in \mathcal{R}$ *the following two distribution are statistically indistinguishable:*

$$D_1 = \left\{ \langle \mathcal{P}(pp, x, w), \mathcal{V}(pp, x) \rangle, \ pp \xleftarrow{\$} \mathsf{Setup}(\lambda) \right\}$$

$$D_2 = \left\{ \mathsf{Sim}(pp, x, \mathcal{V}(pp, x)), \ pp \xleftarrow{\$} \mathsf{Setup}(\lambda) \right\}$$

The protocol. Our ZK protocol applies Protocol PoKE to the last step of the Schnorr protocol, which greatly improves the communication efficiency of the classical protocol when the witness is large. In fact, we can interleave the first step of Protocol PoKE where the verifier sends a random prime ℓ with the second step of the Schnorr protocol where the verifier sends a challenge c. This works for the case when u is a base specified in the CRS, i.e. it is the output of a query to the generic group oracle $\mathcal{O}_1$, however a subtlety arises when u is selected by the prover. In fact, we cannot even prove that the Schnorr protocol itself is secure (with negligible soundness error) when u is selected by the prover. The method we used for PoKE on general bases involved sending g^x for g specified in the CRS. This would immediately break ZK since the simulator cannot simulate g^x without knowing the witness x. Instead, in the first step the prover will send a Pedersen commitment $g^x h^\rho$ where ρ is sampled randomly in some interval and h is another base specified in the CRS.

We will first present a ZK proof of knowledge of a representation in terms of bases specified in the CRS and show that there is an extractor that can extract the witness. We then use this as a building block for constructing a ZK protocol for the relation $\mathcal{R}_{\mathsf{PoKE}}$.

Protocol ZKPoKRep for Relation $\mathcal{R}_\phi$ where $\phi := Rep$

Params: $(g_1, \dots g_n) \in \mathbb{G}$, $\mathbb{G} \xleftarrow{\$} GGen(\lambda), B > 2^{2\lambda}|\mathbb{G}|$; Inputs: $w \in \mathbb{G}$;

Witness: $\vec{x} = (x_1, \dots, x_n) \in \mathbb{Z}^n$; Claim: $Rep(\vec{x}) = \prod_{i=1}^n g_i^{x_i} = w$

1. Prover chooses random $k_1, \dots, k_n \xleftarrow{\$} [-B, B]$, sends $A = \prod_{i=1}^n g_i^{k_i}$ to Verifier.
2. Verifier sends $c \xleftarrow{\$} [0, 2^\lambda], \ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
3. Prover computes $s_i = k_i + c \cdot x_i \forall i \in [1, n]$ and then derives quotients $\vec{q} \in \mathbb{Z}^n$ and residues $\vec{r} \in [\ell]^n$ such that $q_i \cdot \ell + r_i = s_i$ for all $1 \le i \le n$. Prover sends $Q = \prod_{i=1}^n g_i^{q_i}$ and $\vec{r}$ to the Verifier.
4. Verifier accepts if $r_i \in [\ell]$ for all $1 \le i \le n$ and that $Q^\ell \prod_{i=1}^n g_i^{r_i} = Aw^c$.

Theorem 20 (Protocol ZKPoKRep)**.** *Protocol ZKPoKRep is an honest-verifier statistically zero-knowledge argument of knowledge for relation $\mathcal{R}_{Rep}$ in the generic group model.*

Proof. See Section 4.11. □

Finally, we use the protocol above to obtain a ZK protocol for the relation $\mathcal{R}_{\mathsf{PoKE}}$. The protocol applies (in parallel) the Σ-protocol for PoKRep to a Pedersen commitment $g^x h^\rho$ for g and h specified in the CRS. In order to achieve statistical zero-knowledge we require that g and h generate the same subgroup of $\mathbb{G}$. This requirement can be lifted when computation zero-knowledge suffices. The extractor for this protocol will invoke the PoKRep extractor to open the commitment. The protocol works as follows:

<u>Protocol ZKPoKE for $\mathcal{R}_{\mathsf{PoKE}}$</u>

Params: $(g,h) \in \mathbb{G}$ s.t. $\langle g \rangle = \langle h \rangle$, $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$; Inputs: $u, w \in \mathbb{G}$, $B > 2^{2\lambda}|\mathbb{G}|$;

Witness: $x \in \mathbb{Z}$; Claim: $u^x = w$

Let $Com(x;r) := g^x h^r$.

1. Prover chooses random $k, \rho_x, \rho_k \xleftarrow{\$} [-B, B]$ and sends (z, A_g, A_u) to the verifier where $z = Com(x;\rho_x)$, $A_g = Com(k;\rho_k)$, $A_u = u^k$.
2. Verifier sends $c \xleftarrow{\$} [0, 2^\lambda], \ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
3. Prover computes $s_x = k + c \cdot x$ and $s_\rho = \rho_k + c \cdot \rho_x$ and then derives quotients $q_1, q_2 \in \mathbb{Z}$ and residues $r_x, r_\rho \in [\ell]$ such that $q_x \cdot \ell + r_x = s_x$ and $q_\rho \cdot \ell + r_\rho = s_\rho$. Prover sends $Q_g = Com(q_x; q_\rho)$, $Q_u = u^{q_x}$ and r_x, r_ρ to the Verifier.
4. Verifier accepts if $r_x, r_\rho \in [\ell]$ and

$$Q_g^\ell \cdot Com(r_x; r_\rho) = A_g z^c \qquad \text{and} \qquad Q_u^\ell \cdot u^{r_x} = A_u w^c.$$

Theorem 21 (Protocol ZKPoKE). *Protocol ZKPoKE is an honest verifier statistically zero-knowledge argument of knowledge for relation $\mathcal{R}_{\mathsf{PoKE}}$ in the generic group model.*

Proof. See Section 4.11. □

4.10 More Accumulator techniques

4.10.1 Accumulator unions

Yet another application of our succinct proofs to accumulators is the ability to prove that an accumulator is the union of two other accumulators. Given three accumulators $A_1 = g_1^{\prod_{s \in S_1} s}$, $A_2 = g_2^{\prod_{s \in S_2} s}$ and $A_3 = A_2^{\prod_{s \in S_1} s}$ a prover can use the NI-PoDDH protocol to convince a verifier that (A_1, A_2, A_3) forms a valid DDH tuple. If S_1 and S_2 are guaranteed to be disjoint, then A_3 will be an accumulator of $S_1 \cup S_2$. If they are not disjoint, then resulting accumulator will be an accumulator for a multi-set as described in the next paragraph. The

NI-PoDDH is independent of the size of S_1 and S_2 in both the proof size and the verification time. This union proof can be used to batch exclusion proofs over multiple accumulators. The prover verifiably combines the accumulators and then creates a single aggregate non-membership proof in the union of the accumulators. This is sound but only works if the domains of the accumulators are separate.

4.10.2 Multiset accumulator

A dynamic multiset accumulator is an accumulator where items can be added and deleted more than once, and every element has a count. In other words, it is a commitment to a mapping from items to non-negative integer counters. It has the following properties:

- Each element in the domain is implicitly in the mapping with a counter of 0.
- **Add** increments the counter of the added element by 1
- **Del** decrements the counter of the added element by 1
- A membership witness for an element x and a counter k proves that the counter of x is at least k
- A membership witness for x^k and a non-membership witness for $A^{x^{-k}}$ proves that the counter for x is exactly k. Note that $A^{x^{-k}}$ is exactly the membership witness for x^k.

To build the multi-set accumulator we again employ a hash function mapping an arbitrary domain to an exponentially large set of primes. The **Add** and **Del** algorithms are as described in Section 4.5.2. The membership witness change in that they now also contain a counter of how many times a certain element has been added. That is if an element x is k times in the accumulator the membership witness is the x^kth root of the accumulator as well as k. **VerMem**, **MemWitCreate**, **MemWitUpAdd**, **MemWitUpDel** are changed accordingly. The completeness definition also needs to be updated to reflect the new multi-set functionalities.

4.11 Security Proofs

4.11.1 Preliminary lemmas

In the following lemmas, which all concern the generic group model, we restrict ourselves to adversaries that do not receive any group elements as input. This is sufficient to prove our theorems. For our proof protocols we require that the adversary itself outputs the instance after receiving a description of the group. We require this in order to prevent that the instance itself encodes a trapdoor, such as the order of the group.

Lemma 22 (Element representation [153])**.** *Using the notation of Section 4.3.2, let $\mathbb{G}$ be a generic group and $\mathcal{A}$ a generic algorithm making q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. Let $\{g_1, \ldots, g_m\}$ be the outputs of $\mathcal{O}_1$. There is an efficient algorithm Ext that given as input the transcript of $\mathcal{A}$'s interaction with the generic group oracles, produces for every element $u \in \mathbb{G}$ that $\mathcal{A}$ outputs, a tuple $(\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}^m$ such that $u = \prod_{i=1}^m g_i^{\alpha_i}$ and $\alpha_i \leq 2^{q+2}$.*

Lemma 23 (Computing multiple of orders of random elements)**.** *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$. Let $\mathcal{A}$ be a generic algorithm making q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. The probability that $\mathcal{A}$ succeeds in computing $0 \neq k \in \mathbb{N}$ such that for a g which is a response to an $\mathcal{O}_1$ query $g^k = 1$ is at most $\frac{(q_1+q_2)^3}{M}$, where $1/M$ is negligible whenever $|B - A| = \exp(\lambda)$. When $\mathcal{A}$ succeeds we say that event* Root *happened.*

We denote $\mathrm{ord}_{\mathbb{G}}(g)$ as the order of $g \in \mathbb{G}$. By definition $g^k = 1 \wedge 0 \neq k \in \mathbb{Z} \leftrightarrow k \bmod \mathrm{ord}_{\mathbb{G}}(g) = 0$.

Proof. This lemma is a direct corollary of Theorem 1 from [72]. That theorem shows that an adversary that interacts with the two generic group oracles cannot solve the strong RSA problem with probability greater than $(q_1 + q_2)^3/M$, where M is as in the statement of the lemma. Recall that a strong RSA adversary takes as input a random $g \in \mathbb{G}$ and outputs (u, x) where $u^x = g$ and x is an odd prime. Let $\mathcal{A}$ be an adversary from the statement of the lemma, that is, $\mathcal{A}$ outputs $0 < k \in \mathbb{Z}$ where $k \equiv 0 \bmod |\mathbb{G}|$ with some probability ϵ. This $\mathcal{A}$

immediately gives a strong RSA adversary that also succeeds with probability ϵ: run $\mathcal{A}$ to get k and g such that $g^k = 1 \in \mathbb{G}$. Then find an odd prime x that does not divide k, and output (u, x) where $u = g^{(x^{-1} \bmod k)}$. Clearly $u^x = g$ which is a solution to the given strong RSA challenge. It follows by Theorem 1 from [72] that $\epsilon \leq (q_1 + q_2)^3/M$, as required. □

Lemma 24 (Discrete Logarithm)**.** *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$, where $1/A$ and $1/|B - A|$ are negligible in λ. Let $\mathcal{A}$ be a generic algorithm and let $\{g_1, \ldots, g_m\}$ be the outputs of $\mathcal{O}_1$. Then if $\mathcal{A}$ runs in polynomial time, it succeeds with at most negligible probability in outputting $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m \in \mathbb{Z}$ such that $\prod_{i=1}^m g_i^{\alpha_i} = \prod_{i=1}^m g_i^{\beta_i}$ and $\alpha_i \neq \beta_i$ for some i. We call this event* DLOG.

Proof sketch. We follow the structure of Shoup's argument [153]. By Lemma 22 every group element $u \in \mathbb{G}$ that the adversary obtains in response to an $\mathcal{O}_2$ query can be written as $u = \prod_{i=1}^m g_i^{\alpha_i}$ for some known $\alpha_i \in \mathbb{Z}$. Let $g = \prod_{i=1}^m g_i^{\alpha_i}$ and $h = \prod_{i=1}^m g_i^{\beta_i}$ be two such group elements. If there is some i for which $\alpha_i \not\equiv \beta_i \pmod{\mathrm{ord}_{\mathbb{G}}(g_i)}$ then the probability that $g = h$ is at most negligible, as shown in [72]. Hence, if $g = h$ then with overwhelming probability we have that $\alpha_i \equiv \beta_i \pmod{\mathrm{ord}_{\mathbb{G}}(g_i)}$ for all i. From this it follows by Lemma 23 that $\alpha_i = \beta_i \in \mathbb{Z}$ with overwhelming probability, since otherwise one obtains a multiple of $|\mathbb{G}|$. Since $\mathcal{A}$ constructs at most polynomially many group elements, there are at most polynomially many pairs of such elements. Therefore, a union bound over all pairs shows that the probability that event DLOG happens is at most negligible, as required. □

Lemma 25 (Dlog extraction)**.** *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$ and g an output of a query to $\mathcal{O}_1$. Let $\mathcal{A}$ be a generic algorithm that outputs $w \in \mathbb{G}$ and then runs the interactive protocol* PoKE^* *with g in the CRS. Let (ℓ_1, Q_1, r_1) and (ℓ_2, Q_2, r_2) two accepting transcripts for* PoKE^* *generated one after the other. If $1/A$ and $1/|B - A|$ are negligible in λ, then with overwhelming probability there exist integers α and β such that $\alpha \cdot l_1 + r_1 = \beta \cdot l_2 + r_2$ and $g^{\alpha \cdot l_1 + r_1} = w$. Further if $\mathcal{A}$ makes q queries to $\mathcal{O}_2$ then $|\alpha|, |\beta|$ are bounded by 2^q.*

Proof. W.l.o.g. let $g_1 = g$ be encoded in the PoKE^* CRS. The PoKE^* verification equations give us $w = Q_1^{\ell_1} g^{r_1} = Q_2^{\ell_2} g^{r_2}$. We can write $Q_1 = \prod_{i=1}^m g_i^{\alpha_i}$ and $Q_2 = \prod_{i=1}^m g_i^{\beta_i}$. This implies that $Q_1^{\ell_1} g^{r_1} = g^{\alpha_1 \cdot \ell_1 + r_1} \prod_{i=2}^m g_i^{\alpha_i \cdot \ell_1} = g^{\beta_1 \cdot \ell_2 + r_2} \prod_{i=2}^m g_i^{\beta_i \cdot \ell_2}$. By Lemma 24, $\alpha_i \ell_1 = \beta_i \ell_2 \in \mathbb{Z}$ for all $i \neq 1$ with overwhelming probability (i.e. unless event DLOG occurs), and therefore $\ell_2 | \alpha_i \ell_1$. The primes ℓ_1 and ℓ_2 are co-prime unless $\ell_1 = \ell_2$, which happens with probability $\frac{\ln(2)\lambda}{2^\lambda}$. Thus, with overwhelming probability $\ell_2 | \alpha_i$. However, $\alpha_i \leq 2^{q_2}$ and α_i is chosen before ℓ_2 is sampled, hence the probability that $\ell_2 | \alpha_i$ for $\alpha_i \neq 0$ is at most $\frac{q_2 \lambda \ln(2)}{2^\lambda}$. We conclude that with overwhelming probability $\alpha_i = \beta_i = 0$ for all $i \neq 1$. It follows that except with probability $\Pr[\mathsf{DLOG}] + \frac{2 q_2 \lambda \ln(2)}{2^\lambda}$, we can express $w = g^{\alpha_1 \ell_1 + r_1} = g^{\beta_1 \ell_2 + r_2}$ for integers $\alpha_1, r_1, \beta_1, r_2$ such that $\alpha_1 \ell_1 + r_1 = \beta_1 \ell_2 + r_2$. □

In what follows we will use the following notation already introduced in Section 4.4: for generators $g_1, \ldots, g_n \in \mathbb{G}$ we let $Rep : \mathbb{Z}^n \to \mathbb{G}$ be the homomorphism

$$Rep(\vec{x}) = \prod_{i=1}^n g_i^{x_i}.$$

Lemma 26 (Representation extraction). *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$ and let $g_1, \ldots, g_n \in \mathbb{G}$ be responses to queries to oracle $\mathcal{O}_1$. Let $\mathcal{A}$ be a generic algorithm that outputs $w \in \mathbb{G}$ and then runs the interactive protocol PoKRep on input w with $g_1, ..., g_n$ in the CRS. Let $(\ell_1, Q_1, \mathbf{r}_1)$ and $(\ell_2, Q_2, \mathbf{r}_2)$ be two accepting transcripts for PoKRep. If $1/A$ and $1/|B - A|$ are negligible in λ, then with overwhelming probability there exist integer vectors $\vec{\alpha}, \vec{\beta} \in \mathbb{Z}^n$ such that $\vec{\alpha} l_1 + \vec{r}_1 = \vec{\beta} l_2 + \vec{r}_2$ and $Rep(\vec{\alpha} l_1 + \vec{r}_1) = w$. Further if $\mathcal{A}$ makes q queries to $\mathcal{O}_2$ then each component α_j and β_j of $\vec{\alpha}$ and $\vec{\beta}$ are bounded by 2^q.*

Proof. The proof is a direct generalization of the argument in Lemma 25 above. From the verification equations of the protocol we have $Q_1^{\ell_1} Rep(\vec{r}_1) = Q_2^{\ell_2} Rep(\vec{r}_2) = w$. With overwhelming probability, the generic group adversary knows $\alpha_1, ..., \alpha_m$ and $\beta_1, .., \beta_m$ for $m > n$ such that it can write $Q_1 = \prod_{i=1}^m g_i^{\alpha_i}$ and $Q_2 = \prod_{i=1}^m g_i^{\beta_i}$. From the verification equation and Lemma 24, with overwhelming probability $\alpha_i \ell_1 + \vec{r}_1[i] = \beta_i \ell_2 + \vec{r}_2[i]$ for each $i \leq n$ and $\alpha_i \ell_1 = \beta_i \ell_2$ for each $i > n$. As explained in the proof of Lemma 25, this implies that with overwhelming probability $\alpha_i = \beta_i = 0$ for each $i > n$, in which case $w = \prod_{i=1}^{n} g_i^{\alpha_i \ell_1 + r_1[i]} = \prod_{i=1}^{n} g_i^{\beta_i \ell_2 + r_2[i]}$. Setting $\vec{\alpha} := (\alpha_1, ..., \alpha_n)$ and $\vec{\beta} := (\beta_1, ..., \beta_n)$, we conclude that with overwhelming probability $w = \mathit{Rep}(\vec{\alpha}\ell_1 + \vec{r}_1) = \mathit{Rep}(\vec{\beta}\ell_2 + \vec{r}_2)$ and $\vec{\alpha}\ell_1 + \vec{r}_1 = \vec{\alpha}\ell_2 + \vec{r}_2$. Finally, if $\mathcal{A}$ has made at most q queries to $\mathcal{O}_2$ then $\alpha_i < 2^q$ and $\beta_i < 2^q$ for each i.

□

The next two corollaries show that the adaptive root problem and the known order element problem are intractable in a generic group.

Corollary 4 (Adaptive root hardness)**.** *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$ such that $1/|A|$ and $1/|B - A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in breaking the adaptive root assumption on $\mathbb{G}$ with at most negligible probability in λ.*

Proof. Recall that in the adaptive root game the adversary outputs $w \in \mathbb{G}$, the challenger then responds with a prime $\ell \in [2, 2^\lambda]$, and the adversary succeeds if it outputs u such that $u^\ell = w$. According to Lemma 22 we can write $u = \prod_{i=1}^{m} g_i^{\alpha_i}$ and $w = \prod_{i=1}^{m} g_i^{\beta_i}$, where $g_1, \ldots, g_m$ are the responses to oracle $\mathcal{O}_1$ queries. By Lemma 24 we know that $\alpha_i \ell = \beta_i \bmod |\mathbb{G}|$ for all $i = 1, \ldots, m$ with overwhelming probability, namely $1 - \Pr[\mathsf{DLOG}]$. Therefore, $\alpha_i \ell = \beta_i + k \cdot |\mathbb{G}|$ for some $k \in \mathbb{Z}$. By Lemma 23, an efficient adversary can compute a multiple of the order of the group with at most negligible probability $\Pr[\mathsf{Root}]$. It follows that $k = 0$ and $\alpha_i \ell = \beta_i \in \mathbb{Z}$ with probability greater than $1 - \Pr[\mathsf{DLOG}] - \Pr[\mathsf{Root}]$, since otherwise $\alpha_i \ell - \beta_i$ is a multiple of $\mathbb{G}$. Now, because $\alpha_i \ell = \beta_i$ we know that ℓ must divide β_i. However, β_i is chosen before ℓ and if $\mathcal{A}$ makes q_2 generic group queries then $\beta_i \leq 2^{q_2}$. The probability that ℓ divides β_i, for $\beta_i \neq 0$, is bounded by the probability that a random prime in $\mathsf{Primes}(\lambda)$ divides a number less than 2^{q_2}. Any such number has less than q_2 distinct prime factors and there are more than $2^\lambda/\lambda$ primes in $\mathsf{Primes}(\lambda)$. Therefore, the probability that ℓ divides $\beta_i \neq 0$ is at most $\frac{q_2 \cdot \lambda}{2^\lambda}$. Overall, we obtain that a generic adversary can break the adaptive root assumption with probability at most $\frac{(q_1+q_2)^2}{A} + 2 \cdot \frac{(q_1+q_2)^3}{M} + \frac{q_2 \cdot \lambda}{2^\lambda}$, which is negligible if A and $B - A$ are exponential in λ and q_1, q_2 are bounded by some polynomial in λ. $\square$

Corollary 5 (Non-trivial order hardness). *Let $\mathbb{G}$ be a generic group where $|\mathbb{G}|$ is a uniformly chosen integer in $[A, B]$ such that $1/|A|$ and $1/|B - A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in finding an element $h \neq 1 \in \mathbb{G}$ and an integer d such that $h^d = 1$ with at most negligible probability in λ.*

Proof. We can construct an adaptive root adversary that first uses $\mathcal{A}$ to obtain h and d, and then computes the ℓth root of h by computing $c = \ell^{-1} \bmod d$ and $h^c = h^{1/\ell}$. Since the adaptive root assumption holds true in the generic group model (Corollary 4), we can conclude that $\mathcal{A}$ succeeds with negligible probability. $\square$

Fact 3 (Chinese Remainder Theorem (CRT)). *Let $\ell_1, \ldots, \ell_n$ be coprime integers and let $r_1, \ldots, r_n \in \mathbb{Z}$, then there exists a unique $0 \leq x < \prod_{i=1}^{n} \ell_i$ such that $x = r_i \bmod \ell_i$ and there is an efficient algorithm for computing x.*

4.11.2 Proofs of the main theorems

Proof of Theorem 17.

