2018 IEEE Symposium on Security and Privacy

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz[1], Jonathan Bootle[2], Dan Boneh[1],
Andrew Poelstra[3], Pieter Wuille[3], and Greg Maxwell

[1]Stanford University
[2]University College London
[3]Blockstream

***Abstract*—We propose Bulletproofs, a new non-interactive zero-knowledge proof protocol with very short proofs and without a trusted setup; the proof size is only logarithmic in the witness size. Bulletproofs are especially well suited for efficient range proofs on committed values: they enable proving that a committed value is in a range using only $2\log_2(n) + 9$ group and field elements, where n is the bit length of the range. Proof generation and verification times are linear in n.**

Bulletproofs greatly improve on the linear (in n) sized range proofs in existing proposals for confidential transactions in Bitcoin and other cryptocurrencies. Moreover, Bulletproofs supports aggregation of range proofs, so that a party can prove that m commitments lie in a given range by providing only an additive $O(\log(m))$ group elements over the length of a *single* proof. To aggregate proofs from multiple parties, we enable the parties to generate a single proof without revealing their inputs to each other via a simple multi-party computation (MPC) protocol for constructing Bulletproofs. This MPC protocol uses either a constant number of rounds and linear communication, or a logarithmic number of rounds and logarithmic communication. We show that verification time, while asymptotically linear, is very efficient in practice. The marginal cost of batch verifying 32 aggregated range proofs is less than the cost of verifying 32 ECDSA signatures. Bulletproofs build on the techniques of Bootle et al. (EUROCRYPT 2016). Beyond range proofs, Bulletproofs provide short zero-knowledge proofs for general arithmetic circuits while only relying on the discrete logarithm assumption and without requiring a trusted setup. We discuss many applications that would benefit from Bulletproofs, primarily in the area of cryptocurrencies. The efficiency of Bulletproofs is particularly well suited for the distributed and trustless nature of blockchains. The full version of this article is available at [1].

1. Introduction

Blockchain-based cryptocurrencies enable peer-to-peer electronic transfer of value by maintaining a global distributed but synchronized ledger, the *blockchain*. Any independent observer can verify both the current state of the blockchain as well as the validity of all transactions on the ledger. In Bitcoin, this innovation requires that all details of a transaction are public: the sender, the receiver, and the amount transferred. In general, we separate privacy for payments into two properties: (1) anonymity, hiding the identities of sender and receiver in a transaction and (2) confidentiality, hiding the amount transferred. While Bitcoin provides some weak anonymity through the unlinkability of Bitcoin addresses to real world identities, it lacks any confidentiality. This is a serious limitation for Bitcoin and could be prohibitive for many use cases. Would employees want to receive their salaries in bitcoin if it meant that their salaries were published on the public blockchain?

To address the confidentiality of transaction amounts, Maxwell [2] introduced *confidential transactions* (CT), in which every transaction amount involved is hidden from public view using a commitment to the amount. This approach seems to prevent public validation of the blockchain; an observer can no longer check that the sum of transaction inputs is greater than the sum of transaction outputs, and that all transaction values are positive. This can be addressed by including in every transaction a zero-knowledge proof of validity of the confidential transaction.

Current proposals for CT zero-knowledge proofs [3] have either been prohibitively large or required a trusted setup. Neither is desirable. While one could use succinct zero-knowledge proofs (SNARKs) [4], [5], they require a trusted setup, which means that everyone needs to trust that the setup was performed correctly. One could avoid trusted setup by using a STARK [6], but the resulting range proofs while asymptotically efficient are practically larger than even the currently proposed solutions.

Short non-interactive zero-knowledge proofs without a trusted setup, as described in this paper, have many applications in the realm of cryptocurrencies. In any distributed system where proofs are transmitted over a network or stored for a long time, short proofs reduce overall cost.

1.1. Our Contributions

We present Bulletproofs, a new zero-knowledge argument of knowledge[1] system, to prove that a secret committed

1. Proof systems with computational soundness like Bulletproofs are sometimes called argument systems. We will use the terms proof and argument interchangeably.

DOI 10.1109/SP.2018.00020

value lies in a given interval. Bulletproofs do not require a trusted setup. They rely only on the discrete logarithm assumption, and are made non-interactive using the Fiat-Shamir heuristic.

Bulletproofs builds on the techniques of Bootle et al. [7], which yield communication-efficient zero-knowledge proofs. We present a replacement for their inner-product argument that reduces overall communication by a factor of 3. We make Bulletproofs suitable for proving statements on committed values. Examples include a range proof, a verifiable shuffle, and other applications discussed below. We note that a range proof using the protocol of [7] would have required implementing the commitment opening algorithm as part of the verification circuit.

Distributed Bulletproofs generation. We show that Bulletproofs support a simple and efficient multi-party computation (MPC) protocol that allows multiple parties with secret committed values to jointly generate a single small range proof for all their values, without revealing their secret values to each other. One version of our MPC protocol is constant-round but with linear communication. Another variant requires only logarithmic communication, but uses a logarithmic number of rounds. When a confidential transaction has inputs from multiple parties (as in the case of CoinJoin), this MPC protocol can be used to aggregate all the proofs needed to construct the transaction into a single short proof.

Proofs for arithmetic circuits. While we focus on confidential transactions (CT), where our work translates to significant practical savings, we stress that the improvements are not limited to CT. We present Bulletproofs for general NP languages. The proof size is *logarithmic* in the number of multiplication gates in the arithmetic circuit for verifying a witness. The proofs are much shorter than [7] and allow inputs to be Pedersen commitments to elements of the witness.

Optimizations and evaluation. We provide a complete implementation of Bulletproofs that includes many further optimizations described in Section 6. For example, we show how to batch the verification of multiple Bulletproofs so that the cost of verifying every additional proof is significantly reduced. We also provide efficiency comparisons with the range proofs currently used for confidential transactions [2], [8] and with other proof systems. Our implementation includes a general tool for constructing Bulletproofs for any NP language. The tool reads in arithmetic circuits in the Pinocchio [9] format which lets users use their toolchain. This toolchain includes a compiler from C to the circuit format. We expect this to be of great use to implementers who want to use Bulletproofs.

1.2. Applications

We first discuss several applications for Bulletproofs along with related work specific to these applications. Additional related work is discussed in Section 1.3.

1.2.1. Confidential Transactions and Mimblewimble. Bitcoin and other similar cryptocurrencies use a transaction-output-based system where each transaction fully spends the outputs of previously unspent transactions. These unspent transaction outputs are called UTXOs. Bitcoin allows a single UTXO to be spent to many distinct outputs, each associated with a different address. To spend a UTXO a user must provide a signature, or more precisely a *scriptSig*, that enables the transaction SCRIPT to evaluate to true [10]. Apart from the validity of the *scriptSig*, miners verify that the transaction spends previously unspent outputs, and that the sum of the inputs is greater than the sum of the outputs.

Maxwell [2] introduced the notion of a *confidential transaction*, where the input and output amounts in a transaction are hidden in Pedersen commitments [11]. To enable public validation, the transaction contains a zero-knowledge proof that the sum of the committed inputs is greater than the sum of the committed outputs, and that all the outputs are positive, namely they lie in the interval $[0, 2^n]$, where 2^n is much smaller than the group size. All current implementations of confidential transactions [2], [3], [12], [13] use range proofs over committed values, where the proof size is linear in n. These range proofs are the main contributor to the size of a confidential transaction. In current implementations [2], a confidential transaction with only two outputs and 32 bits of precision is 5.4 KB bytes, of which 5 KB are allocated to the range proof. Even with recent optimizations the range proofs would still take up 3.8 KB.

We show in Section 6 that Bulletproofs greatly improve on this, even for a single range proof while simultaneously doubling the range proof precision at marginal additional cost (64 bytes). The logarithmic proof size additionally enables the prover to aggregate multiple range proofs, e.g. for transactions with multiple outputs, into a single short proof. With Bulletproofs, m range proofs are merely $O(\log(m))$ additional group elements over a single range proof. This is already useful for confidential transactions in their current form as most Bitcoin transactions have two or more outputs. It also presents an intriguing opportunity to aggregate multiple range proofs from different parties into one proof, as would be needed, for example, in a CoinJoin transaction [14]. To do so, we present an MPC protocol that lets parties efficiently combine proofs without compromising confidentiality.

Confidential transaction implementations are available in side-chains [3], private blockchains [15], and in the popular privacy-focused cryptocurrency Monero [13]. All these implementations would benefit from Bulletproofs.

At the time of writing, Bitcoin has roughly 50 million UTXOs from 22 million transactions (see statoshi.info). Using a 52-bit representation of bitcoin that can cover all values from 1 satoshi up to 21 million bitcoins, this results in roughly 160GB of range proof data using the current systems. Using aggregated Bulletproofs, the range proofs for all UTXOs would take less than 17GB, about a factor 10 reduction in size.

Mimblewimble. Recently an improvement was proposed to confidential transactions, called Mimblewimble [8], [16], provides further savings.

Jedusor [16] realized that a Pedersen commitment to 0 can be viewed as an ECDSA public key, and that for a valid confidential transaction the difference between outputs, inputs, and transaction fees must be 0. A prover constructing a confidential transaction can therefore sign the transaction with the difference of the outputs and inputs as the public key. This small change removes the need for a scriptSig which greatly simplifies the structure of confidential transactions. Poelstra [8] further refined and improved Mimblewimble and showed that these improvements enable a greatly simplified blockchain in which all spent transactions can be pruned and new nodes can efficiently validate the entire blockchain without downloading any old and spent transactions. Along with further optimizations, this results in a highly compressed blockchain. It consists only of a small subset of the block-headers as well as the remaining unspent transaction outputs and the accompanying range proofs plus an un-prunable 32 bytes per transaction. Mimblewimble also allows transactions to be aggregated before sending them to the blockchain. In Section 4.5, we present a simple and efficient MPC protocol that allows multiple users to generate a single transaction with a single aggregate range proof. The users do not have to reveal their secrets transaction values to any of the other participants. This aggregation of transactions can be seen as a CoinJoin [14] protocol which can improve the anonymity of Mimblewimble transactions.

A Mimblewimble blockchain grows with the size of the UTXO set. Using Bulletproofs, it would only grow with the number of transactions that have unspent outputs, which is much smaller than the size of the UTXO set. Overall, Bulletproofs can not only act as a drop-in replacement for the range proofs in confidential transactions, but it can also help make Mimblewimble a practical scheme with a blockchain that is significantly smaller than the current Bitcoin blockchain.

1.2.2. Provisions. Dagher et al. [17] introduced the Provisions protocol which allows Bitcoin exchanges to prove that they are solvent without revealing any additional information. The protocol crucially relies on range proofs to prevent an exchange from inserting fake accounts with negative balances. These range proofs, which take up over 13GB, are the main contributors to the proof sizes of almost 18GB for a large exchange with 2 million customers. The proof size is in fact linear in the number of customers. Since in this protocol, one party (the exchange) has to construct many range proofs at once, the general Bulletproofs protocol from Section 4.3 is a natural replacement for the NIZK proof used in Provisions. With the proof size listed in Section 6, we obtain that the range proofs would take up less than 2 KB with our protocol. Additionally, the other parts of the proof could be similarly compressed using the protocol from Section 5. The proof would then be dominated by one commitment per customer, with size 62 MB. This is roughly 300 times smaller then the current implementation of Provisions.

1.2.3. Verifiable shuffles. Consider two lists of committed values $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. The goal is to prove that the second list is a permutation of the first. This problem is called a *verifiable shuffle*. It has many applications in voting [18], [19], mix-nets [20], and solvency proofs [17]. Neff [19] gave a practical implementation of a verifiable shuffle and later work improved on it [21], [22]. Currently the most efficient shuffle [23] has size $O(\sqrt{n})$.

Bulletproofs can be used to create a verifiable shuffle of size $O(\log n)$. The two lists of commitments are given as inputs to the circuit protocol from Section 5. The circuit can implement a shuffle by sorting the two lists and then checking that they are equal. A sorting circuit can be implemented using $O(n \cdot \log(n))$ multiplications which means that the proof size will be only $O(\log(n))$. This is much smaller than previously proposed protocols. Given the concrete efficiency of Bulletproofs, a verifiable shuffle using Bulletproofs would be very efficient in practice. Constructing the proof and verifying it takes linear time in n.

1.2.4. NIZK Proofs for Smart Contracts. The Ethereum [24] system uses highly expressive smart contracts to enable complex transactions. Smart contracts, like any other blockchain transaction, are public and provide no inherent privacy. To bring privacy to smart contracts, non-interactive zero-knowledge (NIZK) proofs have been proposed as a tool to enable complex smart contracts that do not leak the user inputs [25]–[27]. However, these protocols are limited as the NIZK proof itself is not suitable for verification by a smart contract. The reason is that communication over the blockchain with a smart contract is expensive, and the smart contract's own computational power is highly limited. SNARKs, which have succinct proofs and efficient verifiers, seem like a natural choice, but current practical SNARKs [4] require a complex trusted setup. The resulting common reference strings (CRS) are long, specific to each application, and possess trapdoors. In Hawk [25], for instance, a different CRS is needed for each smart contract, and either a trusted party is needed to generate it, or an expensive multi-party computation is needed to distribute the trust among a few parties. On the other hand, for small applications like boardroom voting, one can use classical sigma protocols [26], but the proof-sizes and expensive verification costs are prohibitive for more complicated applications. Recently, Campanelli et al. [27] showed how to securely perform zero-knowledge contingent payments (ZKCPs) in Bitcoin, while attacking and fixing a previously proposed protocol [28]. ZKCPs enable the trust less, atomic and efficient exchange of a cryptocurrency vs. some digital good. While ZKCPs support a wide area of applications they fundamentally work for only a single designated verifier and do not allow for public verification. For some smart contracts that have more than two users, public verification is often crucial. In

an auction, for example, all bidders need to be convinced that all bids are well formed.

Bulletproofs improves on this by enabling small proofs that do not require a trusted setup. The Bulletproofs verifier is not cheap, but there are multiple ways to work around this. First, a smart contract may act optimistically and only verify a proof if some party challenges its validity. Incentives can be used to ensure that rational parties never create an incorrect proof nor challenge a correct proof. This can be further improved by using an interactive referee delegation model [29], previously proposed for other blockchain applications [30], [31]. In this model, the prover provides a proof along with a succinct commitment to the verifier's execution trace. A challenger that disagrees with the computation also commits to his computation trace and the two parties engage in an interactive binary search to find the first point of divergence in the computation. The smart contract can then execute this single computation step and punish the party which provided a faulty execution trace. The intriguing property of this protocol is that even when a proof is challenged, the smart contract only needs to verify a single computation step, i.e. a single gate of the verification circuit. In combination with small Bulletproofs, this can enable more complex but privacy preserving smart contracts. Like in other applications, these NIZK proofs would benefit from the MPC protocol that we present in Section 4.5 to generate Bulletproofs distributively. Consider an auction smart contract where bidders in the first round submit commitments to bids and in the second round open them. A NIZK can be used to proof properties about the bids, e.g. they are in some range, without revealing them. Using Bulletproofs' MPC multiple bidders can combine their Bulletproofs into a single proof.

1.2.5. Short Non-Interactive Protocols for Arithmetic Circuits without a Trusted Setup. Non-interactive zero-knowledge protocols for general statements are not possible without using a common reference string, which should be known by both the prover and the verifier. Many efficient non-interactive zero-knowledge proofs and arguments for arithmetic circuit satisfiability have been developed [4]–[6], [32]–[34], and highly efficient protocols are known. However, aside from their performance, these protocols differ in the complexity of their common reference strings. Some, such as those in [4], are highly structured, and sometimes feature a trapdoor, while some are simply chosen uniformly at random. Security proofs assume that the common reference string was honestly generated. In practice, the common reference string can be generated by a trusted third party, or using a secure multi-party computation protocol. The ladder helps to alleviate concerns about embedded trapdoors, as with the trusted setup ceremony used to generate the public parameters for [35].

Zero-knowledge SNARKs have been the subject of extensive research [4], [5], [36]–[40]. They generate constant-sized proofs for any statement, and have extremely fast verification time. However, they have highly complex common reference strings which require lengthy and computationally intensive protocols [41] to generate distributively. They also rely on strong unfalsifiable assumptions such as the knowledge-of-exponent assumption.
A uniformly-random common reference strings, on the other hand, can be derived from common random strings, like the digits of π or by assuming that hash functions behave like random oracle. Examples of non-interactive protocols that do not require a trusted setup include [6], [7], [32], [42], [43].

Ben-Sasson et al. present a proof system [44] and implementation [43] called Scalable Computational Integrity (SCI). While SCI has a simple setup, and relies only on collision-resistant hash functions, the system is not zero-knowledge and still experiences worse performance than [4], [7]. The proof sizes are roughly 42 MB large in practice for a reasonable circuit. In subsequent work Ben-Sasson et al. presented STARKs [6], which are zero-knowledge and more efficient than SCI. However even with these improvements the proof size is still over 200 KB (and grows logarithmically) at only 60-bit security for a circuit of size 2^{17}. A Bulletproof for such a circuit at twice the security would be only about 1 KB. Constructing STARKs is also very costly in terms of memory requirements because of the large FFT that is required to make proving efficient. Ames et al. [45] presented a proof system with linear verification time but only square root proof complexity building on the MPC in the head technique. Wahby [46] recently present a cryptographic zero-knowledge proof system which achieves square root verifier complexity and proof size based on the proofs for muggles [47] techniques in combination with a sub-linear polynomial commitment scheme.