> PoKRep is an argument of knowledge for the relation $\mathcal{R}_\phi$ where $\phi := Rep$, in the generic group model.

Fix $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$ and $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}$. Let $\mathcal{A}_0, \mathcal{A}_1$ be poly-time generic adversaries where $(w, \mathsf{st}) \overset{\$}{\leftarrow} A_0(\mathbf{g})$ and $\mathcal{A}_1(\mathsf{st})$ runs PoKRep with a verifier $V(\mathbf{g}, w)$. We need to show that for all $\mathcal{A}_1$ there exists a poly-time Ext such that for all $\mathcal{A}_0$ the following holds: if $\mathcal{A}_1$ convinces $V(\mathbf{g}, w)$ to accept with probability $\epsilon \geq 1/\text{poly}(\lambda)$, then Ext outputs a vector $\vec{x} \in \mathbb{Z}^n$ such that $Rep(\vec{x}) = w$ with overwhelming probability.

> **Subclaim** In PoKRep, for any polynomial number of accepting transcripts $\{(\ell_i, Q_i, \vec{r}_i)\}_{i=1}^{\mathsf{poly}(\lambda)}$ obtained by rewinding $\mathcal{A}_1$ on the same input (w, st), with

overwhelming probability there exists $\vec{x} \in \mathbb{Z}^n$ such that $\vec{x} = \vec{r}_i \bmod \ell_i$ for each i and $Rep(\vec{x}) = w$. Furthermore, $x_j \leq 2^q$ for each jth component x_j of $\vec{x}$, where q is the total number of queries that $\mathcal{A}$ makes to the group oracle.

The subclaim follows from Lemma 26. With overwhelming probability there exists $\vec{\alpha}, \vec{\beta}$, and $\vec{x}$ in $\mathbb{Z}^n$ such that $\vec{x} = \vec{\alpha}\ell_1 + \vec{r}_1 = \vec{\beta}\ell_2 + \vec{r}_2$ and $Rep(\vec{x}) = w$, and each component of $\vec{x}$ is bounded by 2^q. Consider any third transcript, w.l.o.g. $(\ell_3, Q_3, \vec{r}_3)$. Invoking the lemma again, there exists $\vec{\alpha}'$, $\vec{\beta}'$, and $\vec{x}'$ such that $\vec{x}' = \vec{\alpha}'\ell_2 + \vec{r}_2 = \vec{\beta}'\ell_3 + \vec{r}_3$. Thus, with overwhelming probability, $\vec{x}' - \vec{x} = (\vec{\alpha}' - \vec{\beta})\ell_2$. However, since ℓ_2 is sampled randomly from an exponentially large set of primes independently from $\vec{r}_1, \vec{r}_3, \ell_1$, and ℓ_3 (which fix the value of $\vec{x}' - \vec{x}$) there is a negligible probability that $\vec{x}' - \vec{x} \equiv 0 \pmod{\ell_2}$, unless $\vec{x}' = \vec{x}$. By a simple union bound over the $\mathsf{poly}(\lambda)$ number of transcripts, there exists a single $\vec{x}$ such that $\vec{x} = \vec{r}_i \bmod \ell_i$ for all i.

To complete the proof of Theorem 17 we describe the extractor Ext:

1. run $\mathcal{A}_0$ to get output (w, st)
2. let $R \leftarrow \{\}$
3. run PoKRep with $\mathcal{A}_1$ on input (w, st), sampling fresh randomness for the verifier
4. if the transcript $(\ell, Q, \vec{r})$ is accepting set $R \leftarrow R \cup \{(\vec{r}, \ell)\}$, and otherwise return to Step 3
5. use the CRT algorithm to compute $\vec{x}$ such that $\vec{x} = \vec{r}_i \bmod \ell_i$ for each $(\vec{r}_i, \ell_i) \in R$
6. if $Rep(\vec{x}) = w$ output $\vec{x}$ and stop
7. return to Step 3

It remains to argue that Ext succeeds with overwhelming probability in a $\mathsf{poly}(\lambda)$ number of rounds. Suppose that after some polynomial number of rounds the extractor has obtained M accepting transcripts $\{\ell_i, Q_i, \vec{r}_i\}$ for independent values of $\ell_i \in \mathsf{Primes}(\lambda)$. By the subclaim above, with overwhelming probability there exists $\vec{x} \in \mathbb{Z}^n$ such that $\vec{x} = \vec{r}_i \bmod \ell_i$ and $Rep(\vec{x}) = w$ and $x_j < 2^q$ for each component of $\vec{x}$. Hence, the CRT algorithm used in Step 5 will recover the required vector $\vec{x}$ once $|R| > q$.

Since a single round of interaction with $\mathcal{A}_1$ results in an accepting transcript with probability $\epsilon \geq 1/\text{poly}(\lambda)$, in expectation the extractor obtains $|R| > q$ accepting transcripts for independent primes ℓ_i after $q \cdot \text{poly}(\lambda)$ rounds. Hence, Ext outputs a vector $\vec{x}$ such that $Rep(\vec{x}) = w$ in expected polynomial time, as required. $\square$

Proof of Theorem 13.

> PoKE and $\mathsf{PoKE2}$ are arguments of knowledge for relation $\mathcal{R}_{\mathsf{PoKE}}$ in the generic group model.

Fix $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$ and $g \in \mathbb{G}$. Let $\mathcal{A}_0, \mathcal{A}_1$ be poly-time adversaries where $(u, w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(g)$ and $\mathcal{A}_1$ runs PoKE or $\mathsf{PoKE2}$ with the verifier $\mathcal{V}(g, u, w)$. We need to show that for all $\mathcal{A}_1$ there exists a poly-time Ext such that for all $\mathcal{A}_0$ the following holds: if $\mathcal{V}(g, u, w)$ outputs 1 with non-negligible probability on interaction with $\mathcal{A}_1(g, u, w, \mathsf{st})$ then Ext outputs an integer x such that $u^x = w$ in $\mathbb{G}$ with overwhelming probability.

Proof for Protocol PoKE. PoKE includes an execution of PoKE^* on $g \in \mathbb{G}$ and input z (the first message sent by the prover to the verifier), and the prover succeeds in PoKE only if it succeeds in this subprotocol for PoKE^*. Since PoKE^* is a special case of PoKRep, by Theorem 17 there exists Ext^* for $\mathcal{A}_1$ that outputs $x^* \in \mathbb{Z}$ such that $g^{(x^*)} = z$. Furthermore, as already shown in the analysis of Theorem 17, once Ext^* has obtained x^* it can continue to replay the protocol, sampling a fresh prime $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$, and in each fresh round that produces an accepting transcript it obtains from the Prover a triple (Q, Q', r) such that $r = x^* \bmod \ell$ with overwhelming probability. This is due to the fact that the adversary outputs Q' such that $Q'^{\ell} g^r = z = g^{x^*}$, and the generic group adversary can write $Q' = g^q \prod_{i>1} g_i^{q_i}$ (Lemma 22) such that $q\ell + r = x^*$ with overwhelming probability (Lemma 24).

The extractor Ext will simply run Ext^* to obtain x^*. Now we will show that either $u^{x^*} = w$, i.e. Ext^* extracted a valid witness, or otherwise the adaptive root assumption would be broken, which is impossible in the generic group model (Corollary 4). To see this, we construct an adaptive root adversary $\mathcal{A}_{AR}$ that first runs Ext^* with $\mathcal{A}_0, \mathcal{A}_1$ to obtain x^* and provides $h = w/u^{x^*} \in \mathbb{G}$ to the challenger. When provided with $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ from the challenger, $\mathcal{A}_{AR}$ rewinds $\mathcal{A}_1$, passes ℓ to $\mathcal{A}_1$, and with overwhelming probability obtains Q, r such that $x^* = r \bmod \ell$ and $Q^\ell u^r = w$. Finally, $\mathcal{A}_{AR}$ outputs $v = \frac{Q}{u^{\lfloor \frac{x^*}{\ell} \rfloor}}$, which is an ℓth root of h:

$$v^\ell = (\frac{Q}{u^{\lfloor \frac{x^*}{\ell} \rfloor}})^\ell = (\frac{Q}{u^{\lfloor \frac{x^*}{\ell} \rfloor}})^\ell \frac{u^r}{u^r} = \frac{w}{u^{x^*}} = h$$

If $w \neq u^{x^*}$ so that $h \neq 1$, then $\mathcal{A}_{AR}$ succeeds in the adaptive root game. In conclusion, the value x^* output by Ext satisfies $w = u^{x^*}$ with overwhelming probability.

Proof for protocol PoKE2 Showing that $\mathsf{PoKE2}$ requires a fresh argument (similar to the analysis in Theorem 17) since the protocol no longer directly contains PoKE^* as a subprotocol. Ext first obtains u, w from $\mathcal{A}_0$ and runs the first two steps of $\mathsf{PoKE2}$ with $\mathcal{A}_1$ playing the role of the verifier, sampling $g \xleftarrow{\$} \mathbb{G}$ and receiving $z \in \mathbb{G}$ from $\mathcal{A}_1$. Ext is a simple modification of the extractor for PoKE:

1. Set $R \leftarrow \{\}$ and sample $\alpha \xleftarrow{\$} [0, 2^\lambda]$.
2. Sample $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ and send α, ℓ to $\mathcal{A}_1$.
3. Obtain output Q, r from $\mathcal{A}_0$. If $Q^\ell u^r g^{\alpha r} = w z^\alpha$ (i.e. the transcript is accepting) then update $R \leftarrow R \cup \{(r, \ell)\}$. Otherwise return to step 2.
4. Use CRT to compute $x = r_i \mod \ell_i$ for each $(r_i, \ell_i) \in R$. If $u^x = w$ then output x, otherwise return to step 2.

Note that the extractor samples a fresh prime challenge ℓ each time it rewinds the adversary but keeps the challenge α fixed each time. Since these are independently sampled in the real protocol, keeping α fixed while sampling a fresh prime does not change the output distribution of the adversary. This subtle point of the rewinding strategy is important.

There is a negligible probability that the random g sampled by the extractor was contained in the group oracle queries from $\mathcal{A}_0$ to $\mathcal{O}_1$. Thus, by Lemma 22, $\mathcal{A}_0$ knows representations $w = \prod_i g_i^{\omega_i}$ and $u = \prod_i g_i^{\mu_i}$ such that $g_i \neq g$ for all i. $\mathcal{A}_0$ also knows a representation

$z = g^{\zeta}\prod_i g_i^{\zeta_i}$ and for each Q obtained $\mathcal{A}_0$ knows a representation $Q = g^q \prod_i g_i^{q_i}$, which it can pass in st to $\mathcal{A}_1$. If $Q^{\ell}u^r g^{\alpha r} = wz^{\alpha}$, then $\mathcal{A}_1$ obtains an equation $g^{q\ell+\alpha r}\prod_i g_i^{q_i\ell+\mu_i r} = g^{\zeta\alpha}\prod_i g_i^{\zeta_i\alpha+\omega_i}$.

By Lemma 24, with overwhelming probability $q\ell + \alpha r = \zeta\alpha$, which implies $\alpha | q\ell$. Since $gcd(\alpha, \ell) = 1$ with overwhelming probability, it follows that $\alpha | q$ and setting $a = q/\alpha$ shows that $\zeta = a\ell + r$, i.e. $\zeta = r \bmod \ell$. Also for the same reasoning $q_i\ell + \mu_i r = \zeta_i\alpha + \omega_i$ with overwhelming probability. Repeating the argument for a different ℓ' sampled by the extractor yields a similar equation $\zeta = a'\ell' + r'$, hence $a\ell + r = a'\ell' + r'$ for some $a' = q'/\alpha$. Also $q_i\ell + \mu_i r - \zeta_i\alpha = q_i'\ell' + \mu_i r' - \zeta_i\alpha$. Substituting for r and r' gives $q_i\ell + \mu_i(\zeta - a\ell) = q_i'\ell' + \mu_i(\zeta - a'\ell')$ implying:

$$(q_i - \mu_i a)\ell = (q_i' - \mu_i a')\ell'$$

(This is where it was important that α is fixed by the extractor, as otherwise we could not cancel the $\zeta_i\alpha$ term on each side of the equation). Now since $\ell \neq \ell' \neq 0$ with overwhelming probability, it follows that $\ell | q_i' - \mu_i a'$ and $\ell' | q_i - \mu_i a$. However, $q_i - \mu_i a$ was fixed independently before ℓ' was sampled, hence there is a negligible probability that it has ℓ' as a factor unless $q_i - \mu_i a = 0$, in which case $q_i' - \mu_i a' = 0$ as well. We conclude that with overwhelming probability $q_i\ell + \mu_i r = q_i'\ell' + \mu_i r' = \mu_i\zeta$. In other words, for each ℓ sampled, as long as $Q^{\ell}u^r g^{\alpha r} = wz^{\alpha}$ then with overwhelming probability:

$$wz^{\alpha} = g^{q\ell+\alpha r}\prod_i g_i^{q_i\ell+\mu_i r} = g^{\zeta\alpha}\prod_i g_i^{\mu_i\zeta} = g^{\zeta\alpha}u^{\zeta}$$

Finally, if $u^{\zeta} \neq w$ then $g^{\zeta}/z \neq 1$ and yet $(g^{\zeta}/z)^{\alpha} = u^{\zeta}/w$. Since α is sampled independently from u, w, g, and ζ, this relation can only hold true with non-negligible probability over the choice of α if both g^{ζ}/z and u^{ζ}/w are elements of a small (i.e. $\mathsf{poly}(\lambda)$ size) subgroup generated by g^{ζ}/z. In other words, g^{ζ}/z is an element of low order, and it is possible to compute its order in polynomial time. This would be a contradiction in the generic group model since it is hard to find a non-trivial element and its order (Corollary 5). In conclusion, with overwhelming probability $u^{\zeta} = w$.

Repeating this analysis for each accepting transcript (ℓ_i, Q_i, r_i) shows that $\zeta = r_i \bmod \ell_i$ with overwhelming probability. The remainder of the analysis is identical to the last part of the proof of Theorem 17. Namely, since $\zeta < 2^q$ where $q < \mathsf{poly}(\lambda)$ is an upper bound on the number of queries the adversary makes to the group oracle, we can show there exists a polynomial number of rounds after which Ext would succeed in extracting ζ with overwhelming probability.

$\square$

Proof of Theorem 18.

> For any homomorphism $\phi : \mathbb{Z}^n \to \mathbb{G}$, PoKHP for relation $\mathcal{R}_\phi = \{(w; \vec{x}) : \phi(\vec{x}) = w\}$ is an argument of knowledge in the generic group model.

The proof is a direct generalization of the proof of Theorem 13 for PoKE. As usual, fix $\mathbb{G} \overset{\$}{\leftarrow} GGen(\lambda)$ and $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}$. Let $\mathcal{A}_0, \mathcal{A}_1$ be poly-time generic adversaries where $(w, \mathsf{st}) \overset{\$}{\leftarrow} A_0(\mathbf{g})$ and $\mathcal{A}_1(\mathsf{st})$ runs PoKHP with the verifier $V(\mathbf{g}, w)$. We need to show that for all $\mathcal{A}_1$ there exists a poly-time Ext such that for all $\mathcal{A}_0$ the following holds: if $\mathcal{A}_1$ convinces $V(\mathbf{g}, w)$ to accept with probability at least $1/\text{poly}(\lambda)$ then Ext outputs $\vec{x} \in \mathbb{Z}^n$ such that $\phi(\vec{x}) = w$ with overwhelming probability.

PoKHP includes an execution of PoKRep on $g_1, ..., g_n \in \mathbb{G}$ and input z (the first message sent by the prover to the verifier), and the prover succeeds in PoKHP only if it succeeds in this subprotocol for PoKRep. By Theorem 17 there exists Ext^* for each $\mathcal{A}_1$ that outputs $\vec{x}^*$ such that $Rep(\vec{x}^*) = z$. Furthermore, as shown in the analysis of Theorem 17, once Ext^* has obtained $\vec{x}^*$ it can continue to replay the protocol, sampling a fresh prime $\ell \overset{\$}{\leftarrow} \mathsf{Primes}(\lambda)$, and in each fresh round that produces an accepting transcript it obtains from the Prover values Q, Q' and $\vec{r}$ such that $\vec{r} = x^* \bmod \ell$ with overwhelming probability.

The extractor Ext simply runs Ext^* to obtain $\vec{x}^*$. Now we will show that either $\phi(\vec{x}^*) = w$, i.e. Ext^* extracted a valid witness, or otherwise the adaptive root assumption would be broken, which is impossible in the generic group model (Corollary 4). To see this, we construct an adaptive root adversary $\mathcal{A}_{AR}$ that first runs Ext^* with $\mathcal{A}_0, \mathcal{A}_1$ to obtain $\vec{x}^*$ and provides $h = w/\phi(\vec{x}^*) \in \mathbb{G}$ to the challenger. When provided with $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ from the challenger, $\mathcal{A}_{AR}$ rewinds $\mathcal{A}_1$, passes ℓ to $\mathcal{A}_1$, and with overwhelming probability obtains $Q, \vec{r}$ such that $\vec{x}^* = \vec{r} \bmod \ell$ and $Q^\ell \phi(\vec{r}) = w$. Finally, define $\lfloor \vec{x}^*/\ell \rfloor$ to be the vector obtained by replacing each component x_i with the quotient $\lfloor x_i/\ell \rfloor$. $\mathcal{A}_{AR}$ outputs $v = \frac{Q}{\phi(\lfloor \vec{x}^*/\ell \rfloor)}$. Using the fact that ϕ is a group homomorphism we can show that this is an ℓth root of h:

$$v^\ell = \Big(\frac{Q}{\phi(\lfloor \frac{\vec{x}^*}{\ell} \rfloor)}\Big)^\ell = \frac{Q^\ell}{\phi(\ell \cdot \lfloor \frac{\vec{x}^*}{\ell} \rfloor)} = \frac{Q^\ell}{\phi(\vec{x}^* - \vec{r})} \frac{\phi(\vec{r})}{\phi(\vec{r})} = \frac{w}{\phi(\vec{x}^*)} = h$$

If $w \neq \phi(\vec{x}^*)$ so that $h \neq 1$, then $\mathcal{A}_{AR}$ succeeds in the adaptive root game. In conclusion, the value $\vec{x}^*$ output by Ext satisfies $w = \phi(\vec{x}^*)$ with overwhelming probability.

□

Proof of Theorem 20.

ZKPoKRep is an honest-verifier statistical zero-knowledge argument of knowledge for relation $\mathcal{R}_{\mathsf{Rep}}$ in the generic group model.

Part 1: HVZK To show that the protocol is honest-verifier zero-knowledge we build a simulator Sim. Sim samples $(\tilde{A}, \tilde{c}, \tilde{\ell}, \tilde{\vec{r}}, \tilde{Q})$ as follows. Let $\mathbb{G}_i$ denote the subgroup of $\mathbb{G}$ generated by the base g_i.

1. $\tilde{c} \xleftarrow{\$} [0, 2^\lambda]$, $\tilde{\ell} \xleftarrow{\$} \mathsf{Primes}(\lambda)$
2. $\tilde{\vec{q}} \xleftarrow{\$} [B]^n$
3. $\tilde{\vec{r}} \xleftarrow{\$} [\ell]^n$
4. $\tilde{Q} \leftarrow \prod_{i=1}^n g_i^{\tilde{q}_i}$
5. $\tilde{A} \leftarrow \tilde{Q}^{\tilde{\ell}} (\prod_{i=1}^n g_i^{\tilde{r}_i})^{-1} w^{-\tilde{c}}$.