1.3. Additional Related Work

Much of the research related to electronic payments that predates Bitcoin [48] focused on efficient anonymous and confidential payments [20], [49] . With the advent of blockchain-based cryptocurrencies, the question of privacy and confidentiality in transactions has gained a new relevance. While the original Bitcoin paper [48] claimed that Bitcoin would provide anonymity through pseudonymous addresses early work on Bitcoin showed that the anonymity is limited [50], [51]. Given these limitations, various methods have been proposed to help improve the privacy of Bitcoin transactions. CoinJoin [14], proposed by Maxwell, allows users to hide information about the amounts of transactions by merging two or more transactions. This ensures that among the participants who join their transactions, it is impossible to tell which transaction inputs correspond to which transaction outputs. However, users do require some way of searching for other users, and furthermore, should be able to do so without relying on a trusted third party. CoinShuffle [52] tried to fulfill this requirement by taking developing the ideas of CoinJoin and proposing a new Bitcoin mixing protocol which is completely decentralized. Monero [53] is a cryptocurrency which employs cryptographic techniques to achieve strong privacy guarantees. These include stealth addresses, ring-signatures [54], and

ring confidential transactions [13]. ZeroCash [35] offers optimal privacy guarantees but comes at the cost of expensive transaction generation and the requirement of a trusted setup.

Range proofs. Range proofs are proofs that a secret value, which has been encrypted or committed to, lies in a certain interval. Range proofs do not leak any information about the secret value, other than the fact that they lie in the interval. Lipmaa [55] presents a range proof which uses integer commitments, and Lagrange's four-square theorem which states that every positive integer y can be expressed as a sum of four squares. Groth [56] notes that the argument can be optimized by considering $4y + 1$, since integers of this form only require three squares. The arguments require only a constant number of commitments. However, each commitment is large, as the security of the argument relies on the Strong RSA assumption. Additionally, a trusted setup is required to generate the RSA modulus or a prohibitively large modulus needs to be used [57]. Camenisch et al. [58] use a different approach. The verifier provides signatures on a small set of digits. The prover commits to the digits of the secret value, and then proves in zero-knowledge that the value matches the digits, and that each commitment corresponds to one of the signatures. They show that their scheme can be instantiated securely using both RSA accumulators [59] and the Boneh-Boyen signature scheme [60]. However, these range proofs require a trusted setup. Approaches based on the n-ary digits of the secret value are limited to proving that the secret value is in an interval of the form $\left[0, n^k - 1\right]$. One can produce range proofs for more general intervals by using homomorphic commitments to translate intervals, and by using a combination of two different range proofs to conduct range proofs for intervals of different widths. However, [61] presented an alternative digital decomposition which enables an interval of general width to be handled using a single range proof.

2. Preliminaries

Before we present Bulletproofs, we first review some of the underlying tools. In what follows, a PPT adversary $\mathcal{A}$ is a probabilistic interactive Turing Machine that runs in polynomial time in the security parameter λ. We will drop the security parameter λ from the notation when it is implicit.

2.1. Commitments

Definition 1 (Commitment)**.** *A non-interactive commitment scheme consists of a pair of probabilistic polynomial time algorithms* $(\mathrm{Setup}, \mathrm{Com})$. *The setup algorithm* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda)$ *generates public parameters* pp *for the scheme, for security parameter* λ. *The commitment algorithm* $\mathrm{Com}_{\mathrm{pp}}$ *defines a function* $\mathsf{M}_{\mathrm{pp}} \times \mathsf{R}_{\mathrm{pp}} \rightarrow \mathsf{C}_{\mathrm{pp}}$ *for message space* M_{pp}, *randomness space* R_{pp} *and commitment space* C_{pp} *determined by* pp. *For a message* $x \in \mathsf{M}_{\mathrm{pp}}$, *the algorithm draws* $r \xleftarrow{\$} \mathsf{R}_{\mathrm{pp}}$ *uniformly at random, and computes commitment* $\mathbf{com} = \mathrm{Com}_{\mathrm{pp}}(x; r)$.

Definition 2 (Homomorphic Commitments)**.** *A homomorphic commitment scheme is a non-interactive commitment scheme such that* $\mathsf{M}_{\mathrm{pp}}, \mathsf{R}_{\mathrm{pp}}$ *and* C_{pp} *are all abelian groups, and for all* $x_1, x_2 \in \mathsf{M}_{\mathrm{pp}}$, $r_1, r_2 \in \mathsf{R}_{\mathrm{pp}}$, *we have*

$$\mathrm{Com}(x_1; r_1) + \mathrm{Com}(x_2; r_2) = \mathrm{Com}(x_1 + x_2; r_1 + r_2)$$

Definition 3 (Hiding Commitment)**.** *A commitment scheme is said to be hiding if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mu(\lambda)$ *such that.*

$$\left| \mathrm{P}\left[b = b' \;\middle|\; \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda); \\ (x_0, x_1) \in \mathsf{M}_{\mathrm{pp}}^2 \leftarrow \mathcal{A}(\mathrm{pp}), \\ b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} \mathsf{R}_{\mathrm{pp}}, \\ \mathbf{com} = \mathrm{Com}(x_b; r), \\ b' \leftarrow \mathcal{A}(\mathrm{pp}, \mathbf{com}) \end{array} \right] - \frac{1}{2} \right| \leqslant \mu(\lambda)$$

where the probability is over b, r, Setup *and* $\mathcal{A}$. *If* $\mu(\lambda) = 0$ *then we say the scheme is* perfectly hiding.

Definition 4 (Binding Commitment)**.** *A commitment scheme is said to be binding if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* μ *such that.*

$$\mathrm{P}\left[\begin{array}{l} \mathrm{Com}(x_0; r_0) \\ = \mathrm{Com}(x_1; r_1) \\ \wedge x_0 \neq x_1 \end{array} \;\middle|\; \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda), \\ x_0, x_1, r_0, r_1 \leftarrow \mathcal{A}(\mathrm{pp}) \end{array} \right] \leqslant \mu(\lambda)$$

where the probability is over Setup *and* $\mathcal{A}$. *If* $\mu(\lambda) = 0$ *then we say the scheme is* perfectly binding.

In what follows, the order p of the groups used is implicitly dependent on the security parameter λ to ensure that discrete log in these groups is intractable for PPT adversaries.

Definition 5 (Pedersen Commitment)**.** $\mathsf{M}_{\mathrm{pp}}, \mathsf{R}_{\mathrm{pp}} = \mathbb{Z}_p$, $\mathsf{C}_{\mathrm{pp}} = \mathbb{G}$ *of order* p.
$\mathrm{Setup} : g, h \xleftarrow{\$} \mathbb{G}$
$\mathrm{Com}(x; r) = (g^x h^r)$

Definition 6 (Pedersen Vector Commitment)**.** $\mathsf{M}_{\mathrm{pp}} = \mathbb{Z}_p^n$, $\mathsf{R}_{\mathrm{pp}} = \mathbb{Z}_p$, $\mathsf{C}_{\mathrm{pp}} = \mathbb{G}$ *with* $\mathbb{G}$ *of order* p
$\mathrm{Setup} : \mathbf{g} = (g_1, \ldots, g_n), h \xleftarrow{\$} \mathbb{G}$
$\mathrm{Com}(\mathbf{x} = (x_1, \ldots, x_n); r) = h^r \mathbf{g}^{\mathbf{x}} = h^r \prod_i g_i^{x_i} \in \mathbb{G}$

The Pedersen vector commitment is perfectly hiding and computationally binding under the discrete logarithm assumption. We will often set $r = 0$, in which case the commitment is binding but not hiding.

2.2. Zero-Knowledge Arguments of Knowledge

Bulletproofs are zero-knowledge arguments of knowledge. A zero-knowlege proof of knowledge is a protocol in which a prover can convince a verifier that some statement holds without revealing any information about why it holds. A prover can for example convince a verifier that a confidential transaction is valid without revealing why that is the case, i.e. without leaking the transacted values. An argument is a proof which holds only if the prover is computationally bounded and certain computational hardness assumptions

hold. We give a formal definition of zero-knowledge arguments of knowledge in Appendix A

2.3. Notation

Let $\mathbb{G}$ denote a cyclic group of prime order p, and let $\mathbb{Z}_p$ denote the ring of integers modulo p. Let $\mathbb{G}^n$ and $\mathbb{Z}_p^n$ be vector spaces of dimension n over $\mathbb{G}$ and $\mathbb{Z}_p$ respectively. Let $\mathbb{Z}_p^\star$ denote $\mathbb{Z}_p \backslash \{0\}$. Generators of $\mathbb{G}$ are denoted by $g, h, v, u \in \mathbb{G}$. Group elements which represent commitments are capitalized and blinding factors are denoted by Greek letters, i.e. $C = g^a h^\alpha \in \mathbb{G}$ is a Pedersen commitment to a. If not otherwise clear from context $x, y, z \in \mathbb{Z}_p^\star$ are uniformly distributed challenges. Throughout the paper, we will also be using vector notations defined as follows. Bold font denotes vectors, i.e. $\mathbf{a} \in \mathbb{F}^n$ is a vector with elements $a_1, \ldots, a_n \in \mathbb{F}$. Capitalized bold font denotes matrices, i.e. $\mathbf{A} \in \mathbb{F}^{n \times m}$ is a matrix with n rows and m columns such that $a_{i,j}$ is the element of $\mathbf{A}$ in the ith row and jth column. For a scalar $c \in \mathbb{Z}_p$ and a vector $\mathbf{a} \in \mathbb{Z}_p^n$, we denote by $\mathbf{b} = c \cdot \mathbf{a} \in \mathbb{Z}_p^n$ the vector where $b_i = c \cdot a_i$. Furthermore, let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i \cdot b_i$ denote the inner product between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$, and $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, \ldots, a_n \cdot b_n) \in \mathbb{F}^n$ the Hadamard product or entry wise multiplication of two vectors. The matrix product between a matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{Z}_p^m$ is denoted as $\mathbf{A} \cdot \mathbf{b} = \mathbf{c} \in \mathbb{Z}_p^n$ with $c_i = \sum_{j=1}^m a_{i,j} \cdot b_j \in \mathbb{Z}_p$. Note that the product is not commutative, i.e. for $\mathbf{c} \in \mathbb{Z}_p^n$ and $\mathbf{d} \in \mathbb{Z}_p^m$: $\mathbf{c} \cdot \mathbf{A} = \mathbf{d} \in \mathbb{Z}_p^m$ such that $d_j = \sum_{i=1}^n c_i \cdot a_{i,j} \in \mathbb{Z}_p$.
Further, we extend the vector notation to vectors of group elements. This is useful when dealing with Pedersen vector commitments. Specifically let $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ be a vector of generators then $C = \mathbf{g}^{\mathbf{a}} = \prod_{i=1}^n g_i^{a_i}$ is a binding (but not hiding) commitment to vector $\mathbf{a} \in \mathbb{Z}_p^n$. Given such commitment C and a vector $\mathbf{b} \in \mathbb{Z}_p^n$ with non-zero entries, it is possible to view C as a new commitment to $\mathbf{a} \circ \mathbf{b}$. This is done by defining $g_i' = g_i^{(b_i^{-1})}$ such that $C = \prod_{i=1}^n g_i'^{a_i \cdot b_i}$. The binding property of this new commitment holds if the old commitment was binding.
Let $\mathbf{a} \,\|\, \mathbf{b}$ denote the concatenation of two vectors: if $\mathbf{a} \in \mathbb{Z}_p^n$ and $\mathbf{b} \in \mathbb{Z}_p^m$ then $\mathbf{a} \,\|\, \mathbf{b} \in \mathbb{Z}_p^{n+m}$. We denote slices of vectors using Python notation:

$$\mathbf{a}_{[:k]} = (a_1, \ldots, a_k) \in \mathbb{F}^k, \quad \mathbf{a}_{[k:]} = (a_{k+1}, \ldots, a_n) \in \mathbb{F}^{n-k}.$$

For $k \in \mathbb{Z}_p^\star$ we use $\mathbf{k}^n$ to denote the vector containing the first n powers of k, i.e. $\mathbf{k}^n = (1, k, k^2, \ldots, k^{n-1}) \in (\mathbb{Z}_p^\star)^n$. For example, $\mathbf{2}^n = (1, 2, 4, \ldots, 2^{n-1})$. Equivalently $\mathbf{k}^{-n} = (\mathbf{k^{-1}})^n = (1, k^{-1}, \ldots, k^{-n+1})$.

Finally, we write $\{(\text{Public Input}; \text{Witness}) : \text{Relation}\}$ to denote the relation Relation using the specified Public Input and Witness.

3. Improved Inner-Product Argument

Bootle et al. [7] introduced a communication efficient inner-product argument and show how it can be leveraged to construct zero-knowledge proofs for arithmetic circuit satisfiability with low communication complexity. The argument is an argument of knowledge that the prover knows the openings of two binding Pedersen vector commitments that satisfy a given inner product relation.

We reduce the communication complexity of the argument from $6\log_2(n)$ in [7] to only $2\log_2(n)$, where n is the dimension of the two vectors. We achieve this improvement by modifying the relation being proved. Our argument is sound, but is not zero-knowledge. We then show that this protocol gives a public-coin communication efficient zero-knowledge range proof on a set of committed values, and a zero-knowledge proof system for arbitrary arithmetic circuits (Sections 4 and 5). By applying the Fiat-Shamir heuristic we obtain a short non-interactive proof (Section 4.4).

Overview. The inner product argument takes as input a binding vector commitment to the two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$, as well as $c \in \mathbb{Z}_p$, and proves that $c = \langle \mathbf{a}, \mathbf{b} \rangle$. Logarithmic communication is achieved by running $\log_2 n$ iterations, where in each iteration the the dimension of $\mathbf{a}$ and $\mathbf{b}$ is halved.

To give some intuition, consider a simple example: for independent generators g_1, g_2, suppose the verifier is given g_1, g_2 and a binding commitment $P = g_1^{a_1} g_2^{a_2} \in \mathbb{G}$. The prover can prove knowledge of a_1, a_2 by sending both values to the verifier, but we can do better. Suppose the verifier has $R = g_1^{a_2}$ and $L = g_2^{a_1}$. It sends to the prover a random challenge $x \in \mathbb{Z}_p$ and they both compute:

$$g' = g_1^{(x^{-1})} g_2^x \quad \text{and} \quad P' = L^{(x^2)} \cdot P \cdot R^{(x^{-2})}.$$

A simple calculation shows that $P' = (g')^{a_1 \cdot x + a_2 \cdot x^{-1}}$. Now, it can be shown that the prover can prove knowledge of a_1, a_2 by simply sending $a' = a_1 \cdot x + a_2 \cdot x^{-1} \in \mathbb{Z}_p$ to the verifier. The verifier accepts if $P' = (g')^{a'}$. This a' is half the size of (a_1, a_2).

In this example the prover sends L, R and a' to the verifier, so this proof of knowledge is no better then sending a_1, a_2. However, this technique generalizes to more dimensions. When the two dimensional vector (a_1, a_2) is replaced by an n dimensional vector $\mathbf{a} \in \mathbb{Z}_p^n$, the prover can prove knowledge of $\mathbf{a}$ by only sending $L, R \in \mathbb{G}$ and a vector of dimension $n/2$. This is a significant savings over sending all of $\mathbf{a}$ to the verifier. We can then use the same communication efficient method to recursively prove knowledge of the vector of dimension $n/2$. The resulting $\log_2(n)$ round protocol generates only $O(\log_2(n))$ traffic.

Moreover, we show in Protocol 2 that this can be done for two vectors in parallel such that the inner product of the two vectors only changes by a correction factor that the verifier can compute itself from the challenge. This lets the prover convince the verifier that the inner-product of two committed vectors $\mathbf{a}, \mathbf{b}$ is a value c.

The inner-product argument. The input to the inner product argument are independent generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, a scalar $c \in \mathbb{Z}_p$, and the binding vector commitment $P \in \mathbb{G}$ such that

$$\mathcal{V}_{\mathsf{IP}} : x \xleftarrow{\$} \mathbb{Z}_p^\star \quad (3)$$
$$\mathcal{V}_{\mathsf{IP}} \rightarrow \mathcal{P}_{\mathsf{IP}} : x \quad (4)$$
$$P' = P \cdot u^{x \cdot c} \quad (5)$$
$$\text{Run Protocol 2 on Input } (\mathbf{g}, \mathbf{h}, u^x, P'; \mathbf{a}, \mathbf{b}) \quad (6)$$

Protocol 1: Proof system for Relation (1) using Protocol 2. $u \in \mathbb{G}$ is a fixed group element with an unknown discrete-log relative to $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$.

$P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$. The argument demonstrates that $\langle \mathbf{a}, \mathbf{b} \rangle = c$ given pairwise hardness of computing discrete logarithm relations between each pair of two distinct group elements from $\mathbf{g}, \mathbf{h}$. We assume that n is a power of 2. When using the argument we can easily pad the circuit/range proof construction to ensure that this holds.