We now argue that $(\tilde{A}, \tilde{c}, \tilde{\ell}, \tilde{\vec{r}}, \tilde{Q})$ is statistically indistinguishable from a transcript between an honest prover and verifier: $(A, c, \ell, \vec{r}, Q)$. Sim chooses $\tilde{\ell}$ and $\tilde{c}$ identically to the honest verifier in the real protocol. It also solves for $\tilde{A}$ uniquely from the other values such that the verification holds. Therefore, it remains only to show that $\tilde{\vec{r}}$ and $\tilde{Q}$ have the correct distribution. We must show that in the real protocol, independent of ℓ and c, $\vec{r}$ has statistical distance less than $2^{-\lambda}$ from the uniform distribution over $[\ell]^n$ and each $g_i^{q_i}$ has statistical distance less than $2^{-\lambda}$ from uniform over $\mathbb{G}_i$ (recall that $Q = \prod_i g_i^{q_i}$). In addition we must argue that Q and $\vec{r}$ are independent.

For this we use the following facts, which are easy to verify:

1. Fact 1: If Z is uniform random variable over N consecutive integers and $m < N$ then $Z \bmod m$ has statistical distance at most m/N from the uniform distribution over $[m]$.

2. Fact 2: For independent random variables X_1, X_2, Y_1, Y_2, the distance between the joint distributions (X_1, X_2) and (Y_1, Y_2) is at most the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2. Similarly, if these variables are group elements in $\mathbb{G}$, the statistical distance between $X_1 \cdot X_2$ and $Y_1 \cdot Y_2$ is no greater than the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2.

3. Fact 3: Consider random variables X_1, X_2, Y_1, Y_2 with statistical distances $s_1 = \Delta(X1, X2)$ and $s_2 = \Delta(Y_1, Y_2)$, where $Pr(X_1 = x|Y_1 = y) < Pr(X_1 = x) + \epsilon_1$ and $Pr(X_2 = x|Y_2 = y) < Pr(X_1 = x) + \epsilon_2$ for all values x, y. Then the joint distributions (X_1, X_2) and (Y_1, Y_2) have statistical distance at most $s_1 + s_2 + \epsilon_2|supp(X_1)| + \epsilon_1|supp(Y_1)|$, where $supp$ is the support.

Consider fixed values of c, x_i and ℓ. In the real protocol, for each $i \in [n]$ the prover computes $s_i = cx_i + k_i$ where k_i is uniform in $[-B, B]$ and sets $r_i = s_i \bmod \ell$ and $q_i = \lfloor \frac{s_i}{\ell} \rfloor$. The value of s_i is distributed uniformly over a range of $2B + 1$ consecutive integers, thus r_i has statistical distance at most $\ell/(2B + 1)$ from uniform over $[\ell]$. This bounds the distance between r_i and the simulated $\tilde{r_i}$, which is uniform over $[\ell]$.

Next we show that each $g_i^{q_i}$ is statistically indistinguishable from uniform in $\mathbb{G}_i$. Consider the distribution of $\lfloor \frac{s_i}{\ell} \rfloor$ over the consecutive integers in $[\lfloor \frac{cx_i - B}{\ell} \rfloor, \lfloor \frac{cx_i + B}{\ell} \rfloor]$. Denote this by the random variable Z_i. The distribution of $g_i^{q_i}$ over $\mathbb{G}_i$ is determined by the distribution of $q_i \bmod |\mathbb{G}_i|$. The probability that $q_i = z$ is the probability that s_i falls in the interval $[z\ell, (z + 1)\ell - 1]$. This probability is $\ell/(2B + 1)$ for all points where $z\ell \geq cx_i - B$ and

$(z+1)\ell \leq cx_i + B$, which includes all points except possibly the two endpoints $z = \lfloor \frac{cx_i - B}{\ell} \rfloor$ and $z = \lfloor \frac{cx_i + B}{\ell} \rfloor$. Call this set of points Y. The distance of q_i from a uniform random variable U_Y over Y is largest when $cx_i - B = 1 \bmod \ell$ and $cx_i + B = 0 \bmod \ell$. In this case, q_i is one of the two endpoints outside Y with probability $1/B$. For each $z \in Y$, $Pr[q_i = z] = \ell/(2B+1)$. As $|Y| = 2(B-1)/\ell$, the statistical distance of q_i from U_Y is at most: $\frac{1}{2}[Y(\frac{1}{Y} - \frac{\ell}{2B}) + \frac{1}{B}] = \frac{1}{2}(1 - \frac{B-1}{B} + \frac{1}{B}) = \frac{1}{B}$. Moreover, the statistical distance of $q_i \bmod |\mathbb{G}_i|$ from $U_Y \bmod |\mathbb{G}_i|$ is no larger.

As noted in the Fact 1 above, $U_Y \bmod |\mathbb{G}_i|$ has statistical distance at most $|\mathbb{G}_i|/|Y| \leq \ell|\mathbb{G}|/2(B-1) < 1/(n2^{\lambda+1})$ for $B > n2^{2\lambda}|\mathbb{G}|$. By the triangle inequality, the statistical distance of $q_i \bmod |\mathbb{G}_i|$ from uniform is at most $1/B + 1/n2^{\lambda+1} < 1/(n2^{\lambda})$. This also bounds the distance of $g_i^{q_i}$ from uniform in $|G|_i$. The simulated $g_i^{\tilde{q}_i}$ has distance at most $1/(n2^{2\lambda})$ from uniform in G_i since $\tilde{q}_i \bmod |G|_i$ has distance $B/|G|_i < 1/(n2^{2\lambda})$ from uniform (again by the Fact 1 above). By the triangle inequality, the distance between $g_i^{\tilde{q}_i}$ and $g_i^{q_i}$ is at most $1/B + 1/(n2^{\lambda+1}) + 1/(n2^{2\lambda}) < (1/2^{\lambda} + 1/2 + 1/2^{\lambda})1/(n2^{\lambda}) < 1/(n2^{\lambda})$.

We have shown that each r_i is statistically indistinguishable from the simulated $\tilde{r}_i$ and each g^{q_i} is statistically indistinguishable from the simulated $g^{\tilde{q}_i}$. However, we must consider the distances between the joint distributions. Since q_i and r_i are not independently distributed, arguing about the joint distributions requires more work. The simulated $\tilde{q}_i$ and $\tilde{r}_i$ are independent on the other hand.

Consider the conditional distribution of $q_i|r_i$ (i.e. the distribution of the random variable for q_i conditioned the value of r_i). Note that $q_i = z$ if $(s_i - r_i)/\ell = z$. We repeat a similar argument as above for bounding the distribution of q_i from uniform. For each possible value of z, there always exists a unique value of s_i such that $s_i//\ell = z$ and $s_i = 0 \bmod \ell$, except possibly at the two endpoints of the range of q_i (i.e. $e_1 = \lfloor \frac{cx_i - B}{\ell} \rfloor$ and $e_2 = \lfloor \frac{cx_i + B}{\ell} \rfloor$). When r_i disqualifies the two points e_1 and e_2, then each of the remaining points $z \notin \{e_1, e_2\}$ still have equal probability mass, and thus the probability $Pr(q_i = z|r_i)$ increases by at most $[Pr(q_i = e_1) + Pr(q_i = e_2)]/(2\lfloor \frac{B}{\ell} \rfloor) < 1/B^2$. The same applies to the variable $q_i|r_i \bmod |G|_i$ and hence the variable $g^{q_i}|r_i$.

We can compare the joint distribution $X_i = (g^{q_i}, r_i)$ to the simulated $Y_i(g^{\tilde{q}_i}, \tilde{r}_i)$ using

Fact 3 above. Setting $\epsilon_1 = 1/B^2$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $1/(n2^\lambda) + \ell/(2B+1) + \ell/B^2$. Moreover, as each $X_i = (g^{q_i}, r_i)$ is independent from $X_j = (g^{q_j}, r_j)$ for $i \neq j$, we use Fact 2 to bound the distances between the joint distributions $(g^{q_1}, ..., g^{q_n}, r_1, ..., r_n)$ and $(g^{\tilde{q_1}}, ..., g^{\tilde{q_n}}, \tilde{r_1}, ..., \tilde{r_n})$ by the sum of individual distances between each X_i and Y_i, which is at most $1/2^\lambda + n\ell/(2B+1) + n\ell/B^2 < 2^{-\lambda+1}$. Finally, this also bounds the distance between $(Q, \vec{r})$ and $(\tilde{Q}, \tilde{\vec{r}})$ where $Q = \prod_i g^{q_i}$ and $\tilde{Q} = \prod_i g^{\tilde{q_i}}$.

Part 2: PoK For extraction we describe an efficient extractor Ext. Ext randomly samples two random challenges c and c', and $c \neq c'$ with probability $\frac{1}{2^\lambda}$. Ext then uses the extractor from *Theorem* 17 to extract $\vec{s}$ and $\vec{s}'$ such that $\prod_{i=1}^n g_i^{s_i} = Aw^c$ and $\prod_{i=1}^n g_i^{s_i'} = Aw^{c'}$. We now compute $\Delta s_i = s_i - s_i'$ for all $i \in [1, n]$ and $\Delta c = c - c'$. This gives us $\prod_{i=1}^n g_i^{\Delta s_i} = w^{\Delta c}$. We now claim that $\Delta c \in \mathbb{Z}$ divides $\Delta s_i \in \mathbb{Z}$ for each $i \in [1, n]$ with overwhelming probability and that $\prod_{i=1}^n g_i^{\Delta s_i/\Delta c} = w$. By Lemma 22, we can write $w = \prod_{i=1}^m g_i^{\alpha_i}$, for integers $\alpha_i \in \mathbb{Z}$ that can be efficiently computed from $\mathcal{A}$'s queries to the generic group oracle. Since $\prod_{i=1}^n g_i^{\Delta s_i} = w^{\Delta c}$ it follows by Lemma 24 that, with overwhelming probability, $\alpha_j = 0$ for all $j > n$ and $\Delta s_i = \alpha_i \Delta c$ for all $i \in [1, n]$.

Furthermore, if $\mu = \prod_{i=1}^n g_i^{\Delta s_i/\Delta c} \neq w$, then since $\mu^{\Delta c} = \prod_{i=1}^n g_i^{\Delta s_i} = w^{\Delta c}$ it would follow that μ/w is an element of order $\Delta c > 1$. As Δc is easy to compute this would contradict the hardness of computing a non-trivial element and its order in the generic group model (Corollary 5). We can conclude that $\mu = w$ with overwhelming probability. The extractor outputs $\vec{\alpha} = (\alpha_1, ..., \alpha_n)$ where $\alpha_i = \Delta s_i/\Delta c$. □

Proof of Theorem 21.

> ZKPoKE is an honest-verifier statistically zero-knowledge argument of knowledge for relation $\mathcal{R}_{\mathsf{PoKE}}$ in the generic group model.

To prove that the protocol is honest-verifier zero-knowledge we build a simulator Sim which generates valid transcripts that are statistically indistinguishable from honestly generated ones. The simulator generates a transcript as follows:

1. $\tilde{c} \xleftarrow{\$} [0, 2^\lambda], \tilde{\ell} \xleftarrow{\$} \mathsf{Primes}(\lambda)$

2. $\tilde{z} \leftarrow h^{\tilde{\rho}}, \rho \xleftarrow{\$} [B]$

3. $\tilde{q}_x, \tilde{q}_r \xleftarrow{\$} [B]^2$

4. $\tilde{r}_x, \tilde{r}_\rho \in [\ell]^2$

5. $\tilde{Q}_g \leftarrow g^{\tilde{q}_x} h^{\tilde{q}_\rho}, \tilde{Q}_u \leftarrow u^{\tilde{q}_x}$

6. $\tilde{A}_g \leftarrow \tilde{Q}_g^{\ell} g^{\tilde{r}_x} h^{\tilde{r}_\rho} z^{-\tilde{c}}, \tilde{A}_u \leftarrow \tilde{Q}_u^{\ell} u^{\tilde{r}_x} w^{-\tilde{c}}$

We now argue that the transcript $(\tilde{z}, \tilde{A}_g, \tilde{A}_u, \tilde{c}, \tilde{\ell}, \tilde{Q}_g, \tilde{Q}_u, \tilde{r}_x, \tilde{r}_\rho)$ is statistically indistinguishable from a transcript between an honest prover and verifier: $(z, A_g, A_u, c, \ell, Q_g, Q, u, r_x, r_\rho)$ $\tilde{\ell}, \tilde{c}$ are identically chosen as by the random verifier and $\tilde{A}_g, \tilde{A}_u$ are uniquely defined by the rest of the transcript and the verification equations. It thus suffices to argue that $\tilde{z}, \tilde{Q}_g, \tilde{Q}_u, \tilde{r}_x, \tilde{r}_\rho$ as well as z, Q_g, Q_u, r_x, r_ρ are statistically indistinguishable from uniform in their respective domain.

Using Fact 1 stated in the proof of Theorem 20 and that $B > 2^\lambda |\mathbb{G}|$ we can see that $\tilde{z}$ is indistinguishable from a uniform element in the subgroup of $\mathbb{G}$ generated by h. Since g and h generate the same subgroup the same argument applies to z. For $\tilde{Q}_g, \tilde{Q}_u, \tilde{r}_x, \tilde{r}_\rho$ and Q_g, Q_u, r_x, r_ρ the same argument as in the proof of $Theorem$ 20 apply, showing that all values are nearly uniform. The simulation therefore produces valid, statistically indistinguishable transcripts. Note that the requirement that g, h generate the same group can be relaxed under computational assumptions. The assumption states that it is difficult to distinguish between g, h which generate the same subgroup and g', h' which don't. Given this we can use a hybrid argument which replaces g', h' with g, h and the applies the same simulation argument as above.

For extraction, note that the protocol contains ZKPoKRep as a subprotocol on input A_g and bases g, h in the CRS, and therefore we can use the ZKPoKReP and PoKRep extractors to extract x, ρ such that $z = g^x h^\rho$ and s_1, s_2 such that $g^{s_1} h^{s_2} = A_g z^c$ with overwhelming probability. Moreover, as shown in the analysis for the PoKRep extractor, we can rewind the adversary on fresh challenges so that each accepting transcript outputs an r_1, ℓ where $s_1 = r_1 \bmod \ell$ with overwhelming probability. If $u^{s_1} \neq A_u w^c = Q_u^\ell u^{r_1}$ then $\gamma = (r_1 - s_1)/\ell$

is an integer and $Q_u u^\gamma$ is an ℓth root of $A_u w^c / u^{s_1} \neq 1$. This would break the adaptive root assumption, hence by Corollary 4 it follows that $u^{s_1} = A_u w^c$ with overwhelming probability.

Recall from the analysis of Theorem 20 that the extractor obtains a pair of accepting transcripts with $s_1, s_2, s'_1, s'_2, c, c'$ so that $x = \Delta s_1/\Delta c = (s_1 - s'_1)/(c - c')$ and $\rho = \Delta s_2/\Delta c = (s_2 - s'_2)/(c - c')$. Since $u^{s_1} = A_u w^c$ and $u^{s'_1} = A_u w^{c'}$ with overwhelming probability, we obtain $u^{\Delta s_1} = w^{\Delta c}$ with overwhelming probability. Finally, this implies $(u^x)^{\Delta c} = w^{\Delta c}$. If $u^x \neq w$, then u^x/w is a non-trivial element of order Δc, which would contradict the hardness of computing a non-trivial element and its order in the generic group model (Corollary 5). Hence, we conclude that $u^x = w$ with overwhelming probability.

□

4.12 Non-interactive PoE and PoKE variants

```
NI-PoE

$\{x, u, w : u^x = w\}$
$\mathsf{Prove}(x, u, w)$ :
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(x, u, w)$
    $q \leftarrow \lfloor x/\ell \rfloor$
    $Q \leftarrow u^q$

$\mathsf{Verify}(x, u, w, Q)$ :
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(x, u, w)$
    $r \leftarrow x \bmod \ell$
    Check: $Q^\ell u^r = w$
```

```
NI-PoKE2

$\{(u, w; x) : u^x = w\}$
$\mathsf{Prove}(x, u, w)$ :
    $g \leftarrow \mathsf{H}_{\mathbb{G}}(u, w), z = g^x$
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(u, w, z), \alpha = \mathsf{H}(u, w, z, \ell)$
    $q \leftarrow \lfloor x/\ell \rfloor, r \leftarrow x \bmod \ell$
    $\pi \leftarrow \{z, (ug^\alpha)^q, r\}$
$\mathsf{Verify}(u, w, z, Q, r)$ :
    $g \leftarrow \mathsf{H}_{\mathbb{G}}(u, w)$
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(u, w, z), \alpha \leftarrow \mathsf{H}(u, w, z, \ell)$
    Check: $Q^\ell (ug^\alpha)^r = wz^\alpha$
```

```
NI-PoDDH

${(y_1, y_2, y_3); (x_1, x_2) : g^{x_1} = y_1 \wedge g^{x_2} = y_2 \wedge y_1^{x_2} = y_3$
Prove$(\mathbf{x} = (x_1, x_2), \mathbf{y} = (y_1, y_2, y_3))$ :
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(\mathbf{y})$
    $(q_1, q_2) \leftarrow (\lfloor x_1/\ell \rfloor, \lfloor x_2/\ell \rfloor)$
    $(r_1, r_2) \leftarrow (x_1 \bmod \ell, x_2 \bmod \ell)$
    $\pi \leftarrow \{(g^{q_1}, g^{q_2}, y_1^{q_2}), r_1, r_2\}$
Verify$(\mathbf{y}, \pi)$ :
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(\mathbf{y})$
    $\{Q_{y_1}, Q_{y_2}, Q_{y_3}, r_1, r_2\} \leftarrow \pi$
    Check:
 $\mathbf{r} \in [\ell]^2 \wedge Q_{y_1}^{\ell} g^{r_1} = y_1 \wedge Q_{y_2}^{\ell} g^{r_2} = y_2 \wedge Q_{y_3}^{\ell} y_1^{r_2} = y_3$
```

```
NI-ZKPoKE

$\{(u, w; x) : u^x = w\}$
Prove$(x, u, w)$ :
    $k, \rho_x, \rho_k \xleftarrow{\$} [-B, B]$;   $z = g^x h^{\rho_x}$;   $A_g = g^k h^{\rho_k}$;   $A_u = u^k$;
    $\ell \leftarrow \mathsf{H}_{\mathsf{prime}}(u, w, z, A_g, A_u)$;   $c \leftarrow \mathsf{H}(\ell)$;
    $q_x \leftarrow \lfloor (k + c \cdot x)/\ell \rfloor$;   $q_\rho \leftarrow \lfloor (\rho_k + c \cdot \rho_x)/\ell \rfloor$;
    $r_x \leftarrow (k + c \cdot x) \bmod \ell$;   $r_\rho \leftarrow (\rho_k + c \cdot \rho_x) \bmod \ell$;
    $\pi \leftarrow \{\ell, z, g^{q_x} h^{q_\rho}, u^{q_x}, r_x, r_\rho\}$
Verify() :
    $\{c, z, Q_g, Q_u, r_x, r_\rho\} \leftarrow \pi$
    $c = \mathsf{H}(\ell)$   $A_g \leftarrow Q_g^{\ell} g^{r_x} h^{r_\rho} z^{-c}$;   $A_u \leftarrow Q_u^{\ell} u^{r_x} w^{-c}$
    Check: $r_x, r_\rho \in [\ell]$;  $\ell = \mathsf{H}_{\mathsf{prime}}(u, w, z, A_g, A_u)$
```

Chapter 5

Tight proofs of space and replication

Proof-of-space (PoS) has been proposed as an alternative to proof-of-work (PoW) for applications such as SPAM prevention, DOS attacks, and Sybil resistance in blockchain-based consensus mechanisms [77, 92, 130]. Several industry projects[1] are underway to deploy cryptocurrencies similar to Bitcoin that use proof-of-space instead of proof-of-work. Proof-of-space is promoted as more egalitarian and eco-friendly that proof-of-work because it is ASIC-resistant and does not consume its resource (space instead of energy), but rather reuses it.

A PoS is an interactive protocol between a prover and verifier in which the prover uses a minimum specified amount of space in order to pass verification. The protocol must have compact communication relative to the prover's space requirements and efficient verification. A PoS is *persistent* if repeated audits force the prover to utilize this space over a period of time. More precisely, there is an "offline" phase in which the prover obtains challenges from a verifier, generates a (long) string σ that it stores, and outputs a compact verification tag τ to the verifier. (The offline phase can be made non-interactive using the Fiat-Shamir transform). This is followed by an "online" challenge-response protocol in which the verifier

[1]`https://chia.net/`,`https://spacemesh.io/`, `https://filecoin.io/`

uses τ to generate challenges and the prover uses σ to efficiently compute responses to the verifier's challenges.

The soundness of the PoS relies on a time bound on the online prover that is enforced by frequent verifier audits. A time bound is necessary as otherwise the prover could store its compact transcript and simulate the setup to re-derive the advice whenever it needs to pass an online proof. If the PoS guarantees that an adversary must use the minimum amount of space to pass challenges within the wall-clock time allotted no matter how much (polynomially bounded) computation it expends then the PoS is said to resist parallelization attacks. A PoS must resist parallelization attacks in order to be considered unconditionally secure. Otherwise, the security may still be reasoned through a cost benefit analysis for a *rational* prover, who will not expend significant computation to save a relatively small fraction of space.