With this setup, the inner product protocol, described in Protocol 1, is an efficient proof system for the following relation:

$$\left\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n,\ P \in \mathbb{G},\ c \in \mathbb{Z}_p\ ;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\right\} \quad (1)$$

Protocol 1 uses internally a fixed group element $u \in \mathbb{G}$ that has an unknown discrete-log relative to $\mathbf{g}, \mathbf{h}$. The protocol's total communication is $2 \cdot \lceil \log_2(n) \rceil$ elements in $\mathbb{G}$ plus 2 elements in $\mathbb{Z}_p$. The prover's work is dominated by $4n$ group exponentiations and the verifier's work by $2n$ exponentiations. For more details on our implementation and its optimizations see Section 6.

We describe the protocol in two parts. Protocol 2 is a proof system for the following relation:

$$\left\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a}, \mathbf{b} \rangle}\right\} \quad (2)$$

Then Protocol 1 uses the proof system for Relation (2) to build a new proof system for Relation (1).

Theorem 1 (Inner-Product Argument)**.** *The argument presented in Protocol 1 has perfect completeness and statistical witness-extended-emulation for either extracting a non-trivial discrete logarithm relation between* $\mathbf{g}, \mathbf{h}, u$ *or extracting a valid witness* $\mathbf{a}, \mathbf{b}$*.*

The proofs for all theorems are given in the appendix.

$$\text{Input: } (\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}\ ;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) \quad (7)$$
$$\mathcal{P}_{\mathsf{IP}}\text{'s input: } (\mathbf{g}, \mathbf{h}, u, P, \mathbf{a}, \mathbf{b}) \quad (8)$$
$$\mathcal{V}_{\mathsf{IP}}\text{'s input: } (\mathbf{g}, \mathbf{h}, u, P) \quad (9)$$
$$\text{Output: } \{\mathcal{V}_{\mathsf{IP}} \text{ accepts or } \mathcal{V}_{\mathsf{IP}} \text{ rejects}\} \quad (10)$$
$$\text{if } n = 1 : \quad (11)$$
$$\mathcal{P}_{\mathsf{IP}} \rightarrow \mathcal{V}_{\mathsf{IP}} : a, b \quad (12)$$
$$c = a \cdot b \quad (13)$$
$$\mathcal{V}_{\mathsf{IP}} \text{ checks if } P = g^a h^b u^c : \quad (14)$$
$$\text{if yes, } \mathcal{V}_{\mathsf{IP}} \text{ accepts} \quad (15)$$
$$\text{otherwise, } \mathcal{V}_{\mathsf{IP}} \text{ rejects} \quad (16)$$
$$\text{else: } (n > 1) \quad (17)$$
$$\mathcal{P}_{\mathsf{IP}} \text{ computes:} \quad (18)$$
$$n' = \frac{n}{2} \quad (19)$$
$$c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p \quad (20)$$
$$c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p \quad (21)$$
$$L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L} \in \mathbb{G} \quad (22)$$
$$R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R} \in \mathbb{G} \quad (23)$$
$$\mathcal{P}_{\mathsf{IP}} \rightarrow \mathcal{V}_{\mathsf{IP}} : L, R \quad (24)$$
$$\mathcal{V}_{\mathsf{IP}} : x \xleftarrow{\$} \mathbb{Z}_p^\star \quad (25)$$
$$\mathcal{V}_{\mathsf{IP}} \rightarrow \mathcal{P}_{\mathsf{IP}} : x \quad (26)$$
$$\mathcal{P}_{\mathsf{IP}} \text{ and } \mathcal{V}_{\mathsf{IP}} \text{ compute:} \quad (27)$$
$$\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'} \quad (28)$$
$$\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'} \quad (29)$$
$$P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G} \quad (30)$$
$$\mathcal{P}_{\mathsf{IP}} \text{ computes:} \quad (31)$$
$$\mathbf{a}' = \mathbf{a}_{[:n']} \cdot x + \mathbf{a}_{[n':]} \cdot x^{-1} \in \mathbb{Z}_p^{n'} \quad (32)$$
$$\mathbf{b}' = \mathbf{b}_{[:n']} \cdot x^{-1} + \mathbf{b}_{[n':]} \cdot x \in \mathbb{Z}_p^{n'} \quad (33)$$
$$\text{recursively run Protocol 2 on input} \quad (34)$$
$$(\mathbf{g}', \mathbf{h}', u, P'; \mathbf{a}', \mathbf{b}') \quad (35)$$

Protocol 2: Improved Inner-Product Argument

4. Range Proof Protocol with Logarithmic Size

We now present a novel protocol for conducting short and aggregatable range proofs. The protocol uses the improved inner product argument from Protocol 1. First, in Section 4.1, we describe how to construct a range proof that requires the verifier to check an inner product between two vectors. Then, in Section 4.2, we show that this check can be replaced with an efficient inner-product argument. In Section 4.3, we show how to efficiently aggregate m range proofs into one short proof. In Section 4.4, we discuss how interactive public coin protocols can be made non-interactive by using the Fiat-Shamir heuristic, in the random oracle model. In Section 4.5 we present an efficient MPC protocol that allows multiple parties to construct a single aggregate range proof. In the full version [1], we discuss an extension that enables a switch to quantum-secure range proofs in the future.

4.1. Inner-Product Range Proof

We present a protocol which uses the improved inner-product argument to construct a range proof. The proof convinces the verifier that a commitment V contains a number v that is in a certain range, without revealing v. Bootle et al. [7] presents a proof system for arbitrary arithmetic circuits, and in Section 5, we demonstrate that our improvements to the inner product argument also transfer to this proof system. It is of course possible to prove that a commitment is in a given range using an arithmetic circuit and asymptotically [7] could be used to construct logarithmically (in the length of v) sized range proofs.

However, the circuit would need to implement the commitment function, e.g. a multi-exponentiation for Pedersen commitments, leading to a large and complex circuit.

We, therefore, demonstrate that we can construct a range proof more directly. The range proof takes advantage of the fact that if V is a Pedersen commitment, then it is an element in the same group that is used to perform the inner product argument. We extend this idea in Section 5 to show that our circuit can take an arbitrary number of commitments as input.

The proof system uses the homomorphic property of Vector Pedersen commitments to construct commitments to two polynomials $l(X)$ and $r(X)$ in $\mathbb{Z}_p^n[X]$, i.e. the coefficients of $l(X)$ and $r(X)$ are vectors in $\mathbb{Z}_p^n$. Using these vector-polynomial commitments, the prover and verifier engage in an inner product argument to verifiably compute the inner product of $l(X)$ and $r(X)$. These polynomials are carefully constructed such that the zero-coefficient of $\langle l(X), r(X)\rangle \in \mathbb{Z}_p[X]$ has a special form if and only if v is in the range. This can be viewed as encoding the range proof circuit in the zero-coefficient of $\langle l(x), r(x)\rangle$. For simplicity, we describe the product as an interactive protocol where all the verifiers messages are random elements in $\mathbb{Z}_p$. As discussed in Section 4.4, this protocol can be turned into a non-interactive range proof using the Fiat-Shamir heuristic. In Section 4.2 we show how to use the inner product argument to turn the range proof into a highly efficient proof whose size only grows logarithmically in the bits of the range proven.

Formally, let v be a number in $[0, 2^n - 1]$ with $n = O(\lambda)$, and V be a commitment to v using randomness γ. Let $\mathbf{a} = (a_1, \ldots, a_n)$ be the vector containing the bits of v, so that $\langle \mathbf{a}, \mathbf{2}^n\rangle = v$. The prover $\mathcal{P}$ commits to $\mathbf{a}$ as well as blinding vectors $\mathbf{s}_L, \mathbf{s}_R$ using constant sized vector commitments. $\mathcal{P}$ then constructs the polynomial $t(X) \in \mathbb{Z}_p[X]$ as a function of $\mathbf{a}, \mathbf{s}_L, \mathbf{s}_R$ whose zero coefficient is independent of $\mathbf{a}$ if and only if $\mathbf{a}$ indeed contains only bits. $t(X)$ is exactly the inner product of $l(X), r(X) \in \mathbb{Z}_p^n[X]$. $l(X)$ and $r(X)$ are such that the Verifier $\mathcal{V}$ can himself construct a commitment to them.