More formally, correctness and (S, T, μ)-soundness for a PoS protocol is defined as follows. First, if the prover commits to persistently utilize N blocks of space then the honest prover algorithm defined by the protocol must use $O(N)$ persistent space and must succeed in passing the verifier's challenges without error. Next, a pair of offline/online adversaries is considered. The "offline" adversary generates an adversarial string σ' and offline proof π'. An (S, T, μ)-sound protocol guarantees that if the string length of σ' output by the offline adversary is less than S then either the verifier accepts π' with negligible probability or otherwise any online adversary who runs in time less than T on the input σ' and the verifier's challenge will fail verification with probability at least $1 - \mu$.

There is generally a gap between the honest space utilization and the lower bound S on the adversary's space. If the honest prover uses S' space and some adversary is able to use $(1 - \epsilon)S'$ space then this PoS protocol has at least an ϵ *space gap*. Loosely speaking, a *tight PoS* construction makes ϵ arbitrarily small. The construction is allowed to involve ϵ as a parameter, and the value of ϵ may impact efficiency. All else equal, a tighter PoS is more desirable as it has tighter provable security. Nearly all existing PoS constructions have enormous space gaps, including those that are currently being used in practice [4, 77]. The one exception is a recent PoS protocol by Pietrzak [133]. Although this PoS construction is provably tight, the concrete parameters required in the analysis result in an impractically large offline proof.

Proof-of-replication (PoRep) [3, 42, 133, 80] is a recently proposed variant of PoS. A PoRep demonstrates that the prover is dedicating unique resources to storing a retrievable copy of a committed data file, and is therefore a *useful proof of space*. It has been proposed as an alternative Sybil resistance mechanism (e.g. for a blockchain) that is not only ASIC resistant and eco-friendly, but also has a useful side-effect: file storage. Furthermore, since the prover may run several independent PoReps for the same file that each require unique resources, PoReps may be used as a publicly verifiable proof of data replication/duplication. Unfortunately, it is not possible to cryptographically guarantee that a prover is persistently storing data in a replicated format. A prover can always sabotage the format (e.g. by encrypting it and storing the key separately) and can then recover the original format quickly when challenged. The best we can do is limit the adversary's incentive to cheat, i.e. by designing a protocol in which the adversary can save at most an ϵ fraction of its space by deviating from storing the data in a replicated format. This implies a PoS with an ϵ space gap.

The goal of this work is to construct a practical and provably tight PoS that can also be used as a PoRep.

5.1 Summary of results

We construct a new tight PoS based on graph labeling with asymptotic proof size $O(\lambda \log n/\epsilon)$ where ϵ is the achieved space gap and λ is a security parameter. We can instantiate this construction with relatively weak[2] depth robust graphs that do not require any special properties other than retaining $\Omega(n)$ depth in subgraphs on some constant fraction of the nodes bounded away from 1 (e.g. our concrete analysis assumes 80%).

[2]There is experimental evidence that a simple DRG construction with concretely small constant degree (even degree 2) has this property on a graph of size $n = 2^{20}$ [8]. With a constant degree DRG and vector commitment scheme with constant size openings the resulting proof size would be just $O(\lambda/\epsilon)$

PoS from Stacked DRGs and Expanders Our basic approach is a combination of the stacked bipartite expanders of Ren and Devadas [141] with depth robust graphs. Instead of stacking λ path graphs we stack $O(\log(1/\epsilon))$ levels of fixed-degree DRGs where ϵ is a construction parameter. We refer to this graph construction as Stacked DRGs. We are able to show that this results in a PoS that has only an ϵ space gap. Intuitively, the expander edges between layers amplify the dependence of nodes on the last layer and nodes on earlier layers so that deletion of a small ϵ fraction of node labels on the last level will require re-derivation of nearly all the node labels on the first several layers. Thus, since every layer is a DRG, recomputing the missing ϵ fraction of labels requires $\Omega(N)$ sequential computation. It is easy to see that this would be the case if the prover were only storing $(1-\epsilon)n$ labels on the last level and none of the labels on earlier levels, however the analysis becomes much more difficult when the prover is allowed to store any arbitrary $(1-\epsilon)n$ labels. This analysis is the main technical contribution of this work. Concretely, we analyze the construction with an $(n, 0.80n, \Omega(n))$ DRG, i.e. deletion of 20% of nodes leaves a high depth graph on the 80% remaining nodes, regardless of the value of ϵ.

Our construction is efficient compared to prior constructions of tight PoS primarily because we can keep the degree of the graphs fixed for arbitrary ϵ while keeping the number of levels proportional to $\log(1/\epsilon)$. In a graph labeling PoS, the offline PoS proofs sample $O(1/\epsilon)$ labels along with their parent labels, which the verifier checks for consistency. Thus, any construction based on this approach that requires scaling the degree of graphs by $1/\epsilon$ also scales the proof size by $1/\epsilon$, resulting in a proof complexity of at least $O(1/\epsilon^2)$. In our stacked DRG PoS construction the offline proof must include queries from each level to prove that each level of computed labels are "mostly" correct. If done naively, $O(1/\epsilon)$ challenge labels are sampled from each level, resulting in a proof complexity $O(d/\epsilon \cdot \log(1/\epsilon))$ where d is the degree of the level graphs. This is already an improvement, however with a more delicate analysis we are able to go even further and show that the total number of queries over all layers can be kept at $O(1/\epsilon)$, achieving an overall proof complexity $O(d/\epsilon)$.[3]

[3]Asymptotically, this is close to the optimal proof complexity achievable for any PoS based on graph labeling that has an ϵ space gap. If the prover claims to be storing n labels and the proof queries less than $1/\epsilon$ then a random deletion of an ϵ fraction of these labels evades detection with probability at least

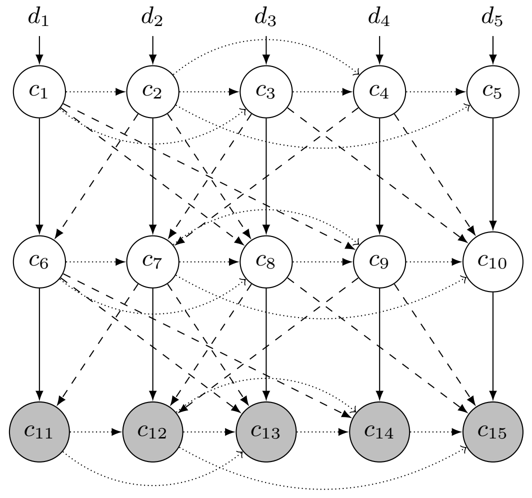

Figure 5.1: **Stacked DRGs.** Dotted edges are the DRG edges and dashed edges are expander edges. In the PoS on Stacked DRGs the prover computes a labeling of the graph and stores the labels on the nodes in green.

The PoS on Stacked DRGs can also be used as the basis for a PoRep that satisfies ϵ-rational replication for arbitrarily small ϵ. The PoRep simply uses the labels on the $\ell - 1$st level as keys to encode the n-block data input $D = d_1, ..., d_n$ on the ℓth (last) level, using the same method described earlier for encoding data into the labels of a PoS (see Related Work, [133, 42, 80]).

5.2 Additional related work

The original PoS of Dziembowski et. al. [77] was based on hard to pebble directed acyclic graphs (DAGs), using a blend of techniques from superconcentrators, random bipartite expander graphs and depth robust graphs [132]. During the offline initialization the prover computes a *labeling* of the graph using a collision-resistant hash function where the label e_v on each node $v \in \mathcal{G}$ of the graph is the output of the hash function on the labels of all parent nodes of v. It outputs a commitment to this labeling along with a proof that the committed labeling was "mostly" correct. This offline proof consists of randomly sampled

$(1-\epsilon)^{1/\epsilon} \approx 1/e$.

labels and their parent labels, which the verifier checks for consistency. During the online challenge-response phase the verifier simply asks for random labels that the prover must produce along with a standard proof that these labels are consistent with the commitment. The construction leaves a space gap of at least $1 - \frac{1}{512}$.

Ren and Devadas [141] construct a PoS from stacked bipartite expander graphs that dramatically improved on the space gap, although it is not secure against parallel attacks. Their construction involves λ levels $V_1, ..., V_\lambda$ consisting of n nodes each, with edges between the layers defined by the edges of a constant-degree bipartite expander. The prover computes a labeling of the graph just as in the Dziembowski et. al. PoS, however it only stores the labels on the final level. Their construction still leaves a space gap of at least[4] 1/2.

Recently, Abusalah et. al. [4] revived the simple PoS approach based on storing tables of random functions. The basic idea is for the prover to compute and store the function table of a random function $f : [N] \to [N]$ where f is chosen by the verifier or a random public challenge. During the online challenge-response the verifier asks the prover to invert f on a randomly sampled point $x \in [n]$. Intuitively, a prover who has not stored most of the function table will likely have to brute force $f^{-1}(x)$, performing $\Omega(N)$ work. This simple approach fails to be a PoS due to Hellman's time/space tradeoffs, which enable a prover to succeed with S space and T computation for any $ST = O(N)$. However, Abusalah et. al. build on this approach to achieve a provable time/space tradeoff of $S^k T = \Omega(\epsilon^k N^k)$. This PoS is not secure against parallel attacks, and also has a very large (even asymptotic) space gap.

Pietrzak [133] and Bonneau et. al. [42, 80] independently proposed simpler variants of the graph labeling PoS by Dziembowski et. al. based solely on pebbling a depth robust graph (DRG). A degree d DAG on n nodes is (α, β)-depth robust if any subgraph on αn nodes contains a path of at least length βn. It is trivial to construct DRGs of large degree (a complete DAG is depth robust), but much harder to construct DRGs with small degree. Achieving constant α, β is only possible asymptotically with degree $\Omega(\log N)$. The graph

[4]For practical parameters, the Ren and Devadas construction has a space gap larger than 1/2. For example, it requires a graph of degree at least 40 in order to achieve a space gap of less than 2/3.

labeling PoS on a DRG results in a PoS with an α space gap that is also secure against parallel attacks. Fisch et. al. also suggested combining this labeling PoS with a *verifiable delay function* (VDF) [38] to increase the expense of labeling the graph without increasing the size of the proof verification complexity. The delay on the VDF can be tuned depending on the value of n. Both of these constructions were proposed for PoReps. In this variant of the PoS protocol, the prover uses the labeling of the graph to encode a data file on n blocks $D = d_1, ..., d_n$. The ith label e_i is computed by first deriving a key k_i by hashing the labels on the parents of the ith node, and then setting $e_i = k_i \oplus d_i$. If all the labels are stored then any data block can be quickly extracted from e_i by recomputing k_i. More generally, this *DAG encoding* of the data input could use any encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$, where Enc is sequentially slow and Dec is fast, in order to derive $e_i = \mathsf{Enc}(k_i, d_i)$. The data is decoded by computing $d_i = \mathsf{Dec}(k_i, e_i)$.

The labeling PoS on a DRG is not technically a tight PoS because decreasing α also decreases the time bound βn on the prover's required computation to defeat the PoS. Moreover, while there exist constructions of (α, β)-DRGs for arbitrarily small α, these constructions have concretely very high degrees and are thus not useful for building a practical PoS. Pietrzak [133] improved on the basic construction by relying on a stronger property of special DRGs [132, 9] that have degree $\Omega((\log n)/\epsilon)$ and are (α, β)-depth robust for all (α, β) such that $1 - \alpha + \beta \geq 1 - \epsilon$. This DRG can be constructed for any value of $\epsilon < 1$. In Pietrzak's PoS, the prover builds a DRG on $4n$ nodes and only stores the labels on the topologically last n nodes. This can similarly be used as a PoRep where the data is encoded only on the last level and the labels on previous levels are just used as keys. This PoRep has a slow data extraction time because extracting the data requires recomputing most of the keys from scratch, which is as expensive as the PoS initialization.

Pietrzak shows that a prover who deletes an ϵ' fraction of the labels on the last n nodes will not be able to re-derive them in fewer than n sequential steps. The value ϵ' can be made arbitrarily small, but at the expense of increasing the degree of the graph proportionally to $1/\epsilon'$. The resulting proof has asymptotic size $\Omega((\log n)/\epsilon^2)$. Moreover, although these special DRGs achieve asymptotic efficiency, their current analysis requires the graphs to have impractically large degrees. According to the analysis in [9], achieving just a 0.05 space gap would require instantiating these graphs with degree at least $27,600 \log n$. The proof size is proportional to the graph degree, so to achieve the space gap $\epsilon = 0.05$ with soundness $\mu = 2^{-10}$ and $n = 2^{30}$ the proof size would be at least 2.6 GB.

Boneh et. al. [38] describe a simple PoRep (also a PoS) just based on storing the output of a *verifiable delay function* (VDF) on n randomly sampled points, which generalizes an earlier proposal by Sergio Demian Lerner [111]. This is in fact an arbitrarily tight PoS with very practical proof sizes (essentially optimal). However, the time complexity of initializing the prover's $O(n)$ storage is $O(n^2)$, and therefore is not practically feasible for large n. This construction is similar to the PoS based on storing function tables [4], but uses the VDF as a moderately hard (non-parallelizable) function on a much larger domain (exponential in the security parameter) and stores a random subset of its function table. The reason for the large initialization complexity is that the prover cannot amortize its cost of evaluating the VDF on the entire subset of points.

Cecchetti et. al. [61] developed a similar idea to our stacked DRGs, using butterfly networks instead of expander graphs, to construct a primitive they call publicly incompressible encodings (PIEs).

5.3 Preliminaries

5.3.1 Proofs of Space

A PoS interactive protocol has three procedures:

1. Setup The setup runs on security parameters λ and outputs public parameters pp for the scheme. The public parameters are implicit inputs to the next two protocols.

2. Initialization is an interactive protocol between a prover P and verifier V that run on shared input (id, N). P outputs Φ and S, where S is its storage advice of length N and Φ is a compact $O(polylog(N))$ length string given to the verifier.

3. Execution is an interactive protocol between P and V where P runs on input S and V

runs on input Φ. V sends challenges to P, obtains back a proof π, and outputs accept or reject.

Efficiency. The commitment Φ is $O(polylog(N))$ size, the storage S is size N, and the verifier runs in time $O(polylog(N))$.

Completeness. The prover succeeds with probability 1 (causes verifier to accept) if it follows the protocol honestly.

Soundness. The PoS is (s, t, μ)-sound if for all adversaries P^* running in time t and storing advice of size s during $\mathsf{Execution}$, P^* passes verification with probability at most μ. The PoS is parallel (s, t, μ)-sound if P^* may run in parallel time t. We say that a PoS construction is *tight* if for any constant $\epsilon < 1$ the PoS can be parametrized so that the resulting PoS is $(\epsilon N, t, \mu)$-sound for $t \in \Omega(N)$ and $\mu = \mathsf{negl}(\lambda)\,(\lambda)$.

Amortization-free soundness An (s, t, μ) PoS is *amortization-free* if for any k distinct ids $id_1, ..., id_k$, the modified PoS protocol that runs $\mathsf{Initialization}$ on k independent inputs (id_i, N) for each i to get outputs (S_i, Φ_i) and then runs $\mathsf{Execution}$ independently on each (S_i, Φ_i) is $(ks, t, k\mu)$-sound.

5.3.2 Graph pebbling games

Pebbling games are the main analytical tool used in graph-based proofs of space and memory hard functions.

Black pebbling game The black pebbling game is a single-player game on a DAG $\mathcal{G} = (V, E)$. At the start of the game the player chooses a starting configuration of $P_0 \subseteq V$ of vertices that contain black pebbles. The game then proceeds in rounds where in each round the player may place a black pebble on a vertex only if all of its parent vertices currently contain pebbles placed in some prior round. In this case we say that the vertex is *available*. Placing a pebble constitutes a *move*, whereas placing pebbles on all simultaneously available vertices consumes a *round*. The adversary may also remove any black pebble at any point. The game stops once the adversary has placed pebbles on all vertices in some target/challenge set $V_C \subseteq V$.

Pebbling complexity The pebbling game on graph G with vertex set V and target set $V_C \subseteq V$ is (s,t)-*hard* if no player can pebble the set V_C in t moves (or fewer) starting from s initial pebbles, and is (s,t)-*parallel-hard* if no player can complete the pebbling in t rounds (or fewer) starting from an initial configuration of at most s pebbles. If a $1-\alpha$ fraction of the nodes in V_C each require t rounds to pebble then the pebbling game on (G, V_C) is (s,t,α)-parallel-hard, i.e. every subset containing more than an α fraction of the nodes in VC requires t rounds to pebble.

In a random pebbling game a challenge node is sampled randomly from V_C after the player commits to the initial configuration P_0 of s vertices, and the hardness measure includes the adversary's probability of success. The random pebbling game is (s,t,ϵ)-(parallel)-hard if from any s fixed initial pebbles the probability that a uniformly sampled challenge node can be pebbled in t or fewer moves (resp. t or fewer rounds) is less than ϵ.

Fact 4. *The random pebbling game on a DAG G on n nodes with target set V_C is (s,t,α)-parallel-hard if and only if the deterministic pebbling game on G with target set V_C is (s,t,α)-parallel-hard.*

Proof. Fix any P_0 of size s. If the random pebbling game on G with target set V_C is (s,t,α)-parallel-hard then less than an α fraction of the nodes in V_C can be pebbled individually in t rounds starting from P_0. Therefore, every subset U in V_C of size $\alpha|V_C|$ contains at least one node that cannot be pebbled individually in t rounds, hence the (deterministic) pebbling game is (s,t,α)-parallel-hard. Conversely, if G is (s,t,α)-parallel-hard then less than an α fraction of nodes in V_C can be pebbled individually in r rounds. Otherwise, these nodes form a subset U of size $\alpha|V_C|$ and they can all be simultaneously pebbled in parallel in t rounds. This implies that the probability a randomly sampled node from V_C can be pebbled in t rounds is less than α. □

Fact 5. *A random pebbling game with a single challenge is (s, t, α)-parallel-hard if and only if the the random pebbling with κ challenges is (s, t, α^k)-parallel-hard.*

Proof. If the random pebbling game is (s, t, α) hard then by Fact 4 the deterministic pebbling game is (s, r, α) hard, hence there are at most an α fraction of the nodes in V_C that can be pebbled in r rounds from s initial pebbles. The probability that κ independent random challenges are all nodes from this α fraction is at most α^κ. Conversely, if the random pebbling game is not (s, t, α) hard then the adversary can pebble all the κ challenges simultaneously in parallel time t succeeding on each challenge individually with probability greater than α, hence succeeding on all the challenges with probability greater that α^κ. $\square$

DAG labeling game A labeling game on a degree d DAG $\mathcal{G}$ is analogous to the pebbling game, but involves a cryptographic hash function $H : \{0,1\}^{dm} \to \{0,1\}^m$, often modeled as a random oracle. The vertices of the graph are indexed in $[n]$ and each ith vertex associated with the label c_i where $c_i = H(i)$ if i is a source vertex, or otherwise $c_i = H(i||c_{\mathsf{parents}}(i))$ where $c_{\mathsf{parents}}(i) = \{c_{v_1}, ..., c_{v_d}\}$ if $v_1, ..., v_d$ are the parents of the ith vertex, i.e. the vertices with a directed edge to vertex i. The game ends when the player has computed all the labels on a target/challenge set of vertices V_C. A "fresh" labeling of $\mathcal{G}$ could be derived by choosing a salt id for the hash function so that $H_{id}(x) = H(id||x)$, and the labeling may be associated with the identifier id.

The complexity of the labeling game (on a fresh identifier id) is measured in queries to the hash function instead of pebbles. This includes the number of labels initially stored, the total number of queries, and the total rounds of sequential queries, etc. The labeling game is $(s, r, q, \epsilon, \delta)$-labeling-hard if no algorithm that stores initial advice of size s and after receiving a uniform random challenge node $v \in [n]$ makes a total of q queries to H in r sequential rounds can output the correct label on v with probability greater than ϵ over the challenge v and δ over the random oracle H.

Random oracle query complexity A general correspondence between the complexity of the black pebbling game on the underlying graph $\mathcal{G}$ and the random oracle labeling game is not yet known. However, Pietrzak [133] recently proved an equivalence between the parallel hardness of the randomized pebbling game and the parallel hardness of the random oracle labeling game for arbitrary initial configurations S_0 adapting the "ex post facto" technique from [70].

Theorem 22 (Pietrzak[133])**.** *If the random pebbling game on a DAG G with n nodes and in-degree d is (s, r, ϵ)-parallel-hard then the labeling game on G with a random oracle $H : \{0,1\}^{md} \to \{0,1\}^m$ is $(s', r, \epsilon, \delta, q)$-labeling-hard with $s' = s(m - 2(\log n + \log q)) - \log(1/\delta)$.*

Generic PoS from graph labeling game Many PoS constructions are based on the graph labeling game [77, 141, 133]. Let $\mathcal{G}(\cdot)$ be a family of d-in-regular DAGs such that $G_n \leftarrow \mathcal{G}(n)$ is a d-in-regular DAG on $N > n$ nodes and $V_C(n)$ is a subset of n nodes from G_n. Let $H : \{0,1\}^{dm} \to \{0,1\}^m$ be a collision-resistant hash function (or random oracle). Let $\mathsf{Chal}(n, \Lambda)$ denote a distribution over challenge vectors in $[N]^\lambda$. For each $n \in \mathbb{N}$, the generic PoS based on the labeling game with G_n and target set $V_C(n)$ is as follows:

$\mathsf{Initialization}$: The prover plays the labeling game on G_n using a hash function $H_{id} = H(id||\cdot)$. The prover does the following:

1. Computes the labels $c_1, ..., c_N$ on all nodes of $\mathcal{G}$ and commits to them in com using any vector commitment scheme.