Concretely the proof system proves the following relation which is equivalent to a range proof relation by Definition 12 using a Pedersen commitment scheme and range $[0, 2^n - 1]$:

$$\begin{aligned}&\big\{(V, g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n; v, \gamma \in \mathbb{Z}_p) :\\ &V = h^\gamma g^v \wedge v \in [0, 2^n - 1]\big\}\end{aligned}$$

To prove the statement, $\mathcal{P}$ and $\mathcal{V}$ engage in the following zero knowledge protocol.

$$\mathbf{a}_L \in \{0,1\}^n \text{ s.t. } \langle \mathbf{a}_L, \mathbf{2}^n\rangle = v \tag{36}$$
$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \qquad \in \mathbb{Z}_p^n \tag{37}$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p \tag{38}$$
$$A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \qquad \in \mathbb{G} \tag{39}$$
$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n \tag{40}$$
$$\rho \xleftarrow{\$} \mathbb{Z}_p \tag{41}$$
$$S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \qquad \in \mathbb{G} \tag{42}$$
$$\mathcal{P} \to \mathcal{V} : A, S \tag{43}$$
$$\mathcal{V} : y, z \xleftarrow{\$} \mathbb{Z}_p^\star \tag{44}$$
$$\mathcal{V} \to \mathcal{P} : y, z \tag{45}$$

The prover now constructs the two degree 1 polynomials $l(X)$ and $r(X)$ in $\mathbb{Z}_p^n[X]$ and computes

$$t(X) = \langle l(X), r(X)\rangle \in \mathbb{Z}_p[X]$$

$\mathcal{V}$ can construct commitments to $l(X), r(X)$ from V, A, and S, as well as y and z.

$$l(X) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot X \qquad \in \mathbb{Z}_p[X]$$
$$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \qquad \in \mathbb{Z}_p[X]$$
$$t(X) = \langle l(X), r(X)\rangle = \sum_{i=0}^{2} t_i \cdot X^i \qquad \in \mathbb{Z}_p[X]$$
$$\begin{aligned} t_0 &= \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle + z \cdot \langle \mathbf{a}_L - \mathbf{a}_R, \mathbf{y}^n\rangle \\ &\quad + z^2 \cdot \langle \mathbf{2}^n, \mathbf{a}_L\rangle + k(y,z) \qquad \in \mathbb{Z}_p \end{aligned}$$
$$k(y,z) = -z^2 \cdot \langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3 \cdot \langle \mathbf{1}^n, \mathbf{2}^n\rangle \qquad \in \mathbb{Z}_p$$

Note that if

$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \wedge \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n \wedge \langle \mathbf{a}_L, \mathbf{2}^n\rangle = v \tag{46}$$

then

$$\begin{aligned} t_0 &= z \cdot \langle \mathbf{1}^n, \mathbf{y}^n\rangle + z^2 \cdot \langle \mathbf{a}_L, \mathbf{2}^n\rangle + k(y,z) \\ &= z \cdot \langle \mathbf{1}^n, \mathbf{y}^n\rangle + z^2 \cdot v + k(y,z) \end{aligned}$$

i.e. t_0 is a function of y, z and v. Furthermore, a commitment to t_0 can be constructed from y, z, and a homomorphic commitment to v. The proof of Theorem 1 shows if t_0 has this form, then (46) must hold. The prover therefore commits to t_1, t_2 using Pedersen commitments, and $\mathcal{P}$ and

$\mathcal{V}$ engage in a polynomial identity testing protocol to show that $t(X) = \langle l(X), r(X) \rangle$.

$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p \tag{47}$$
$$T_i = g^{t_i} h^{\tau_i} \quad i = \{1, 2\} \qquad \in \mathbb{G} \tag{48}$$
$$\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2 \tag{49}$$
$$\mathcal{V} : x \xleftarrow{\$} \mathbb{Z}_p^\star \tag{50}$$
$$\mathcal{V} \rightarrow \mathcal{P} : x \tag{51}$$
$$\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 + z^2 \cdot \gamma \qquad \in \mathbb{Z}_p \tag{52}$$
$$\mu = \alpha + \rho \cdot x \qquad \in \mathbb{Z}_p \tag{53}$$
$$t = \langle \mathbf{l}, \mathbf{r} \rangle \qquad \in \mathbb{Z}_p \tag{54}$$
$$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x \qquad \in \mathbb{Z}_p^n \tag{55}$$
$$\mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) \tag{56}$$
$$+ z^2 \cdot \mathbf{2}^n \qquad \in \mathbb{Z}_p^n \tag{57}$$
$$\mathcal{P} \rightarrow \mathcal{V} : \tau_x, \mu, t, \mathbf{l}, \mathbf{r} \tag{58}$$

The verifier checks that $\mathbf{l}$ and $\mathbf{r}$ are in fact $l(x)$ and $r(x)$ and checks that $t(x) = \langle \mathbf{l}, \mathbf{r} \rangle$:

$$h'_i = h_i^{y^{-i+1}} \quad \forall i \in [1, n] \qquad \in \mathbb{G} \tag{59}$$
$$t \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \qquad \in \mathbb{Z}_p \tag{60}$$
$$g^t h^{\tau_x} \stackrel{?}{=} g^{k(y,z) + z \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle} \cdot V^{z^2} \cdot T_1^x \cdot T_2^{x^2} \tag{61}$$
$$P = AS^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h'}^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} \qquad \in \mathbb{G} \tag{62}$$
$$P \stackrel{?}{=} h^\mu \mathbf{g}^{\mathbf{l}} \mathbf{h'}^{\mathbf{r}} \tag{63}$$

Corollary 1 (Range Proof)**.** *The range proof presented in Section 4.1 has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

Proof. The range proof is a special case of the aggregated range proof from section 4.3 with $m = 1$. This is therefore a direct corollary of Theorem 2. □

4.2. Logarithmic Range Proof

Finally, we can describe the efficient range proof that uses the improved inner product argument. In the range proof protocol from Section 4.1, $\mathcal{P}$ transmits $\mathbf{l}$ and $\mathbf{r}$, which are already linear in n. We can omit this transfer by using the inner-product argument from Section 3. Note that verifying both (63) and (60) is exactly equivalent to verifying that the witness $\mathbf{l}, \mathbf{r}$ satisfies the inner product relation (1) on public input $(\mathbf{g}, \mathbf{h}', g, P \cdot h^{-\mu}, t)$. We can therefore replace (58) with a transfer of τ_x, μ, t and an execution of an inner product argument. Instead of transmitting $\mathbf{l}$ and $\mathbf{r}$, which has a communication cost of $2 \cdot n$ elements, the inner-product argument requires transmission of just $2 \cdot \lceil \log_2(n) \rceil + 2$ elements. In total the prover sends $2 \cdot \lceil \log_2(n) \rceil + 4$ group elements and 5 elements in $\mathbb{Z}_p$.

4.3. Aggregating Logarithmic Proofs

In many of the range proof applications described in Section 1.2, a single prover needs to perform multiple range proofs at the same time.

For example, a confidential transaction often contains multiple outputs, and in fact, most transactions require a so-called *change output* to send any unspent funds back to the sender. In Provisions [17] the proof of solvency requires the exchange to conduct a range proof for every single account. Given the logarithmic size of the range proof presented in Section 4.2, there is some hope that we can perform a proof for m values which is more efficient than conducting m individual range proofs. In this section, we show that this can be achieved with a slight modification to the proof system from Section 4.1.

Concretely, we present a proof system for the following relation:

$$\big\{(g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^{m \cdot n}, \mathbf{V} \in \mathbb{G}^m; \mathbf{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) : V_j = h_j^\gamma g^{v_j} \wedge v_j \in [0, 2^n - 1] \forall j \in [1, m]\big\} \tag{64}$$

The prover is very similar to the prover for a simple range proof with $n \cdot m$ bits, with the following slight modifications. In line (36), the prover should compute $\mathbf{a}_L \in \mathbb{Z}_p^{n \cdot m}$ such that $\langle \mathbf{a}_{L,[(j-1) \cdot m : j \cdot m]}, \mathbf{2}^n \rangle = v_j$ for all j in $[1, m]$, i.e. $\mathbf{a}_L$ is the concatenation of all of the bits for each v_j.

We adjust $r(X)$ accordingly so that

$$r(X) = \mathbf{y}^{n \cdot m} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s}_R \cdot X) + \sum_{j=1}^m z^{1+j} \cdot \mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n} \tag{65}$$

In the computation of τ_x, we need to adjust for the randomness of each commitment V_j, so that $\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 + \sum_{j=1}^m z^{1+j} \cdot \gamma_j$. Further, $k(y, z)$ is updated to incorporate more cross terms.

$$k(y, z) = -z^2 \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle - \sum_{j=1}^m z^{j+2} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$$

The verification check (61) needs to be updated to include all the V_j commitments.

$$g^t h^{\tau_x} \stackrel{?}{=} g^{k(y,z) + z \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle} \cdot \mathbf{V}^{z^2 \cdot \mathbf{z}^m} \cdot T_1^x \cdot T_2^{x^2}$$

Finally, we change the definition of P (62) such that it is a commitment to the new $\mathbf{r}$.

$$P = AS^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m}} \prod_{j=1}^m \mathbf{h'}_{[(j-1) \cdot m : j \cdot m]}^{z^{j+1} \cdot \mathbf{2}^n}$$

The aggregated range proof which makes use of the inner product argument uses $2 \cdot \lceil \log_2(n \cdot m) \rceil + 4$ group elements and 5 elements in $\mathbb{Z}_p$. Note that the proof size only grows by an additive term of $2 \cdot \log_2(m)$ when conducting multiple range proofs as opposed to a multiplicative factor of m when creating m independent range proofs.