2. Obtains vector of λ challenges $\vec{r} \leftarrow \mathsf{Chal}(n)$ from the verifier (or non-interactively derives them using as a seed $H_{id}(com)$).

3. For challenges $r_1, ..., r_\lambda$, the prover opens the label on the r_ith node of G_n, which was committed in com, as well as the labels $c_{\mathsf{parents}}(r_i)$ of all its parent nodes. The labels are added to a list L with corresponding opening proofs in a list Λ and the prover outputs the proof $\Phi = (com, L, \Lambda)$.

The verifier checks the openings Λ with respect to com. It also checks for each challenge specifying an index $v \in [N]$, the label c_v in L label and its parent labels $c_{\mathsf{parents}}(c_v)$, that

$c_v = H_{id}(v||c_{\mathsf{parents}}(c_v))$. Finally, the prover stores as S only the n labels in V_C.

Execution: The verifier selects κ challenge nodes $v_1, ..., v_\kappa$ uniformly at random from V_C. The online prover uses its input S to respond with the label on v and an opening of *com* at the appropriate index. The verifier can repeat this sequentially, or ask for a randomly sampled vector of challenge vertices to amplify soundness.

Red-black pebbling game An adversary places both black and red pebbles on the graph initially. The red pebbles correspond to incorrect labels that the adversary computes during Initialization and the black pebbles correspond to labels the adversary stores in its advice S. Without loss of generality, an adversary that cheats generates some label that does not require any space to store, which is why red pebbles will be "free" pebbles and counted separately from black pebbles. The adversary's choice of red pebble placements (specifically how many to place in different regions of the graph) is constrained by the λ non-interactive challenges, which may catch these red pebbles and reveal them to the verifier. The formal description of the red-black pebbling security game for a graph labeling PoS construction with $\mathcal{G}(n)$, $V_C(n)$, and $\mathsf{Chal}(n)$ is as follows.

$\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, \mathsf{Chal}, t)$:

1. $\mathcal{A}$ outputs a set $R \subseteq [N]$ (of red pebble indices) and $S \subseteq [N]$ (of black pebble indices).
2. The challenger samples $c_1, ..., c_\lambda \leftarrow \mathsf{Chal}(n)$. If $c_i \in R$ for some i then $\mathcal{A}$ immediately loses. The challenger additionally samples $v_1, ..., v_\kappa$ uniformly at random from indices in $V_C(n)$ and sends these to $\mathcal{A}$.
3. $\mathcal{A}$ plays the random (black) pebbling game on $\mathcal{G}(n)$ with the challenges $v_1, ..., v_\kappa$ and initial pebble configuration $P_0 = R \cup S$. It runs for t parallel rounds and outputs its final pebble configuration P_t. $\mathcal{A}$ wins if P_t contains pebbles on all of $v_1, ..., v_\kappa$.

Graph labeling PoS soundness Given the correspondence between the hardness of the random oracle labeling game and parallel black pebbling game, we can entirely capture the soundness of the graph labeling PoS in terms of the complexity of $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, t)$. Let $c : \mathbb{N} \to N$ denote a cost function $c : \mathbb{N} \to \mathbb{N}$ representing the parallel time cost (e.g. in sequential steps on a PRAM machine) of computing a label on a node of $\mathcal{G}(n)$ for each $n \in \mathbb{N}$.

Definition 30. *A graph labeling PoS with $\mathcal{G}(n), V_C(n), \mathsf{Chal}(n)$ and cost function $c(n)$ is parallel $(s, c(n) \cdot t, \mu)$-sound if and only if the probability that any $\mathcal{A}$ wins $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}, V_C, \mathsf{Chal}, t)$ is bounded by μ where $|S| = s$.*

5.3.3 Depth Robust Graphs

A directed acyclic graph (DAG) on n nodes with d-indegree is (n, α, β, d) *depth robust graph* (DRG) if every subgraph of αn nodes contains a path of length at least βn.

DRGs have been constructed for constant α, β and $d = O(\log n)$ using extreme constant-degree bipartite expander graphs, or local expanders [132, 118, 9]. Explicit constructions of local expanders exist [129], however they are complicated to implement and their concrete practicality is hindered by very large hidden constants. The most efficient way to instantiate these extreme expander graphs is probabilistically. A probabilistic DRG construction outputs a graph that is a DRG with overwhelming probability. The most efficient probabilistic construction to date is due to Alwen et. al. [8]. The analysis still leaves large gaps between security and efficiency although was shown to resist depth-reducing attacks empirically. Their construction is also *locally navigatable*, meaning that it comes with an efficient parent function to derive the parents of any node in the graph using polylogarithmic time and space.

5.3.4 Expander graphs

The vertex expansion of a graph $\mathcal{G}$ on vertex set V characterizes the size of the boundary of vertex subsets $S \subseteq V$ (i.e. the number of vertices in $V \setminus S$ that are neighbors with vertices in S).

Definition 31. *Let $\mathcal{G}$ be an undirected graph on a vertex set V of size $n \in \mathbb{N}$ and for any subset $S \subseteq V$ define $\Gamma(S)$ to be the set of vertices in $V \setminus S$ that have an edge to some vertex in S. For any constants $0 < \alpha < \beta < 1$, $\mathcal{G}$ is an (n, α, β) expander graph if and only if $|\Gamma(S) \cup S| \geq \beta n$ for all S of size $|S| \geq \alpha n$. For any $\delta > 0$, the set S is called $(1 + \delta)$-expanding if $|\Gamma(S) \cup S| \geq (1 + \delta)|S|$.*

In the case of directed bipartite graphs, vertex expansion is defined by the minimum number of sources connected to any given number of sinks.

Definition 32. *For any constants α, β where $0 < \alpha < \beta < 1$ and integer $n \in \mathbb{N}$, an (n, α, β) bipartite expander is a directed bipartite graph with n sources and n sinks such that any subset of αn sinks are connected to at least βn sources. For any $\delta > 0$, a subset S of sinks is called $(1 + \delta)$-expanding if it is connected to at least $(1 + \delta)|S|$ sources.*

It is easy to construct an undirected expander graph given a bipartite expander as defined above.

Fact 6. *Let $\mathcal{H}$ be a (n, α, β) bipartite expander graph with bounded degree d, sources and sinks labeled by indices in $[n]$ and edge set $E \subseteq [n] \times [n]$. Define the undirected graph $\mathcal{G}$ with vertices labeled in $[n]$ and edge set E' such that $(i, j) \in E'$ (for $i \neq j$) if and only if $(i, j) \in E$ or $(j, i) \in E$. Then $\mathcal{G}$ has bounded degree $2d$ and is an (n, α, β) expander.*

Chung's bipartite expander The randomized construction of Chung [66] defines the edges of a d-regular bipartite expander on $2n$ vertices by connecting the dn outgoing edges of the sources to the dn incoming edges of the sinks via a random permutation $\Pi : [d] \times [n] \to [d] \times [n]$. The ith source is connected to the jth sink if there is some $k_1, k_2 \in [d]$ such that $\Pi(k_1, i) = (k_2, j)$.

Lemma 27 (RD[141])**.** *The Chung random bipartite graph construction is a d-regular (n, α, β) expander with probability $1 - \mathsf{negl}(\lambda)\,(nH_b(\alpha))$ for all d, α, β satisfying:*

$$H_b(\alpha) + H_b(\beta) + d(\beta H_b(\alpha/\beta) - H_b(\alpha)) < 0 \tag{5.1}$$

where $H_b(x) = -x \log_2 x - (1-x)\log_2(1-x)$ is the binary entropy function.

For example, the above formula shows that for $\alpha = 1/2$ and $\beta = 0.80$ Chung's construction gives an $(n, 0.5, 0.80)$ expander for $d \geq 8$, meaning any subset of 50% of the sinks are connected to at least 80% of the sources when the degree is at least 8.

The following lemmas establish further properties of Chung's bipartite expander construction that will be used in the analysis of our PoS. The proofs are included in the full version of this paper. Let $\beta_G(\alpha)$ denote the smallest expansion of a subset of αn sources in a bipartite graph G, i.e. every subset of αn sources is connected to at least $\beta_G(\alpha)$ sinks.

Lemma 28. *For any $k > 1$ and $d > 2$, if the output of Chung's construction is a d-regular $(n, \alpha, k\alpha)$ bipartite expander for some $\alpha < \frac{d-k-1}{k(d-2)}$ with probability $1 - \mathsf{negl}(\lambda)\,(nH_b(\alpha))$ then $\beta_G(\alpha') \geq k\alpha'$ for every $\alpha' < \alpha$ with probability $1 - \mathsf{negl}(\lambda)\,(nH_b(\alpha'))$.*

Corollary 6. *For $d = 8$ Chung's construction is an 8-regular bipartite graph such that every subset of at most 1/3 of the nodes is 2-expanding, i.e. it is an $(n, \alpha, 2\alpha)$-bipartite expander for every $\alpha \leq 1/3$ with overwhelming probability.*

Proof. Plugging $\alpha = 1/3$ and $\beta = 2/3$ into the formula for degree (Equation 5.1) gives $d = 7.21 < 8$. With $d = 8$ and $k = 2$ the condition in Lemma 28 is satisfied: $\alpha = 1/3 < (d-k-1)/k(d-2) = 5/12$.

$\square$

For fixed d the expansion improves further as α decreases. Figure 5.2 provides a table of expansion factors over a range of α with fixed degree $d = 8$. Figure 5.3 plots the expansion as a function of subset size.

In a bipartite expanders, the "boundary" of a set of sources is the set of sinks connected to these sources that have distinct index labels from the sources, which is at least $\beta_G(\alpha) - \alpha$. Lemma 29 gives a smooth lower bound on $\beta_G(\alpha) - \alpha$ for Chung's bipartite expander graphs

Size (α)	0.01	0.10	0.20	0.30	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80
Expansion (β)	0.04	0.33	0.53	0.65	0.75	0.78	0.81	0.84	0.88	0.89	0.91	0.93	0.94
Factor (β/α)	4	3.3	2.65	2.1	1.8	1.73	1.62	1.53	1.47	1.37	1.3	1.24	1.17

Figure 5.2: A table of the maximum expansion (β) satisfying the condition from Lemma 27 for Chung's construction with fixed degree $d = 8$ over a range of subset sizes (α).

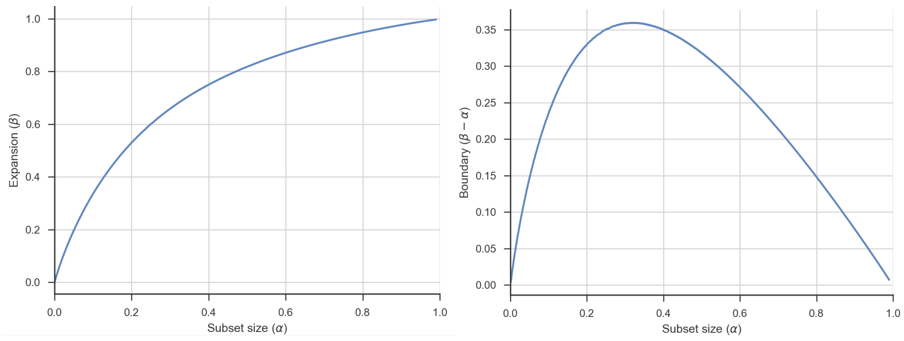

Figure 5.3: The graph on the left plots the lower bound from Lemma 27 on the expansion β as a function of the subset size α (in fractions of the sources/sinks) for Chung's construction with fixed $d = 8$. The graph on the right plots the corresponding lower bound on $\beta - \alpha$, which is the analog of the subgraph boundary in non-bipartite expanders. Specifically, this is a lower bound on the fraction of sinks connected to an α fraction of sources that have distinct index labels from the sources.

that we can show has a unique local maximum in $(0, 1)$.To simplify the analysis, we look at the function defined by the zeros of $\phi(\alpha, \beta) = d(\beta H_b(\alpha/\beta) - H_b(\alpha)) + 2 = 0$. Any α, β satisfying this relation also satisfies the relation in Lemma 27 because $H_b(\alpha) + H_b(\beta) < 2$ when $\beta > \alpha$. This implicitly defines β as a function of α, as well as the function $\hat{\beta} = \beta - \alpha$, by pairs of points $(\alpha, \hat{\beta}(\alpha))$ such that $\phi(\alpha, \alpha + \hat{\beta}(\alpha)) = 0$ is a lower bound to the boundary of subsets of size α, which holds at any point α with probability at least $1 - \mathsf{negl}(\lambda)\,(nH_b(\alpha))$.

Lemma 29. *Define $\phi(x, y) = d(yH_b(x/y) - H_b(x)) + c$ where c is any constant and let $\hat{\beta}$ be the function on $(0, 1)$ defined by pairs of points $(\alpha, \beta - \alpha)$ such that $\phi(\alpha, \beta) = 0$ and $0 < \alpha < \beta < 1$. The function $\hat{\beta}$ is continuously differentiable on $(0, 1)$ and has a unique local maximum.*

Corollary 7. *With overwhelming probability in n, Chung's construction (with $d = 8$) is an 8-regular bipartite graph on n sinks and n sources each indexed in $[n]$ such that for all $\alpha \in (0.10, 0.80)$ every αn sinks are connected to at least $0.12n$ sources with distinct indices.*

Lemma 30. *For any $d \geq 4$, Chung's construction yields a d-regular bipartite graph that is an $(n, \alpha, (d/3)\alpha)$ bipartite expander for every $\alpha \leq \frac{3}{2d}$ with probability $1 - \mathsf{negl}(\lambda)(nH_b(\alpha))$.*

Ramanujan graphs The explicit bipartite expander construction of Lubotzky, Phillips, and Sarnak [7] achieves similar expansion to Chung's construction for similarly good parameters. It is more complicated to implement as it involves generating a certain Cayley graph of the group $PGL(2, \mathbb{F}_q)$, the 2-dimensional projective general linear group on $\mathbb{F}_q$. This construction yields a bipartite exapnder of degree $(p+1)$ on $q(q^2-1)$ vertices for any primes p, q such that $p, q \equiv 1 \bmod 4$ and p is a quadratic non-residue modulo q. The resulting graph is called a *Ramanujan graph* because it is an optimal *spectral expander*, meaning the absolute value of all the non-trivial eigenvalues of its adjacency matrix (i.e. except the eigenvalue -d) are bounded by $2\sqrt{d-1}$. Other explicit Ramanujan graphs are known, for instance the isogeny graph of supersingular elliptic curves is also Ramanujan [136]. Due to a theorem of Tanner [156] relating spectral and vertex expansion, any d-regular bipartite Ramanujan graph on n vertices is an $(n, \alpha, \frac{d}{4(1-\alpha)+\alpha d})$ bipartite expander for all $\alpha < 1$. In particular, for $d \geq 4$ every fraction of $\alpha < 1/d$ sinks is at least $d/4$-expanding.

Lemma 31 (Tanner [156])**.** *For any $d \geq 4$ and $n \in \mathbb{N}$ a d-regular bipartite Ramanujan graph on n vertices is an $(n, \alpha, (d/4)\alpha)$ bipartite expander for every $\alpha < 1/d$.*

5.4 Tight PoS from stacked DRGs

In this section we show that stacking DRGs with bipartite expander edges between layers yields an arbitrarily tight proof of space with the number of layers increasing as $O(\log_2(1/\epsilon))$ where ϵ is the desired space gap. Moreover, the proof size is also $O(\log_2(1/\epsilon))$, which is asymptotically optimal. Our proofs attempt a tight analysis as well, e.g. showing that just 10 layers achieve a PoS with a 1% space gap, degree $8 + d$ graphs where d is the degree of the DRG, and relies only on a DRG that retains depth in 80% subgraphs.

5.4.1 Review of the PoS from stacked expanders

The PoS construction by Ren and Devadas [141] based on stacked bipartite expander graphs is a building block towards our tight PoS construction. Their construction uses a layered graph where each layer is a directed line on n nodes and the directed edges of a bipartite expander graph are placed between layers. This was shown to be an $(\epsilon\gamma n, (1-2\epsilon)\gamma n)$-sound PoS for parameters $\epsilon < 1/2$ and $\gamma < 1$ [141].

The graph $\mathcal{G}_{SE}$ The stacked-expander PoS uses the same underlying graph as the Balloon Hash memory hard function [99]. The graph $\mathcal{G}_{SE}$ consists of $\ell = O(\lambda)$ layers $V_1, ..., V_\ell$ consisting each of n vertices indexed in each level by the integers $[n]$, and where λ is a security parameter. First directed edges are placed from each kth vertex to the $k+1$st vertex in each level, i.e. forming a directed line. Next directed edges are placed from V_{i-1} to V_i according to the edges of an (n, α, β) bipartite expander on (V_{i-1}, V_i). Finally a "localization" operation is applied so that each kth vertex u_k in V_{i-1} is connected to the kth vertex v_k in V_i and any directed edge from the kth vertex of V_{i-1} to some jth vertex of V_i where $j > k$ is replaced with a directed edge from the kth vertex of V_i to the jth vertex of V_i. $\mathcal{G}_{SE}$ can be pebbled in $n\ell$ steps using a total of n pebbles.

Stacked expander PoS The PoS follows the generic PoS based on graph labeling. We remark only on several nuances. Due to the topology of $\mathcal{G}_{SE}$ after localization, the prover only needs to use a buffer of size n and deletes the labels of V_{i-1} as it derives the labels of V_i. After completing the labels C_i in the ith level it computes a vector commitment (e.g. Merkle commitment) to the labels in C_i denoted com_i. Once it has derived the labels C_ℓ of the final level V_ℓ it computes $com = H_{id}(com_1 || \cdots || com_\ell)$ and uses $H_{id}(com||j)$ to derive λ non-interactive challenges for each jth level.

5.4.2 PoS from stacking DRGs with expanders

By simply replacing each of the path graphs V_i in the stacked-expander PoS construction with a depth robust graph results in an arbitrarily tight PoS. Specifically, only $O(\log 1/\epsilon)$

layers are needed to achieve a $((1-\epsilon)n, \Omega(n))$-parallel-sound PoS. We demand only very basic properties from the DRG, e.g. that any subgraph on 80% of the nodes contains a long path of $\Omega(n)$ length. The rest of the PoS protocol remains the same.

Construction of $\mathcal{G}_{SDR}[\ell]$ The graph $\mathcal{G}_{SDR}[\ell]$ will be exactly like $\mathcal{G}_{SE}$ only each of the ℓ layers $V_1, ..., V_\ell$ contains a copy of an $(n, 0.80n, \beta n)$-depth-robust graph for some constant β. For concreteness, we define the directed edges between the layers using the degree 8 Chung random bipartite graph construction. Note that this graph is not "localized" and requires a buffer of size $2n$ rather than n. The PoS is still "tight" with respect to the persistent space storage.

The graph $\mathcal{G}_{SDR}[3]$ is illustrated in Figure 5.1.

Vector commitment storage If the vector commitment storage overhead required for the PoS is significant then this somewhat defeats the point of a tight PoS. Luckily this is not the case. Most vector commitment protocols, including the standard Merkle tree, offer smooth time/space tradeoffs. With a Merkle tree the honest prover can delete the hashes on nodes on the first k levels of the tree to save a factor 2^k space and re-derive all hashes along a Merkle path by reading at most 2^k nodes and computing at most 2^k hashes. If $k = 7$ this is less than a 1% overhead in space, and requires at most 128 additional hashes and reads. Furthermore, as remarked in [133] these 2^k reads are sequential memory reads, which in practice are inexpensive compared to the random reads for challenge labels.

Proof size We show that $\ell = O(\log(\frac{1}{3(\epsilon-2\delta)}))$ suffices to achieve $\mathsf{negl}(\lambda)\,(\lambda)$ soundness against any prover running in parallel time less than βn rounds of queries where Chal samples λ/δ nodes in each layer. This would result in a proof size of $O((1/\epsilon)\log(1/\epsilon))$, which is already a major improvement on any PoS involving a graph of degree $O(1/\epsilon)$ (recall that the only previously known tight PoS construction relied on very special DRGs whose degree must scale with $1/\epsilon$, which results in a total proof size of $O(1/\epsilon^2)$). However, we are able to improve the result even further and show that only $O(1/\delta)$ challenge queries are required overall, achieving proof complexity $O(1/\epsilon)$. This is the optimal proof complexity for the generic pebbling-based PoS with at most an ϵ space gap. If the prover claims to be storing n pebbles and the proof queries less than $1/\epsilon$ then a random deletion of an ϵ fraction of these pebbles evades detection with probability at least $(1-\epsilon)^{1/\epsilon} \approx 1/e$. The same applies if a random ϵ fraction of the pebbles the prover claims to be storing are red (i.e. errors).

Analysis outline We prove the hardness of the red-black pebbling game $\mathsf{Red\text{-}Black\text{-}Pebbles}^{\mathcal{A}}(\mathcal{G}_{SDR}[\ell], V_\ell, \mathsf{Chal})$ where Chal samples λ_i uniform challenges over V_i. We first show that it suffices to consider the parallel complexity of pebbling the set $U_\ell \subseteq V_\ell$ of *all* unpebbled nodes on V_ℓ from an initial configuration of γn black pebbles overall and $\delta_i n$ red pebbles in each layer where $\delta_\ell < \epsilon/2$.

As a shorthand notation, we will say that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard if every subset containing a μ fraction of the nodes in V_ℓ require t rounds to pebble (i.e. greater than a $1-\mu$ fraction of the nodes each individually require t rounds to pebble) from an initial configuration of γn black pebbles overall and $\delta_i n$ red pebbles in each layer where $\delta_\ell < \epsilon/2$. In Claim 14 we show that if $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard then the labeling PoS on $\mathcal{G}_{SDR}[\ell]$ is $(\gamma n, t, max\{p^*, \mu^\kappa\})$-sound where $p^* = max_i(1-\delta_i)^{\lambda_i}$. (Recall that κ and $\vec{\lambda}$ are parameters defined in the game).

For $\mu = 1$, $(\gamma, \vec{\delta}, t, 1)$-hardness is nearly equivalent to the standard parallel pebbling complexity of U_ℓ. The one distinction[5] is its dependency on the restriction to δ_i red pebbles in each layer, counted separately from black pebbles. In Claim 16 we show that if $\mathcal{G}_{SDR}[\ell]$ is $(1-\epsilon+2\delta_\ell, \vec{\delta}, t, 1)$-hard then $\mathcal{G}_{SDR}[\ell+1]$ is $(1-\epsilon, \vec{\delta}^*, t, 1-\epsilon/2)$-hard where $\vec{\delta}^*$ is equal to $\vec{\delta}$ on all common indices and $\delta_{\ell+1} = \delta_\ell$.

Finally, we analyze the complexity of pebbling all of U_ℓ, i.e. the $(\gamma, \vec{\delta}, t, 1)$-hardness of $\mathcal{G}_{SDR}[\ell]$. We show in Claim 18 that when the adversary uses at most $\gamma < 1-\epsilon$ black pebbles and δ red pebbles in each layer then pebbling all the unpebbled nodes in layer

[5]In prior uses of the red-black pebbling game to analyze proofs of space, it sufficed to consider parallel black pebbling complexity because replacing red pebbles with "free" black pebbles only increases the adversary's power. Our more refined analysis requires analyzing the weaker adversary who is restricted to a maximum number of red pebbles on each level of the graph, enforced by the construction.

V_ℓ (for ℓ dependent on ϵ and δ) requires pebbling $0.80n$ unpebbled nodes (including both red and black pebbles) in some layer V_i. Since the layer V_i contains a $(n, 0.80, \beta n)$-depth-robust graph, this takes at least βn rounds. We then generalize this analysis (Claim 19) to apply when δ_i is allowed to increase from level ℓ to 1 by a multiplicative factor such that $\sum_i \delta_i = O(\delta_\ell)$.

Theorem 23 ties everything together, taking into account the constraints of each claim to derive the PoS soundness of the labeling PoS on $\mathcal{G}_{SDR}[\ell]$.

Theorem 23. *The labeling PoS on $\mathcal{G}_{SDR}[\ell]$ with* Chal *sampling λ_i challenges in each level V_i and κ online challenges in V_ℓ is $((1-\epsilon-2\delta)n, \beta n, e^{-\lambda})$-sound with $\kappa = 2\lambda/\epsilon$ if either of the following conditions are met for $\epsilon - 2\delta \leq 0.24$:*

(a) $\ell = max(8, \log_2(\frac{1}{3(\epsilon-2\delta)}) + 4)$ *and each* $\lambda_i = \lambda/\delta$ *and* $\delta < min(0.01, \epsilon/3)$

(b) $\ell = max(14, \log_2(\frac{1}{3(\epsilon-3\delta)}) + 5)$ *and each* $\lambda_i = \lambda/\delta_i$ *where* $\delta_\ell = \delta < min(0.01, \epsilon/2)$ *and* $\delta_i = min(0.05, \frac{2}{3}\delta_{i-1})$

Proof. For any $\mathcal{G}_{SDR}[\ell]$, if the set of unpebbled nodes in V_ℓ are connected via unpebbled paths to at least $0.80n$ unpebbled nodes (including red and black) in some prior level V_i, then pebbling all of V_ℓ requires pebbling all these $0.80n$ unpebbled nodes, which requires βn rounds due to the fact that V_i is $(n, 0.80n, \beta n)$ depth robust. Claim 18 implies that $\mathcal{G}_{SDR}[\ell]$ is $(1-\epsilon, \vec{\delta}, \beta n, 1)$-hard for $\vec{\delta}$ and ℓ such that $\delta_i = \delta < \epsilon/2$ for all i and $\ell = max(7, \log_2(\frac{1}{3(\epsilon-2\delta)} + 3))$.

Claim 19 gives a different tradeoff between ℓ and $\vec{\delta}$, showing that the same hardness holds for $\vec{\delta}$ and ℓ such that $\delta_\ell = \delta < \epsilon/3$ and $\delta_i = min(0.05, \frac{2}{3}\delta_{i-1})$ and $\ell = max(13, \log_2(\frac{1}{3(\epsilon-3\delta)}) + 4)$.

Assuming $\epsilon - 2\delta \leq 0.24$, Claim 16 implies that $\mathcal{G}_{SDR}[\ell+1]$ is $(1-\epsilon-2\delta, \vec{\delta}, \beta n, 1-\epsilon/2)$-hard extending $\vec{\delta}$ so that $\delta_{\ell+1} = \delta_\ell = \delta$. Finally, by Claim 14, the labeling PoS on $\mathcal{G}_{SDR}[\ell+1]$ with challenge set V_ℓ and Chal sampling λ_i in each level V_i is $((1-\epsilon-2\delta)n, \beta n, max\{p^*, (1-\epsilon/2)^\kappa\})$-sound where $p^* = max_i(1-\delta_i)^{\lambda_i}$. Setting $\lambda_i = \lambda/\delta_i$ and $\kappa = 2\lambda/\epsilon$, the PoS is $((1-\epsilon-2\delta)n, \beta n, e^{-\lambda})$-sound.

□

Notation 1 (Common analysis notations)**.** *Let U_i denote the entire index set of nodes that are unpebbled in V_i and P_i the set that are pebbled. The total number of pebbles placed in the initial configuration is γn. Each level initially has $\rho_i n$ black pebbles and $\delta_i n$ red pebbles. Finally, $\gamma_i n = \sum_{j<i} \rho_i n$ is the number of black pebbles placed before level i.*

Claim 14. *If $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard then the labeling PoS on $\mathcal{G}_{SDR}[\ell]$ is $(\gamma n, t, max\{p^*, \mu^\kappa\})$-sound where $p^* = max_i(1-\delta_i)^{\lambda_i}$.*

Proof. Fix $\gamma = 1 - \epsilon$ and δ_i for each i. The λ_i challenges during Initialization in each level ensure that $\mathcal{A}$ wins with at most probability $(1-\delta_i)^{\lambda_i}$ if it places more than δ_i red pebbles on V_i. If $\mathcal{A}$ has exceeded the δ_i bound in more than one level this only increases its probability of failure. Thus, in case 1 ($\mathcal{A}$ places more than δ_i red pebbles on some level i), $\mathcal{A}$'s success probability is bounded by maximum value of $(1-\delta_i)^{\lambda_i}$ over all i. In case 2 ($\mathcal{A}$ places fewer than δ_i on each ith level), the fact that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, \mu)$-hard implies that at most a μ fraction of the nodes on V_ℓ can individually be pebbled from the starting configuration in t rounds, hence $\mathcal{A}$'s success probability of answering κ independent challenges is bounded by μ^κ. The success probability is bounded by the maximum of these two cases. □

Claim 15 (trivial)**.** *$\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, 1)$-hard if and only if given any initial configuration P_0 of $\gamma_\ell n$ black pebbles placed on layers $V_1, ..., V_{\ell-1}$ at most $\delta_i n$ red pebbles in each layer, and any set $U \subseteq V_\ell$ of αn unpebbled nodes in V_ℓ such that $\alpha - \gamma_\ell \geq 1 - \gamma - \delta$, no adversary can pebble U in fewer than t rounds.*

Proof. Any placement of $\delta_i n$ red pebbles on each level and γn black pebbles overall with γ_ℓ pebbles on $V_1, ..., V_{\ell-1}$ has $\rho_\ell n = (\gamma - \gamma_\ell)n$ black pebbles on V_ℓ and exactly $(1 - \rho_\ell - \delta_\ell)n$ unpebbled nodes in V_ℓ (assuming without loss of generality that red and black pebbles never share the same node). If for any γ_ℓ pebbles placed on $V_1, ..., V_{\ell-1}$ and any set U of $(1 - \rho_\ell - \delta_\ell)n$ of unpebbled nodes in V_ℓ no adversary can pebble U in t or fewer rounds, then no adversary can pebble the set of unpebbled nodes in V_ℓ (i.e. because U is precisely this set).

Conversely, suppose that $\mathcal{G}_{SDR}[\ell]$ is $(\gamma, \vec{\delta}, t, 1)$-hard, i.e. pebbling the entire set of unpebbled nodes in V_ℓ requires t rounds if at most γn black pebbles are placed overall with at most $\delta_i n$ red pebbles in each level. Then for any placement of γ_ℓ pebbles on $V_1, ..., V_{\ell-1}$ with at most $\delta_i n$ red in each level, and for any $U \subseteq V_\ell$ of size $u \geq (1 - \rho_\ell - \delta_\ell)n$, the remaining nodes in $V_\ell \setminus U$ can be pebbled with just $(\gamma - \gamma_\ell + \delta_\ell)n$ pebbles. This means there exists a pebbling of the entire graph with γn pebbles overall and the same configuration on the first $\ell - 1$ levels that leaves only the set U unpebbled in V_ℓ. Thus, by hypothesis, this unpebbled set U requires more than t rounds to pebble. Any other initial pebbling configuration that has fewer pebbles on $V_\ell \setminus U$ would only make it harder to pebble U. □

Claim 16. *For any $\epsilon - 2\delta \leq 0.24$, if $\mathcal{G}_{SDR}[\ell - 1]$ is $(1 - \epsilon + 2\delta_{\ell-1}, \vec{\delta}, t - 1, 1)$-hard then $\mathcal{G}_{SDR}[\ell]$ is $(1 - \epsilon, \vec{\delta}^*, min(\beta n, t), 1 - \epsilon/2)$-hard where $\vec{\delta}^*$ is identical to $\vec{v}$ on all common indices and $\delta_\ell = \delta_{\ell-1} \leq \epsilon/2$.*

Proof. Refer to Notation 1. Consider the graph $\mathcal{G}_{SDR}[\ell]$ with $\gamma n = (1 - \epsilon)n$ black pebbles initially placed. Let $\delta = \delta_\ell = \delta_{\ell-1} \leq \epsilon/2$. Let $\alpha_\ell = |U_\ell|/|V_\ell|$ denote the fraction of nodes in V_ℓ that are unpebbled. Every $1 - \epsilon/2$ fraction of V_ℓ contains at least $\alpha^* n = (\alpha_\ell - \epsilon/2)n$ unpebbled nodes. If $\alpha^* \geq 0.80$, then every $1 - \epsilon/2$ fraction of V_ℓ contains a path of length βn because V_ℓ is a $(n, 0.80n, \beta n)$-depth robust graph. We consider the two other cases next:

Case $\alpha^* < 1/3$: The $\alpha^* n$ unpebbled nodes have dependencies on at least a $2\alpha^* = 2\alpha_\ell - \epsilon$ fraction of nodes in $V_{\ell-1}$ (Corollary 6, bipartite expansion). These contain at least $\alpha' n = (2\alpha_\ell - \epsilon - \rho_{\ell-1} - \delta)n$ unpebbled nodes because in the worst case they include $\rho_{\ell-1} n$ black pebbles and δn red pebbles. There are $\gamma_{\ell-1} n = (\gamma - \rho_{\ell-1} - \rho_\ell)n$ pebbles placed on all prior levels. By definition $\rho_\ell = 1 - \alpha_\ell - \delta$ and $\gamma = 1 - \epsilon$. Substituting $\alpha_\ell \geq 1 - \gamma - \delta$ shows that $\alpha' - \gamma_{\ell-1} = 2\alpha_\ell - \gamma + \rho_\ell - \epsilon - \delta = \alpha_\ell + 1 - \gamma - \epsilon - 2\delta \geq 1 - 2\gamma - 3\delta + 1 - \epsilon = 1 - \gamma - 3\delta$. Setting $\gamma' = 1 - \epsilon + 2\delta = \gamma + 2\delta$ gives the relation $\alpha' - \gamma_{\ell-1} \geq 1 - \gamma' - \delta$. It then follows from Claim 15 that if $\mathcal{G}_{SDR}[\ell - 1]$ is $(\gamma', \vec{\delta}, t - 1, 1)$-hard then the $\alpha' n$ unpebbled nodes in $V_{\ell-1}$ require $t - 1$ rounds to pebbled. Thus, the $\alpha^* n$ unpebbled nodes in V_ℓ require t rounds to pebble.

Case $\alpha^* \geq 1/3$: In this case $\alpha^* \in (0.33, 0.80)$. It is connected to $\beta^* n$ nodes in $V_{\ell-1}$. Among these at least $\alpha' n$ for $\alpha' \geq \beta^* - \rho_{\ell-1} - \delta$ are unpebbled. Since $\gamma_{\ell-1} = \gamma - \rho_\ell - \rho_{\ell-1}$ we get $\alpha' - \gamma_{\ell-1} \geq \beta^* - \gamma + \rho_\ell - \delta$. According to Corollary 7 on the bipartite expander boundary $\beta^* - \alpha^* \geq 0.12$. Therefore, $\rho_\ell = 1 - \alpha_\ell - \delta \geq 1 - \alpha^* - \epsilon/2 - \delta$ so $\alpha' - \gamma_{\ell-1} \geq \beta^* - \gamma + 1 - \alpha^* - \epsilon/2 - 2\delta = \beta^* - \alpha^* + \epsilon/2 - 2\delta \geq 0.12 + \epsilon/2 - 2\delta$. If $\mathcal{G}_{SDR}[\ell - 1]$ is $(1 - \epsilon + 2\delta, \vec{\delta}, t - 1, 1)$-hard, then by Claim 15 the $\alpha' n$ unpebbled nodes require $t - 1$ rounds as long as $0.12 + \epsilon/2 - 2\delta \geq \epsilon - 3\delta$, which is true for $\epsilon - 2\delta \leq 0.24$.

□

Claim 17. *If $\mathcal{G}_{SDR}$ initially has at most γn black pebbles for $\gamma \leq 1 - \epsilon$ and at most $\delta n < \epsilon n/2$ red pebbles in each layer then for $\ell = \log_2(\frac{1}{3(\epsilon - 2\delta)})$ the unpebbled nodes in V_ℓ have unpebbled paths from at least $n/3$ unpebbled nodes in some layer V_i.*

Proof. Refer to Notations 1. Let $\alpha_i n$ denote the number of unpebbled dependencies of U_ℓ in V_i, i.e. the number of nodes in U_i that have unpebbled paths to U_ℓ. Suppose that α_i is bounded by 1/3 for all levels up to $\ell - k$, i.e. $\alpha_\ell < 1/3, ..., \alpha_{\ell-k} < 1/3$. We will prove the following bound:

$$\alpha_{\ell-k} \geq 2^k(\alpha_\ell - \gamma_\ell/2 - \delta) \geq 2^{k-1}(\alpha_\ell + \epsilon - 3\delta) \geq 2^k(\epsilon - 2\delta) \tag{5.2}$$

Before proving this bound let us note its implication. For $k = \log_2(\frac{1}{3(\epsilon-2\delta)})$ this implies $\alpha_{\ell-k} \geq 1/3$, which contradicts $\alpha_{\ell-k} < 1/3$. Therefore, it follows that $\alpha_i \geq 1/3$ at some index $i > \ell - \log_2(\frac{1}{3(\epsilon-2\delta)})$, which leads to the conclusion that for $\ell \geq \log_2(\frac{1}{3(\epsilon-2\delta)})$ there exists some level V_i with at least $n/3$ unpebbled nodes that have unpebbled dependency paths to the set X_ℓ of unpebbled nodes in V_ℓ.

Let $j = \ell - i$. From Corollary 6 (bipartite expansion), if $\alpha_j \leq 1/3$ then X_j is connected to at least $2\alpha_j$ nodes in V_{j-1}. At most $(\rho_{j-1} + \delta)n$ of these are pebbled. Therefore, $\alpha_{j-1} \geq 2\alpha_j - \rho_{j-1} - \delta$. Now we show by induction that $\alpha_{\ell-k} \geq 2^k\alpha_\ell - 2^{k-1}\rho_{\ell-1} - (2^k - 1)\delta$.

The base case $k = 0$ is trivial. Assuming this holds for k:

$$\begin{aligned}\alpha_{\ell-k-1} &\geq 2\alpha_{\ell-k} - \rho_{\ell-k-1} - \delta \geq 2(2^k\alpha_\ell - 2^{k-1}\rho_{\ell-1} - (2^k-1)\delta) - \rho_{\ell-k-1} - \delta \\ &\geq 2^{k+1}\alpha_\ell - 2^k\rho_{\ell-1} - (2^{k+1}-1)\delta\end{aligned}$$

The last inequality used the fact that $\sum_{i=1}^{k}\rho_{\ell-i} \leq \gamma_\ell$ and therefore $\sum_{i=1}^{k} 2^{k-i}\rho_{\ell-i}$ is maximized by setting $\rho_{\ell-1} = \gamma_\ell$ and $\rho_{\ell-i} = 0$ for all $i > 1$.

From the identities $\gamma_\ell = \gamma - \rho_\ell$ and $\alpha_\ell = 1 - \rho_\ell - \delta$ we derive $\gamma_\ell = \gamma + \alpha_\ell - 1 + \delta \leq \alpha_\ell + \delta - \epsilon$. Finally, inserting this into the bound above and using the fact that $\alpha_\ell \geq \epsilon - \delta$ gives:

$$\alpha_{\ell-k} \geq 2^{k-1}(2\alpha_\ell - \gamma_\ell - 2\delta) \geq 2^{k-1}(\alpha_\ell + \epsilon - 3\delta) \geq 2^k(\epsilon - 2\delta)$$

□

We could stop here as we have already shown unpebbled dependency paths from the unpebbled sinks in V_ℓ to a $1/3$ fraction of nodes in some level for $\ell = O(\log(1/(\epsilon - 2\delta))$ and the remainder of our PoS analysis could rely on a graph that is $(n, 0.33n, \Omega(n))$-depth-robust. However, we can tighten the analysis further so that we only need to assume the graph is $(n, 0.80, \Omega(n))$-depth robust.

Claim 18. *If $\mathcal{G}_{SDR}$ initially has at most γn black pebbles for $\gamma \leq 1 - \epsilon$ and at most $\delta < \epsilon/2$ red pebbles in each layer then the unpebbled nodes in V_ℓ have unpebbled paths to at least $0.80n$ unpebbled nodes in some layer V_i for $\ell = max(\frac{0.68-\epsilon+\delta}{0.12-\delta}, \log_2(\frac{1}{3(\epsilon-2\delta)})) + 3)$. In particular, $\ell = max(7, \log_2(\frac{1}{3(\epsilon-2\delta)})) + 3)$ when $\delta \leq 0.01$.*

Proof. In Claim 17 we showed that for $\ell \geq \log_2(\frac{2}{3(\alpha_\ell+\epsilon-3\delta)})$ there exists an index i where $\alpha_i \geq 1/3$ and $\alpha_\ell + \epsilon - 3\delta \geq 2\epsilon - 4\delta$ (Equation 5.2). Picking up from here, we consider what happens once $\alpha_i \geq 1/3$. We break the analysis into two cases: in the first case $\alpha_\ell < 1/3$ and in the second case $\alpha_\ell \geq 1/3$.