Theorem 2. *The aggregate range proof presented in Section 4.3 has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

4.4. Non-Interactive Proof through Fiat-Shamir

For the purpose of a simpler analysis, the proof was presented as an interactive protocol with a logarithmic number of rounds. The verifier is a public coin verifier, as all the honest verifier's messages are simply random elements from $\mathbb{Z}_p^\star$. It is therefore possible to turn the protocol into a non-interactive protocol that is secure and full zero-knowledge in the random oracle model using the Fiat-Shamir heuristic [62]. All random challenges are replaced by hashes of the transcript up to that point. For instance $y = \mathsf{H}(A, S)$ and $z = \mathsf{H}(A, S, y)$
To avoid a trusted setup we can use such a hash function to generate the public parameters $\mathbf{g}, \mathbf{h}, g, h$ from a small seed. The hash functions needs to map from $\{0,1\}^*$ to $\mathbb{G}\backslash 1$[2]. This also makes it possible to provide a random access into the public parameters. Alternatively, a common random string can be used.

4.5. A Simple MPC Protocol for Bulletproofs

In several of the applications described in Section 1.2, the prover could potentially consist of multiple parties who each want to do a single range proof. For instance, multiple parties may want to create a single joined confidential transaction, where each party knows some of the inputs and outputs and needs to create range proofs for their known outputs. The joint transaction would not only be smaller than the sum of multiple transactions. It would also hide which inputs correspond to which outputs and provide some level of anonymity. These kinds of transactions are called CoinJoin transactions [14]. In Provisions, an exchange may distribute the private keys to multiple servers and split the customer database into separate chunks, but it still needs to produce a single short proof of solvency. Can these parties generate one Bulletproof without sharing the entire witness with each other? The parties could certainly use generic multi-party computation techniques to generate a single proof, but this might be too expensive and incur significant communication costs. This motivates the search for a simple MPC protocol specifically designed for Bulletproofs which requires little modification to the prover and is still efficient. Note that for aggregate range proofs, the inputs of one range proof do not affect the output of another range proof. Given the composable structure of Bulletproofs, it turns out that m parties each having a Pedersen commitment $(V_k)_{k=1}^m$ can generate a single Bulletproof that each V_k commits to a number in some range fixed range. The protocol either uses a constant number of rounds but communication that is linear in both m and the binary encoding of the range, or it uses a logarithmic number of rounds and communication that is only linear in m. We assume for simplicity that m is a power of 2, but the protocol could be easily adapted for other m. We use the same notation as in the aggregate range proof protocol, but use k as an index to denote the kth party's message. That is $A^{(k)}$ is generated just like A but using only the inputs of party k. The MPC protocol works as follows, we assign a set of distinct generators $(\mathbf{g}^{(k)}, \mathbf{h}^{(k)})_{k=1}^m$ to each party and define $\mathbf{g}$ as the interleaved concatenation of all $\mathbf{g}^{(k)}$ such that $g_i = g_{\lceil \frac{i}{m} \rceil}^{((i-1) \mod m+1)}$. Define $\mathbf{h}$ and $\mathbf{h}^{(k)}$ in an analogous way. We first describe the protocol with linear communication. In each of the 3 rounds of the protocol, which correspond to the rounds of the range proof protocol, each party simply generates its part of the proof, i.e. the $A^{(k)}, S^{(k)}; T_1^{(k)}, T_2^{(k)}; \tau_x^{(k)}, \mu^{(k)}, t^{(k)}, \mathbf{l}^{(k)}, \mathbf{r}^{(k)}$ using its inputs and generators. These shares are then sent to a dealer (which could be one of the parties), who simply adds them homomorphically to generate the respective proof component, e.g. $A = \prod_{k=1}^l A^{(k)}$ and $\tau_x = \sum_{k=1}^l \tau_x^{(k)}$. In each round, the dealer generates the challenges using the Fiat-Shamir heuristic and the combined proof components and sends them to each party. Finally, each party sends $\mathbf{l}^{(k)}, \mathbf{r}^{(k)}$ to the dealer who computes $\mathbf{l}, \mathbf{r}$ as the interleaved concatenation of the shares. The dealer runs the inner product argument and generates the final proof. The protocol is complete as each proof component is simply the (homomorphic) sum of each parties' proof components, and the challenges are generated as in the original protocol. It is also secure against honest but curious adversaries as each share constitutes part of a separate zero-knowledge proof. The communication can be reduced by running a second MPC protocol for the inner product argument. The generators were selected in such a way that up to the last $\log_2(l)$ rounds each parties' witnesses are independent and the overall witness is simply the interleaved concatenation of the parties' witnesses. Therefore, parties simply compute $L^{(k)}, R^{(k)}$ in each round and a dealer computes L, R as the homomorphic sum of the shares. The dealer then again generates the challenge and sends it to each party. In the final round the parties send their witness to the dealer who completes Protocol 2. A similar protocol can be used for arithmetic circuits if the circuit is decomposable into separate independent circuits. Constructing an efficient MPC protocol for more complicated circuits remains an open problem.

5. Zero-Knowledge Proof for Arithmetic Circuits

Bootle et al. [7] present an efficient zero-knowledge argument for arbitrary arithmetic circuits using $6\log_2(n)+13$ elements, where n is the multiplicative complexity of the circuit. We can use our improved inner product argument to get a proof of size $2\log_2(n) + 13$ elements, while simultaneously generalizing to include committed values as inputs to the arithmetic circuit. Including committed input wires is important for many applications (notably range proofs) as otherwise the circuit would need to implement a commitment algorithm. Concretely a statement about Pedersen commitments would need to implement the group exponentiation for the group that the commitment is an element of.

2. See [63] for a concrete construction of hash function into an elliptic curve

Following [7], we present a proof for a Hadamard-product relation. A multiplication gate of fan-in 2 has three wires; 'left' and 'right' for the input wires, and 'output' for the output wire. In the relation, $\mathbf{a}_L$ is the vector of left inputs for each multiplication gate. Similarly, $\mathbf{a}_R$ is the vector of right inputs, and $\mathbf{a}_O = \mathbf{a}_L \circ \mathbf{a}_R$ is the vector of outputs. [7] shows how to convert an arbitrary arithmetic circuit with n multiplication gates into a relation containing a Hadamard-product as above, with an additional $Q \leqslant 2 \cdot n$ linear constraints of the form

$$\langle \mathbf{w}_{L,q}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,q}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,q}, \mathbf{a}_O \rangle = c_q$$

for $1 \leqslant q \leqslant Q$, with $\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q} \in \mathbb{Z}_p^n$ and $c_q \in \mathbb{Z}_p$.

We include additional commitments V_i as part of our statement, and give a protocol for a more general relation, where the linear consistency constraints include the openings v_j of the commitments V_j. For simplicity and efficiency we present the scheme with V_i being Pedersen commitments. The scheme can be trivially adapted to work with other additively homomorphic schemes by changing the commitments to $t(X)$ and adapting the verification in line (83).

5.1. Inner-Product Proof for Arithmetic Circuits

As with the range proof we first present a linear proof system where the prover sends two vectors that have to satisfy some inner product relation. In Section 5.2 we show that the inner product relation can be replaced with an efficient inner product argument which yields short proofs for arbitrary circuits where input wires can come from Pedersen commitments. Formally we present a proof system for the following relation.

$$\begin{aligned}&\{(g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n, \mathbf{V} \in \mathbb{G}^m, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n},\\ &\mathbf{W}_V \in \mathbb{Z}_p^{Q \times m}, \mathbf{c} \in \mathbb{Z}_p^Q; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n, \mathbf{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) :\\ &V_j = g^{v_j} h^{\gamma_j}\ \forall j \in [1, m] \wedge \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O\\ &\wedge \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c}\}\end{aligned} \tag{66}$$

Let $\mathbf{W}_V \in \mathbb{Z}_p^{Q \times m}$ be the weights for a commitment V_j. The presented proof system only works for relations where $\mathbf{W}_V$ is of rank m, i.e. the columns of the matrix are all linearly independent. This restriction is minor as we can construct commitments that fulfill these linearly dependent constraints as a homomorphic combination of other commitments. Consider a vector $\mathbf{w}'_V = \mathbf{a} \cdot \mathbf{W}_V \in \mathbb{Z}_p^m$ for a vector of scalars $\mathbf{a} \in \mathbb{Z}_p^Q$ then we can construct commitment $V' = \mathbf{v}^{\mathbf{a} \cdot \mathbf{W}_V}$. Note that if the relation holds then we can conclude that $\langle \mathbf{w}_{L,j}, \mathbf{a}_L \rangle + \langle \mathbf{w}_{R,j}, \mathbf{a}_R \rangle + \langle \mathbf{w}_{O,j}, \mathbf{a}_O \rangle = \langle \mathbf{w}'_V, \mathbf{v} \rangle + \mathbf{c}$. The protocol is presented in Protocol 3. It is split into two parts. In the first part $\mathcal{P}$ commits to $l(X), r(X), t(X)$ in the second part $\mathcal{P}$ convinces $\mathcal{V}$ that the polynomials are well formed and that $\langle l(X), r(X) \rangle = t(X)$.