In both cases we will use a different bound on α_{i-k} because once $\alpha_i > 1/3$ the unpebbled sets may not be 2-expanding. Define the function $\beta(\alpha)$ to be the minimum bipartite expansion of a set of fractional size α, i.e. every set of αn nodes is connected to at least $\beta(\alpha)n$ nodes in the previous level. Let $\hat{\beta}(\alpha) = \beta(\alpha) - \alpha$. Using the relation $\alpha_{i-1} \geq \beta(\alpha_i) - \rho_{i-1} - \delta$ we derive that $\alpha_{i-2} \geq \hat{\beta}(\alpha_{i-1}) + \beta(\alpha_i) - \rho_{i-1} - \rho_{i-2} - 2\delta$ and more generally, since $\sum_{j=1}^{k} \rho_{i-j} \leq \gamma_i$:

$$\begin{aligned}\alpha_{i-k} &\geq \sum_{j=1}^{k-1} \hat{\beta}(\alpha_{i-j}) + \beta(\alpha_i) - k\delta - \sum_{j=1}^{k} \rho_{i-j} \\ &\geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - \delta) + \beta(\alpha_i) - \gamma_i - \delta\end{aligned}$$

The final ingredient is the bound $\gamma_i \leq \alpha_i - \epsilon + \delta$ for all i. To see this, first observe that $\alpha_\ell - \gamma_\ell = 1 - \rho_\ell - \delta - (\gamma - \rho_\ell) = \epsilon - \delta$. If $\alpha_i - \gamma_i \geq \epsilon - \delta$ and $\epsilon > 2\delta$, then $0.80 > \alpha_i > \delta$, and so the $\alpha_i n$ dependencies are connected to $\beta(\alpha_i)n > (\alpha_i + \delta)n$ nodes in level V_{i-1}. Therefore $\alpha_{i-1} \geq \alpha_i - \rho_{i-1} = \alpha_i - (\gamma_i - \gamma_{i-1})$. In words, decreasing the number of dependencies requires using black pebbles 1-to-1, so $\alpha_{i-1} - \gamma_{i-1} \geq \alpha_i - \gamma_i$.

$$\alpha_{i-k} \geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - \delta) + \hat{\beta}(\alpha_i) + \epsilon - 2\delta \tag{5.3}$$

By Corollary 7 to Lemma 29, $\hat{\beta}(\alpha) \geq 0.12$ for $\alpha \in (0.10, 0.80)$.

Case $\alpha_\ell \geq 1/3$*:* First we claim that $\alpha_{\ell-i} \geq 0.12$ for all i. If $\alpha_i \geq 0.12$ then as shown above $\alpha_{i-1} \geq \hat{\beta}(\alpha_i) + \epsilon - 2\delta \geq 0.12$ because $\hat{\beta}(\alpha) \geq 0.12$ for all $\alpha \in (0.10, 0.80)$ and $\epsilon > 2\delta$. Our claim thus follows by induction. Therefore, for all $j \leq k$ we derive that $min_{j\leq k}(\hat{\beta}(\alpha_{\ell-j})) \geq 0.12$. Equation 5.3 then shows that $\alpha_{\ell-k-1} \geq k(0.12 - \delta) + 0.12 + \epsilon - \delta$, or $\alpha_{\ell-k-1} \geq 0.80$ at $k \geq (0.68 - \epsilon + \delta)/(0.12 - \delta)$ (e.g. $k = 7$ when $\delta \leq 0.01$).

Case $\alpha_\ell < 1/3$*:* From Equation 5.2, $\alpha_i \geq 1/3$ at some index $i \geq \ell - k$ for $k = \log_2(\frac{2}{3(\alpha_\ell + \epsilon - 3\delta)})$. At this point $\alpha_i \geq 1/3$ and $\gamma_i < \gamma_\ell \leq \alpha_\ell - \epsilon + \delta$. Combining this with Equation 5.3, we can apply the same analysis as in the previous case to first show by induction that $\alpha_{i-k'} \geq 0.12$ for all k' and then more generally: $\alpha_{i-k'} \geq (k'-1)(0.12 - \delta) + \beta(\alpha_i) - \alpha_\ell + \epsilon - 2\delta \geq$

$k'(0.12-\delta)+0.68-\alpha_\ell+\epsilon-2\delta$. We used the fact that $\beta(\alpha_i)\geq\beta(0.33)\geq 0.68$. Therefore, $\alpha_{i-k'-1}\geq 0.80$ when $k'\geq(0.80-0.68+\alpha_\ell)/(0.12-\delta)$. This shows that the total number of levels where $\alpha_i<0.80$ is at most:

$$\ell=k+k'+1\leq 1+\log_2\big(\frac{2}{3(\alpha_\ell+\epsilon-3\delta)}\big)+\frac{0.12+\alpha_\ell}{0.12-\delta}$$

The derivative of this expression with respect to α_ℓ is $\frac{1}{0.12-\delta}-\frac{1}{\ln(2)(\alpha_\ell+\epsilon-3\delta)}$, which is initially decreasing when $\ln(2)(\alpha_\ell+\epsilon-3\delta)<0.12-\delta$ and then increasing for larger α_ℓ. Therefore, the maxima are on the endpoints of the interval $\alpha_\ell\in(\epsilon-\delta,0.33)$. We already considered the case $\alpha_\ell=0.33$. When $\alpha_\ell=\epsilon-\delta$ then the number of levels is at most $1-\log_2(3(\epsilon-2\delta))+\frac{0.12}{0.12-\delta}$.

In conclusion, the total number of levels before $\alpha_i\geq 0.80$ is at most:

$$\ell\leq max\big((0.68-\epsilon+\delta)/(0.12-\delta),1-\log_2\big(3(\epsilon-2\delta)\big)+1/(1-\delta/12)\big)$$

In particular, when $\delta\leq 0.01$ this becomes $max(7,-\log_2(3(\epsilon-2\delta))+3)$. □

Relaxing δ Claim 18 improved on Claim 17 to show unpebbled dependency paths to 80% of the subgraph in some layer. The final improvement is to redistribute the δ_i such that $\sum_i\delta_i=O(\delta)$ but security is still maintained. Intuitively, ensuring $\delta<\epsilon$ is necessary on level V_ℓ as otherwise $\gamma+\delta\geq 1$ and there are no unpebbled nodes on level V_ℓ (all the missing black pebbles can be covered with red pebbles). However, as the dependencies expand between levels a larger δ can be tolerated as well. Although the number of black pebbles the prover will place on each level isn't fixed a priori, we show that if $\delta<\epsilon/2$ in level V_ℓ then we can tolerate a factor 3/2 increase between levels as long as $\delta\leq 0.05$ in any layer.

That is, if δ_i denotes the bound on the number of red pebbles in the ith layer then our new analysis requires $\delta_\ell<\epsilon/2$ and $\delta_i=min(0.05,(3/2)\delta_{i+1})$. This means that the total number of queries in the PoS over all levels is $O(1/\epsilon)$ because $\sum_{i=1}^{\ell}1/\delta_i\leq max(0.10\ell,\frac{3}{2\delta_\ell})$.

Claim 19. *For any $\gamma\leq 1-\epsilon$ and $\delta<\epsilon/3$, if $\mathcal{G}_{SDR}$ initially has at most γn black pebbles, $\delta_\ell=\delta$ red pebbles in layer V_ℓ, and $\delta_i=min(0.05,(2/3)\delta_{i-1})$ red pebbles in layer V_i, then* *for $\ell = max(13, \log_2(\frac{1}{3(\epsilon - 3\delta)})) + 4)$ the unpebbled nodes in V_ℓ have unpebbled paths to at least $0.80n$ unpebbled nodes in some layer V_i.*

Proof. Modifying Equation 5.2 to account for the different values of δ_i gives:

$$\begin{aligned}\alpha_{\ell-k} &\geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - (2^{k-1}\delta_{\ell-1} + 2^{k-2}\delta_{\ell-2} + \cdots + \delta_{\ell-k}) \\ &\geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - \sum_{i=1}^{k} 2^{k-i}(3/2)^{i-1}\delta_\ell\end{aligned}$$

Let $\sigma_k = \sum_{i=1}^{k} 2^{k-i}(3/2)^{i-1}$. Then $(4/3)\sigma_k = \sigma_k + 2^{k+1}/3 - (3/2)^{k-1}$. Therefore $\sigma_k = 2^{k+1} - 3^k/2^{k-1} < 2^{k+1}$. Using $\gamma_\ell \leq \alpha_\ell + \delta_\ell - \epsilon$ and $\alpha_\ell \geq \epsilon - \delta_\ell$ we derive the new bound:

$$\alpha_{\ell-k} \geq 2^k\alpha_\ell - 2^{k-1}\gamma_\ell - 2^{k+1}\delta_\ell \geq 2^{k-1}(\alpha_\ell + \epsilon - 5\delta_\ell) \geq 2^k(\epsilon - 3\delta) \tag{5.4}$$

This shows that if $\ell \geq \log_2(\frac{1}{3(\epsilon-3\delta)})$ then there is some level V_i where $\alpha_i \geq 1/3$.

We must also modify Equation 5.3 using $\sum_{j=1}^{k} \rho_{i-j} \leq \gamma_i \leq \alpha_i - \epsilon + \delta_i$:

$$\alpha_{i-k} \geq (k-1)min_{j<k}\hat{\beta}(\alpha_{i-j}) - \sum_{j=0}^{k} \delta_{i-j} + \hat{\beta}(\alpha_i) + \epsilon \tag{5.5}$$

When $i = \ell$ and k is small $\delta_{\ell-k} = (3/2)^k\delta_\ell$ and $\delta_\ell \leq \epsilon/3$ implies:

$$\alpha_{\ell-k} \geq (k-1)min_{j<k}\hat{\beta}(\alpha_{i-j}) + \hat{\beta}(\alpha_\ell) + (\frac{2}{3} - \frac{3^k}{2^{k+1}})\epsilon$$

Otherwise, we can use $\delta_i \leq 0.05$.

$$\alpha_{i-k} \geq (k-1)(min_{j<k}\hat{\beta}(\alpha_{i-j}) - 0.05) + \hat{\beta}(\alpha_i) + \epsilon - 0.10$$

Now we turn back to the two cases for $\alpha_i \geq 1/3$.

Case $\alpha_\ell \geq 1/3$: We claim that $\alpha_{\ell-k} \geq 0.11$ for all k. This is true for α_ℓ by hypothesis. From the equation above and the bound $\hat{\beta}(\alpha) \geq 0.12$ for all $\alpha \in (0.10, 0.80)$ (Corollary 7),

$\alpha_{\ell-1} \geq \hat{\beta}(\alpha_\ell) - \epsilon/12 > 0.11$. Therefore, $\alpha_{\ell-2} \geq (0.12 - 0.05) + 0.12 - (11/24)\epsilon \geq 0.18$. Now assume that $\alpha_{\ell-2-j} \geq 0.12$ for all $j < k$, then $\alpha_{\ell-2-j} \geq (k-1)0.07 + 0.12 - (11/24)\epsilon > 0.11$. The claim follows by induction. This also shows that $\alpha_{\ell-k} \geq (k-3)0.07 + 0.11 > 0.80$ when $k = 13$.

Case $\alpha_\ell < 1/3$: From Equation 5.4, $\alpha_i \geq 1/3$ at some index $i \geq \ell - k$ for $k = \log_2(\frac{2}{3(\alpha_\ell + \epsilon - 5\delta_\ell)})$. At this point $\alpha_i \geq 1/3$ and $\gamma_i < \gamma_\ell \leq \alpha_\ell - \epsilon + \delta_\ell$. Combining this with Equation 5.5 gives:

$$\alpha_{i-k'} \geq (k'-1)(min_{j<k'}\hat{\beta}(\alpha_{i-j}) - 0.05) + \beta(\alpha_i) - \alpha_\ell + \epsilon - 0.05 - \delta_\ell$$

We claim that $\alpha_{i-k'} \geq 0.30$ for all k'. Observe that $\alpha i - 1 \geq \beta(\alpha_i) - \alpha_\ell + \epsilon - 0.05 - \delta_\ell \geq 0.68 - 0.38 + \epsilon - \delta_e ll \geq 0.30$ for any value $\alpha_\ell < 0.33$ because $\beta(\alpha_i) \geq \beta(0.33) \geq 0.68$. Assuming this is true for all α_{i-j} where $1 < j \leq k'$ implies $\alpha_{i-k'} \geq (k'-1)0.07 + 0.30 \geq 30$. Therefore, we can state more generally that $\alpha_{i-k'} \geq (k'-1)0.07 + 0.68 - \alpha_\ell$ and $\alpha_{i-k'-1} \geq 0.80$ when $k' = (0.12 + \alpha_\ell)/0.07$. The total number of levels where $\alpha_i < 0.80$ is thus at most:

$$k + k' + 1 \leq 1 - \log_2((3/2)(\alpha_\ell + \epsilon - 5\delta_\ell)) + 2 + \alpha_\ell/0.07$$

Differentiating this expression with respect to α_ℓ shows that the maxima over $\alpha_\ell \in (\epsilon - \delta_\ell, 0.33)$ are on the endpoints. The endpoint $\alpha_\ell = 0.33$ coincides with the case above. At the endpoint $\epsilon - \delta_\ell$ the number of levels is bounded by $3 - \log_2(3(\alpha_\ell + \epsilon - 5\delta)) + \epsilon/0.07$.

In conclusion, considering both cases, the total number of levels before $\alpha_i \geq 0.80$ is at most:

$$\ell \geq max(13, 3 - \log_2(3(\epsilon - 3\delta_\ell)) + \epsilon/0.07)$$

In particular, when $\epsilon \leq 0.07$ and $\delta_\ell = \delta$ then $\ell \leq max(12, 4 - \log_2(3(\epsilon - 3\delta))$.

□

5.5 PoRep from Stacked DRGs

5.5.1 PoRep definitions

1. $\mathsf{Setup}(\lambda) \rightarrow pp$. The setup runs on security parameters $\lambda \in \mathbb{N}$ and outputs public parameters pp for the scheme. The public parameters are implicit inputs to remaining algorithms.

2. $\mathsf{Replicate}(id, D) \rightarrow (\Phi, R)$. The replicate procedure take a data input D, a string identifier id, and outputs the replica R and a compact string Φ. Both Φ and id are bounded in length by $O(polylog(n))$ where $n = |R|$.

3. $\mathsf{ProveReplica}(c, id, \Phi, R) \rightarrow \pi$. This is the first proof generated after replication and shows that the replication procedure was done correctly. Both the input challenge c and proof π are bounded in size as $O(\lambda \cdot polylog(n))$.

4. $\mathsf{VerifyReplica}(c, id, \Phi, \pi) \rightarrow b$. This verifies the proof generated by $\mathsf{ProveReplica}$ against (id, Φ) and outputs a decision bit $b \in \{0, 1\}$.

5. $\mathsf{Extract}(id, R) \rightarrow D$. The extraction procedure is able to recover the data input D from the replica R and the identifier id.

6. $\mathsf{ProveStorage}(c', \Phi, R) \rightarrow \pi'$. This is the repeated "online" proof that shows the prover is still storing R. The input challenge c' and output proof π' are both size $O(\lambda \cdot polylog(n))$.

7. $\mathsf{VerifyStorage}(c', \Phi) \rightarrow b'$. This verifies the proof generated by $\mathsf{ProveStorage}$ and outputs a decision bit $b' \in \{0, 1\}$. It only takes Φ and the challenge c' as input.

The PoRep protocol, similar to PoS, consists of an initialization interactive protocol and repeated execution interactive protocol.

1. $\mathsf{Initialization}$. The prover chooses a data input $D \in \{0, 1\}^n$, a unique identifier id and runs $\mathsf{Replicate}(id, D)$, storing the output Φ, R. The prover gives Φ to the verifier,

who responds with a challenge c. The prover runs $\mathsf{ProveReplica}(c, id, \Phi, R)$ and sends the output π to the verifier, who checks that $\mathsf{VerifyReplica}(c, id, \Phi, \pi)$ returns 1. The verifier aborts if this check fails, otherwise the prover passes.

2. Execution. On a challenge c' from the verifier the prover runs $\mathsf{ProveStorage}(c', \Phi, R)$ and returns the output π' to the verifier. The verifier checks that $\mathsf{VerifyStorage}(c', \Phi)$ returns 1. The verifier aborts if this check fails, otherwise the prover passes.

Correctness In a correct construction, if $\mathsf{Replicate}(id, D) \to (\Phi, R)$ then $\mathsf{Extract}(id, R) \to D$ with probability 1 for any $D \in \{0,1\}^n$. Furthermore, the honest prover passes with probability 1 both Initialization and Execution. The runtime of Extract is $O(poly(\lambda, n))$. Faster extraction times are useful in practice, but an extraction time that is faster (or more parallelizable) than the prover's runtime in Initialization has implications for PoS composition.

5.5.2 Stacked DRGs PoReP

Setup and Replicate The setup is the same as in the Stacked DRGs PoS protocol. $\mathsf{Replicate}(id, D)$ first runs the Initialization labeling of the Stacked DRGs graph with layers of $N = \lceil |D|/\omega \rceil$ using a hash function salted with both id and τ_D, where τ_D is a Merkle commitment to the data D. The output output length of the hash is ω. Let S be the concatenation of all labels on the final level of the graph that the PoS prover would store. The replica output is $R = D \oplus S$. Φ consists of two commitments: a Merkle commitment τ_D to the data input D and a Merkle commitment Λ to a list consisting of all the labels on the graph in topological order followed by the ω size blocks of R in order.

Prove and Verify The challenge c to $\mathsf{ProveReplica}(c, id, \Phi, R)$ and its output proof π are nearly the same as in the Initialization of Stacked DRGs. The challenge is a subset of the nodes in each level and the proof contains Merkle openings of the labels on all challenged nodes and labels of their parent nodes. There is one modification. For each node challenged in the last level of the graph the prover also provides the corresponding blocks of R and D

along with Merkle inclusion proofs of each in Λ and τ_D respectively. (Say the ith node of V_ℓ is challenged, then π contains Merkle openings of r_i, the ith block of R, and d_i, the ith block of D). $\mathsf{VerifyReplica}(c, id, \Phi, \pi)$ is also mostly the same. The only difference pertains to verifying the blocks of R and D against the corresponding labels in V_ℓ. Given a label ℓ_i on the ith node of V_ℓ along with r_i (the ith block of R) and d_i (the ith block of D), the verification checks that $\ell_i \oplus d_i = r_i$. The Merkle inclusion proofs of r_i and d_i are also verified against Λ and τ_D.

$\mathsf{ProveStorage}(c', \Phi, R)$ and $\mathsf{VerifyStorage}(c', \Phi)$ are the same as the challenge-response protocol in the $\mathsf{Execution}$ of the regular $\mathsf{Stacked\ DRGs}$ PoS. The prover simply uses R instead of the labels S and provides Merkle proofs that are verified against Λ. (This is just a proof of retrievability of R).

Extract $\mathsf{Extract}(id, R)$ reruns $\mathsf{Replicate}(id, D)$ up until the point it obtains the labels S, and then recovers $D = R \oplus S$.

5.6 Lemmas on Random Bipartite Expanders

This section contains proofs of select lemmas from Section 5.3.4.

5.6.1 Proof of Lemma 28

For fixed d and k define $\phi(\alpha) = H_b(\alpha) + H_b(k\alpha) + dk\alpha H_b(1/k) - dH_b(\alpha)$. By Lemma 27, Chung's construction outputs a graph that is a d-regular $(n, \alpha, k\alpha)$ bipartite expander with probability $1 - \mathsf{negl}(\lambda)\,(nH_b(\alpha))$ as long as $\phi(\alpha) < 0$. We will show that if $\phi(\alpha) < 0$ for some α within the domain $X = (0, \frac{d-k-1}{k(d-2)})$, then $\phi(\alpha') < 0$ for all $\alpha' < \alpha$. We will first prove two subclaims:

1. ϕ is smooth on $\alpha \in (0, 1/k)$ (i.e. twice differentiable), and $\lim_{\alpha \to 0} \phi = 0$.

2. ϕ is decreasing at $\alpha = 0$ (i.e. increasing in the limit $\alpha \to 0$) and convex on X.

Together these imply that if $\phi(\alpha) < 0$ for $\alpha \in X$ then $\phi(\alpha') < 0$ for all $\alpha' < \alpha$. Suppose not, then there exists some point $\alpha' < \alpha$ such that $\phi(\alpha') \geq 0 > \phi(\alpha)$. Since ϕ is continuous on X and initially negative and decreasing it must increase on some subinterval of $(0, \alpha')$ and then decrease again on some subinterval of (α', α). However, this contradicts the fact that ϕ is convex on X, and hence once it starts increasing at any point in X it will not decrease again in any higher interval.

Proof of subclaim 1: ϕ is a linear combination of $H_b(\alpha)$, $H_b(k\alpha)$, and α, which are all real and twice differentiable on $(0, 1/k)$. In particular, $H'(\alpha) = \log_2(1-\alpha) - \log_2(\alpha)$ and $H''(\alpha) = \frac{-1}{\log_2(e)\alpha(1-\alpha)}$. Finally, $\lim_{\alpha \to 0} H_b(\alpha) = \lim_{\alpha \to 0} H_b(k\alpha) = 0$, hence the limit $\alpha \to 0$ of any linear combination of $H_b(\alpha)$, $H_b(k\alpha)$, and α is 0.

Proof of subclaim 2: We first show that $\lim_{\alpha \to 0} \phi' = -\infty$. Note that $\phi'(\alpha) = H_b'(\alpha) + kH_b'(\alpha k) + dkH_b(1/k) - dH_b'(\alpha)$. As $\alpha \to 0$ the limit is determined by the terms involving $H_b'(\alpha)$ and $H'(\alpha k)$, which go to $-\infty$ while $dkH_b(1/k)$ is constant. Since $kH_b'(\alpha k) - (d-1)H_b'(\alpha) = k(\log_2(1-\alpha k) - \log_2(\alpha k)) < -k\log_2(\alpha k) - (d-1)H_b'(\alpha)$ we get:

$$\lim_{\alpha \to 0} \phi' < \lim_{\alpha \to 0} -k \log_2(\alpha k) + (d-1)\log_2(\alpha) = \lim_{\alpha \to 0} (d-k-1)\log_2(\alpha) = -\infty$$

Now looking at the second derivative, $\phi''(\alpha) = k^2 H_b''(\alpha k) - (d-1)H''(\alpha)$:

$$\phi''(\alpha) = \frac{-1}{\log_2(e)\alpha}\left(\frac{k}{1-\alpha k} + \frac{d-1}{1-\alpha}\right)$$

Hence $\phi'' > 0$ if and only if $(d-1)(1-\alpha k) > k(1-\alpha)$. Rearranging the equation, we get $\alpha < \frac{d-k-1}{k(d-2)}$.

5.6.2 Proof of Lemma 29

Define a change of variables $z = x - y$ to get $\phi(x, z) = d(x+z)H_b(x/(x+z)) - H_b(x)) + c$ so that $\hat{\beta}$ is the function defined implicitly by pairs of points satisfying $\phi(\alpha, \hat{\beta}) = 0$. The function $\phi(x, z)$ is continuously differentiable in both variables on the set $(0,1) \times (0,1)$, i.e. its partial derivates $\phi_z = \frac{\partial\phi}{\partial z}$ and $\phi_x = \frac{\partial\phi}{\partial x}$ are each continuous on $(0,1)\times(0,1)$. By the Implicit Function Theorem, $\hat{\beta} : (0,1) \to \mathbb{R}$ is continuously differentiable with derivative $-\phi_x/\phi_z$ defined in an open interval around every point α where $\phi_z \neq 0$. Expanding the binary entropy function $H_b(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ simplifies the inner expression:

$$\begin{aligned}(1/d)(\phi(x,z)-c) &= (z+x)H_b(x/(x+z)) - H_b(x)\\ &= (z+x)\log_2(z+x) - z\log_2(z) - (1-x)\log_2(1-x)\end{aligned}$$

The partial derivatives are: $\phi_x = d\log_2(\frac{x+z}{1-\alpha})$ and $\phi_z = d\log_2(\frac{x+z}{z})$. $\phi_z(\alpha,\hat{\beta}(\alpha))$ is always positive because $(x+z)/z > 1$ for $x,z \in (0,1)$. On the other hand, $\phi_x(\alpha,\hat{\beta}(\alpha) < 0$ if and only if $\alpha + \hat{\beta}(\alpha) > 1-\alpha$. Setting $\hat{\beta}(\alpha) = \beta - \alpha$ this is the case when $\beta + \alpha > 1$. $\hat{\beta}(\alpha)$ is increasing when $\alpha+\beta < 1$ and decreasing when $\alpha+\beta > 1$. It has a local maximum where $\alpha+\beta = 1$. Since $\beta = \hat{\beta}(\alpha) - \alpha$ is a monotonically non-decreasing function of α it follows that if $\alpha+\beta < 1$ then $\alpha'+\beta' < 1$ at every point $\alpha' < \alpha$. Likewise, if $\alpha+\beta > 1$ then $\alpha'+\beta' > 1$ at every point $\alpha' > \alpha$. We therefore conclude that $\hat{\beta}(\alpha)$ is initially increasing and achieves a unique local maximum on $(0,1)$.

5.6.3 Proof of Corollary 7

The function $\hat{\beta}(\alpha)$ from Lemma 29 the implicit function $\hat{\beta}_c$ defined by pairs of points $(\alpha, \beta-\alpha)$ satisfying $\phi_c(\alpha,\beta) = d(\beta H_b(\alpha/\beta) - H_b(\alpha)) + c = 0$ is a smooth and has a unique maximum in any interval $L \subseteq (0,1)$. Furthermore, if the points α and β satisfy $H_b(\alpha) + H_b(\beta) < c \leq 2$ then $\hat{\beta}(\alpha)$ is a lower bound on the "boundary" of sinks of size αn in Chung's construction with overwhelming probability (Lemma 27). We will split the interval $(0.10, 0.80)$ into subintervals $(0.10, 0.33]$ and $[0.33, 0.80)$ and analyze them separately.

For all $\alpha \in [0.33, 0.80)$ the formula in Lemma 27 shows that the expansion for each α is non-decreasing and is at least 0.80 (see Figure 5.3). Thus we can set $c = 1.64 > H_b(0.33) + H_b(0.80)$ and examine the lower bound $\hat{\beta}_c$. It has a unique local maximum in $(0.33, 0.80)$ therefore $\hat{\beta}_c \geq min(\hat{\beta}_c(0.33), \hat{\beta}_c(0.80)) \geq 0.12$. (Observe that $\phi_c(0.33, 0.45) < 0$ and $\phi_c(0.80, 0.92) < 0$ with $d = 8$.)

For $\alpha \in (0.10, 0.33]$ we lazily set $c = 2$. The implicit function $\hat{\beta}_2$ has a unique local max in $(0.10, 0.33]$, so $\hat{\beta}_2 \geq min(\hat{\beta}_2(0.33), \hat{\beta}_2(0.10)) > 0.12$.

5.6.4 Proof of Lemma 30

First set $\alpha = 3/(2d)$ and $\beta = 1/2$. By Lemma 27, we obtain a graph $\mathcal{G}$ that is an (n, α, β) expander with probability $1-\mathsf{negl}(\lambda)\,(nH_b(\alpha))$ as long as $d - \frac{d}{2}H_b(\frac{3}{d}) - H_b(\frac{3}{2d}) - 1 > 0$. Define $g(x) = 1/x - \frac{1}{2x}H_b(3x) - H_b(3x/2) - 1$ so that the condition is equivalent to $g(1/d) > 0$. $H_b(x)$ is real and continuous on $(0, 1)$. Its derivative is $H_b'(x) = \log_2(1-x) - \log_2(x)$ which is positive on $(0, 1/2)$. Differentiating $g(x)$ with respect to x on $(0, 1/3)$ gives $g'(x) = -1/x^2 - \frac{3}{2x}H_b'(3x) + \frac{1}{2x^2}H_b(3x) - \frac{3}{2}H_b'(\frac{3x}{2})$. Observe that on $(0, 1/3)$ both $H_b'(3x) > 0$ and $\frac{1}{2}H_b(3x) - 1 < 0$, therefore $g'(x) < 0$. This means that $g(1/d)$ is increasing as d increases for $d > 3$. Furthermore $\lim_{d \to 3} g(1/d) = 3 - H_b(1/2) - 1 = 1 > 0$.

Finally, it follows as a special case of Lemma 28 that if $\mathcal{G}$ is a d-regular $(n, \frac{3}{2d}, \frac{1}{2})$ bipartite expander for some fixed d then it is an $(n, \alpha, (d/3)\alpha)$ bipartite expander for every $\alpha \leq \frac{3}{2d}$. Setting $\epsilon = 1/3$, we see that $\frac{3}{2d} < \frac{2-1/d}{d-2} = \frac{1/\epsilon - 1/d - 1}{d-2}$. The inequality holds because $4d - 2 > 3d - 6$ for all $d > 0$.

Bibliography

[1] Zcash. `https://z.cash`.

[2] Filecoin: A decentralized storage network. Protocol Labs, 2017. `https://filecoin.io/filecoin.pdf`.

[3] Proof of replication. Protocol Labs, 2017. `https://filecoin.io/proof-of-replication.pdf`.

[4] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In *ASIACRYPT*, 2017.

[5] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2012.

[6] Mustafa Al-Bassam, Alberto Sonnino, Vitalik Buterin, and Ismail Khoffi. Fraud and data availability proofs: Detecting invalid blocks in light clients. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2021.

[7] Ralph Phillips Alexander Lubotzky and Peter Sarnak. Ramanujan graphs. In *Combinatorica*, 1988.

[8] Joël Alwen, Jeremiah Blocki, and Benjamin Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In *CCS*, 2017.

[9] Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In *EUROCRYPT*, 2018.

[10] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

[11] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23. IEEE Computer Society Press, October 1992.

[12] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *ACM Conference on Computer and Communications Security*, 2007.

[13] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. `https://eprint.iacr.org/2021/1377`.

[14] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31. ACM Press, May 1991.

[15] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. Cryptology ePrint Archive, Report 2017/043, 2017. `https://eprint.iacr.org/2017/043`.

[16] Endre Bangerter, Jan Camenisch, and Stephan Krenn. Efficiency limitations for S-protocols for group homomorphisms. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 553–571. Springer, Heidelberg, February 2010.

[17] Endre Bangerter, Jan Camenisch, and Ueli Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 154–171. Springer, Heidelberg, January 2005.

[18] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.

[19] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

[20] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 37–56. Springer, Heidelberg, August 1990.

[21] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[22] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

[23] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio,

editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019.

[24] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[25] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2014.

[26] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

[27] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for r1cs. Cryptology ePrint Archive, Report 2018/828, 2018. `https://eprint.iacr.org/2018/828`.

[28] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

[29] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

[30] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. Deep-fri: Sampling outside the box improves soundness. 26:44, 2019.

[31] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. *IACR Cryptology ePrint Archive*, 2019:336, 2019.

[32] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, Heidelberg, May 1994.

[33] Jean-François Biasse, Michael J. Jacobson Jr., and Alan K. Silvester. Security estimates for quadratic field based cryptosystems. *CoRR*, abs/1004.5512, 2010.

[34] Anurag Bishnoi, Pete L. Clark, Aditya Potukuchi, and John R. Schmitt. On zeros of a polynomial in a finite grid, 2015.

[35] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

[36] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

[37] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, August 2018.

[38] Dan Boneh, Joseph Bonneau, Benedikt Bunz, and Ben Fisch. Verifiable delay functions. 2018. To appear in CRYPTO 2018.

[39] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, August 2019.

[40] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. `https://eprint.iacr.org/2018/712`.

[41] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019.

[42] Joseph Bonneau, Ben Fisch, Juan Benet, and Nicola Greco. Proofs of replication using depth robust graphs. In *Presentation at Blockchain Protocol Analysis and Security Engineering 2018*, 2018. `https://cyber.stanford.edu/bpase2018`.

[43] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

[44] Wieb Bosma and Peter Stevenhagen. On the computation of quadratic 2-class groups. In *Journal de Theorie des Nombres*, 1996.

[45] Sean Bowe. Bellman zk-snarks library, 2016. `https://github.com/zkcrypto/bellman`.

[46] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE Computer Society Press, May 2020.

[47] Kevin D. Bowers, Ari Juels, and Alina Oprea. Proofs of retrievability: theory and implementation. In *CCSW'09 Proceedings of the 2009 ACM workshop on Cloud computing security)*, 2009.

[48] Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[49] Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[50] Ahto Buldas, Peeter Laud, and Helger Lipmaa. Accountable certificate management using undeniable attestations. In Dimitris Gritzalis, Sushil Jajodia, and Pierangela Samarati, editors, *ACM CCS 2000*, pages 9–17. ACM Press, November 2000.

[51] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[52] Benedikt Bünz and Ben Fisch. Schwartz-zippel for multilinear polynomials mod N. Cryptology ePrint Archive, Report 2022/458, 2022. `https://eprint.iacr.org/2022/458`.

[53] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

[54] Vitalik Buterin. Zk rollup, 2016. `https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477`.

[55] Philippe Camacho, Alejandro Hevia, Marcos A. Kiwi, and Roberto Opazo. Strong accumulators from collision-resistant hashing. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC 2008*, volume 5222 of *LNCS*, pages 471–486. Springer, Heidelberg, September 2008.

[56] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki

and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, Heidelberg, March 2009.

[57] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, Heidelberg, August 2002.

[58] Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 179–194. Springer, Heidelberg, March 2010.

[59] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[60] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.

[61] Ethan Cecchetti, Andrew C. Myers, and Owen Arden. Nonmalleable information flow control. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1875–1891. ACM Press, October / November 2017.

[62] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047, 2019. `https://eprint.iacr.org/2019/1047`.

[63] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

[64] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography, 2019.

[65] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020.

[66] F.R.K. Chung. On concentrators, superconcentrators, generalizers, and nonblocking networks. In *Bell System Technical Journal*, 1979.

[67] Jean-Sébastien Coron and David Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 91–101. Springer, Heidelberg, May 2000.

[68] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Heidelberg, April / May 2017.

[69] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. Cryptology ePrint Archive, Report 1999/001, 1999. `https://eprint.iacr.org/1999/001`.

[70] Moni Naor Cynthia Dwork and Hoeteck Wee. Pebbling and proofs of work. In *CRYPTO*, 2005.

[71] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

[72] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, Heidelberg, April / May 2002.

[73] Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive, Report 2008/538, 2008. `https://eprint.iacr.org/2008/538`.

[74] Richard A DeMillo and Richard J Lipton. A probabilistic remark on algebraic program testing. Technical report, GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUTER SCIENCE, 1977.

[75] Yevgeniy Dodis, Salil Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In *Theory of Cryptography Conference (TCC)*, 2009.

[76] Justin Drake. Accumulators, scalability of utxo blockchains, and data availability. `https://ethresear.ch/t/accumulators-scalability-of-utxo-blockchains-and-data-availability/176`.

[77] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *CRYPTO*, 2015.

[78] 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.

[79] Ben Fisch. PoReps: Proofs of space on useful data. Cryptology ePrint Archive, Report 2018/678, 2018. `https://eprint.iacr.org/2018/678`.

[80] Ben Fisch. Poreps: Proofs of space on useful data. Cryptology ePrint Archive, Report 2018/678, 2018. `https://eprint.iacr.org/2018/678`.

[81] Ben Fisch. Tight proofs of space and replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Heidelberg, May 2019.

[82] Ben Fisch, Rafael Pass, and Abhi Shelat. Socially optimal mining pools. In Nikhil R. Devanur and Pinyan Lu, editors, *Web and Internet Economics - 13th International Conference, WINE 2017, Bangalore, India, December 17-20, 2017, Proceedings*, volume 10660 of *Lecture Notes in Computer Science*, pages 205–218. Springer, 2017.

[83] Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 991–1002. Springer, Heidelberg, July 2014.

[84] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. A decentralized public key infrastructure with identity retention. Cryptology ePrint Archive, Report 2014/803, 2014. `https://eprint.iacr.org/2014/803`.

[85] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, August 1997.

[86] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

[87] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

[88] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. In *NDSS 2014*. The Internet Society, February 2014.

[89] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[90] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139. Springer, Heidelberg, May 1999.

[91] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017.

[92] Antonio Faonio Giuseppe Ateniese, Ilario Bonacina and Nicola Galesi. Proofs of space: when space is of the essence. In *Security and Cryptography for Networks, 2014.*, 2014.

[93] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity. *Journal of the ACM*, 38(3), 1991.

[94] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[95] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[96] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[97] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

[98] Safuat Hamdy and Bodo Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 234–247. Springer, Heidelberg, December 2000.

[99] Dan Boneh Henry Corrigan-Gibbs and Stuart Schechter. Balloon hashing: a provably memory-hard function with a data-independent access pattern. In *Asiacrypt*, 2016.

[100] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash Protocol Specification, 2019.

[101] Yuval Ishai, Eyal Kushilevitz, and Rafael Ostrovsky. Efficeint arguments without short pcps. 2007.

[102] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. Acm, 2007.

[103] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004.

[104] Dmitrii Karp. Normalized incomplete beta function: log-concavity in parameters and other properties, 2015.

[105] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

[106] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`.

[107] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[108] O(1) Labs. Coda protocol, 2018. `https://codaprotocol.com/`.

[109] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors,

CRYPTO 2019, Part I, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019.

[110] Russell W.F. Lai and Giulio Malavolta. Optimal succinct arguments via hidden order groups. Cryptology ePrint Archive, Report 2018/705, 2018. `https://eprint.iacr.org/2018/705`.

[111] Sergio Demian Lerner. Proof of unique blockchain storage, 2014. `https://bitslog.wordpress.com/2014/11/03/proof-of-local-blockchain-storage/`.

[112] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 253–269. Springer, Heidelberg, June 2007.

[113] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

[114] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Heidelberg, February 2010.

[115] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001.

[116] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415. Springer, Heidelberg, November / December 2003.

[117] Helger Lipmaa. Secure accumulators from euclidean rings without trusted setup. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 12*, volume 7341 of *LNCS*, pages 224–240. Springer, Heidelberg, June 2012.

[118] Mohammad Mahmoody, Tal Moran, and Salil P Vadhan. Time-lock puzzles in the random oracle model. In *CRYPTO*. Springer, 2011.

[119] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[120] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. Cryptology ePrint Archive, Report 2019/099, 2019. `https://eprint.iacr.org/2019/099`.

[121] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.

[122] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[123] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE Computer Society Press, May 2013.

[124] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.

[125] Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. Authenticated data structures, generically. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 411–424. ACM, 2014.

[126] Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. Cryptology ePrint Archive, Report 2021/1544, 2021. `https://eprint.iacr.org/2021/1544`.

[127] L. Nguyen. Accumulators from bilinear maps and applications. *CT-RSA*, 3376:275–292, 2005.

[128] Kobbi Nissim and Moni Naor. Certificate revocation and certificate update. In *Usenix*, 1998.

[129] Salil Vadhan Omer Reingold and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS*, 2000.

[130] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. `http://eprint.iacr.org/2015/528`.

[131] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, April / May 2017.

[132] Ronald L. Graham Paul Erdös and Endre Szemeredi. On sparse graphs with dense long paths. In *Computers & Mathematics with Applications*, 1975.

[133] Krzysztof Pietrzak. Proofs of Catalytic Space. Cryptology ePrint Archive # 2018/194, 2018.

[134] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 60:1–60:15, 2019.

[135] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9:230–250, 1980.

[136] Arnold K. Pizer. Ramanujan graphs and hecke operators. In *Bull. Amer. Math. Soc.*, 1990.

[137] Henrich Christopher Pöhls and Kai Samelin. On updatable redactable signatures. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14*, volume 8479 of *LNCS*, pages 457–475. Springer, Heidelberg, June 2014.

[138] Soujanya Ponnapalli, Aashaka Shah, Souvik Banerjee, Dahlia Malkhi, Amy Tai, Vijay Chidambaram, and Michael Wei. Rainblock: Faster transaction processing in public blockchains. In Irina Calciu and Geoff Kuenning, editors, *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, pages 333–347. USENIX Association, 2021.

[139] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

[140] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 262–285. Springer, Heidelberg, October / November 2016.

[141] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In *TCC*, 2016.

[142] Ron Rivest, Adi Shamir, and David Wagner. Time-lock puzzles and timed-release crypto. In *MIT Technical report*, 1996.

[143] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[144] J Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962.

[145] Tomas Sander and Amnon Ta-Shma. Auditable, anonymous electronic cash. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 555–572. Springer, Heidelberg, August 1999.

[146] Tomas Sander and Amnon Ta-Shma. Flow control: A new approach for anonymity control in electronic cash systems. In Matthew Franklin, editor, *FC'99*, volume 1648 of *LNCS*, pages 46–61. Springer, Heidelberg, February 1999.

[147] Tomas Sander, Amnon Ta-Shma, and Moti Yung. Blind, auditable membership proofs. In Yair Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 53–71. Springer, Heidelberg, February 2001.

[148] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[149] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish. Resolving the conflict between generality and plausibility in verified computation. 2013.

[150] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. Cryptology ePrint Archive, Report 2019/550, 2019. `https://eprint.iacr.org/2019/550`.

[151] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *Asiacrypt*, 2008.

[152] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems (TOCS)*, 1(1):38–44, 1983.

[153] 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.

[154] Daniel Slamanig. Dynamic accumulator based discretionary access control for outsourced storage with unlinkable access - (short paper). In Angelos D. Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 215–222. Springer, Heidelberg, February / March 2012.

[155] Michael Straka. Class groups for cryptographic accumulators. 2019.

[156] R. Michael Tanner. Explicit concentrators from generalized n-gons. In *Siam Journal on Algebraic and Discrete Methods*, 1984.

[157] Björn Terelius and Douglas Wikström. Efficiency limitations of S-protocols for group homomorphisms revisited. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 461–476. Springer, Heidelberg, September 2012.

[158] Peter Todd. Making UTXO Set Growth Irrelevant With Low-Latency Delayed TXO Commitments . `https://petertodd.org/2016/delayed-txo-commitments`, May 2016.

[159] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and ETH2.0 networks. *CoRR*, abs/2007.02754, 2020.

[160] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

[161] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them: From theoretical possibility to near practicality. *Communications of the ACM*, 58(2), 2015.

[162] Gaven J. Watson, Reihaneh Safavi-Naini, Mohsen Alimomeni, Michael E. Locasto, and Shivaramakrishnan Narayan. Lost: location based storage. In *CCSW*, 2012.

[163] Benjamin Wesolowski. Efficient verifiable delay functions. Cryptology ePrint Archive, Report 2018/623, 2018. `https://eprint.iacr.org/2018/623`.

[164] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

[165] Douglas Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021. `https://eprint.iacr.org/2021/1265`.

[166] Z. Wilcox. The design of the ceremony, 2016. `https://z.cash/blog/the-design-of-the-ceremony.html`.

[167] Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. Cryptology ePrint Archive, Report 2019/317, 2019. `https://eprint.iacr.org/2019/317`.

[168] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks. In Amar Phanishayee and Vyas Sekar, editors, *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*, pages 493–512. USENIX Association, 2022.

[169] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. Cryptology ePrint Archive, Report 2019/1482, 2019. `https://eprint.iacr.org/2019/1482`.

[170] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880. IEEE Computer Society Press, May 2017.

[171] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979.