Theorem 3. *The proof system presented in Protocol 3 has perfect completeness, perfect honest verifier zero-knowledge and computational special soundness.*

Input: $(g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{Z}_p^{Q \times n}$, $\mathbf{W}_V \in \mathbb{Z}_p^{Q \times m}, \mathbf{c} \in \mathbb{Z}_p^Q; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n, \boldsymbol{\gamma} \in \mathbb{Z}_p^m)$
$\mathcal{P}$'s input: $(g, h, \mathbf{g}, \mathbf{h}, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{W}_V, \mathbf{c}; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \boldsymbol{\gamma})$
$\mathcal{V}$'s input: $(g, h, \mathbf{g}, \mathbf{h}, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{W}_V, \mathbf{c})$
Output: $\{\mathcal{V}$ accepts,$\mathcal{V}$ rejects $\}$
$\mathcal{P}$ computes:
$\alpha, \beta, \rho \xleftarrow{\$} \mathbb{Z}_p$
$A_I = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$
$A_O = h^\beta \mathbf{g}^{\mathbf{a}_O} \in \mathbb{G}$
$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$
$S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$
$\mathcal{P} \to \mathcal{V} : A_I, A_O, S$
$\mathcal{V} : y, z \xleftarrow{\$} \mathbb{Z}_p^\star$
$\mathcal{V} \to \mathcal{P} : y, z$
$\mathcal{P}$ and $\mathcal{V}$ compute:
$\mathbf{y}^n = (1, y, y^2, \ldots, y^{n-1}) \in \mathbb{Z}_p^n$
$\mathbf{z}_{[1:]}^{Q+1} = (z, z^2, \ldots, z^Q) \in \mathbb{Z}_p^Q$
$k(y, z) = \langle \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R), \mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L \rangle$
$\mathcal{P}$ computes:
$l(X) = \mathbf{a}_L \cdot X + \mathbf{a}_O \cdot X^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot X + \mathbf{s}_L \cdot X^3 \in \mathbb{Z}_p^n[X]$
$r(X) = \mathbf{y}^n \circ \mathbf{a}_R \cdot X - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot X + \mathbf{W}_O) + \mathbf{y}^n \circ \mathbf{s}_R \cdot X^3 \in \mathbb{Z}_p^n[X]$
$t(X) = \langle l(X), r(X) \rangle = \sum_{i=1}^{6} t_i \cdot X^i \in \mathbb{Z}_p[X]$
$\mathbf{w} = \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O$
$t_2 = \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle - \langle \mathbf{a}_O, \mathbf{y}^n \rangle + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{w} \rangle + k(y, z) \in \mathbb{Z}_p$
$\tau_i \xleftarrow{\$} \mathbb{Z}_p \quad \forall i \in [1, 3, 4, 5, 6]$
$T_i = g^{t_i} h^{\tau_i} \quad \forall i \in [1, 3, 4, 5, 6]$
$\mathcal{P} \to \mathcal{V} : T_1, T_3, T_4, T_5, T_6$

Protocol 3: Part 1: Computing commitments to $l(X), r(X)$ and $t(X)$

5.2. Logarithmic-Sized Protocol

As for the range proof, we can reduce the communication cost of the protocol by using the inner product argument. Concretely transfer (75) is altered to simply τ_x, μ, t and additionally $\mathcal{P}$ and $\mathcal{V}$ engage in an inner product argument on public input $(\mathbf{g}, \mathbf{h}', g, P \cdot h^{-\mu}, t)$. Note that the statement proven is equivalent to the verification equations (85) and (81). The inner product argument has only logarithmic communication complexity and is thus highly efficient. Note that instead of transmitting $\mathbf{l}, \mathbf{r}$ the inner product argument only requires communication of $2 \cdot \lceil \log_2(2 \cdot n) \rceil$ elements instead of $2 \cdot n$. In total the prover sends $2 \cdot \lceil \log_2(n) \rceil + 9$ group elements and 6 elements in $\mathbb{Z}_p$. Using the Fiat-Shamir

$$\mathcal{V}: x \xleftarrow{\$} \mathbb{Z}_p^\star \tag{67}$$
$$\mathcal{V} \rightarrow \mathcal{P}: x \tag{68}$$
$\mathcal{P}$ computes: (69)
$$\mathbf{l} = l(x) \in \mathbb{Z}_p^n \tag{70}$$
$$\mathbf{r} = r(x) \in \mathbb{Z}_p^n \tag{71}$$
$$t = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p \tag{72}$$
$$\tau_x = \sum_{i=1, i\neq 2}^{6} \tau_i \cdot x^i + x^2 \cdot \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_V \cdot \boldsymbol{\gamma} \rangle \in \mathbb{Z}_p \tag{73}$$
$$\mu = \alpha \cdot x + \beta \cdot x^2 + \rho \cdot x^3 \in \mathbb{Z}_p \tag{74}$$
$$\mathcal{P} \rightarrow \mathcal{V}: \tau_x, \mu, t, \mathbf{l}, \mathbf{r} \tag{75}$$
$\mathcal{V}$ computes and checks: (76)
$$h_i' = h_i^{y^{-i+1}} \quad \forall i \in [1, n] \tag{77}$$
$$W_L = \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_L} \tag{78}$$
$$W_R = \mathbf{g}^{\mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R)} \tag{79}$$
$$W_O = \mathbf{h}'^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_O} \tag{80}$$
$$t \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \tag{81}$$
$$g^t h^{\tau_x} \stackrel{?}{=} g^{x^2 \cdot (k(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle)} \cdot \mathbf{V}^{x^2 \cdot (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_V)} \tag{82}$$
$$\cdot T_1^x \cdot \prod_{i=3}^{6} T_i^{(x^i)} \tag{83}$$
$$P = A_I^x \cdot A_O^{(x^2)} \cdot \mathbf{h}'^{-\mathbf{y}^n} \cdot W_L^x \cdot W_R^x \cdot W_O \cdot S^{(x^3)} \tag{84}$$
$$P \stackrel{?}{=} h^\mu \cdot \mathbf{g}^{\mathbf{l}} \cdot \mathbf{h}'^{\mathbf{r}} \tag{85}$$
if all checks succeed: $\mathcal{V}$ accepts (86)
else: $\mathcal{V}$ rejects (87)

Protocol 3: Part 2: Polynomial identity check for $\langle l(x), r(x) \rangle = t(x)$

heuristic as in 4.4 the protocol can be turned into an efficient non interactive proof. We report implementation details and evaluations in Section 6.

Theorem 4. *The arithmetic circuit protocol using the improved inner product argument (Protocol 2) has perfect completeness, statistical zero-knowledge and computational soundness under the discrete logarithm assumption.*

6. Performance

6.1. Theoretical Performance

In Table 1 we give analytical measurements for the proof size of different range proof protocols. We compare both the proof sizes for a single proof and for m proofs for the range $[0, 2^n - 1]$. We compare Bulletproofs against [3] and a Σ-protocol range proof where the proof commits to each bit and then shows that the commitment is to 0 or 1. The table shows that Bulletproofs have a significant advantage when providing multiple range proofs at once. The proof size for the protocol presented in Section 4.3 only grows by an additive logarithmic factor when conducting m range proofs, while all other solutions grow multiplicatively in m.

TABLE 1: Range proof size for m proofs. $m = 1$ is the special case of a single range proof

	m range proofs for range $[0, 2^n - 1]$	
	# $\mathbb{G}$ elements	# $\mathbb{Z}_p$ elements
Σ Protocol [64]	mn	$3mn + 1$
Poelstra et al. [3]	$0.63 \cdot mn$	$1.26 \cdot mn + 1$
Bulletproofs	$2\big(\log_2(n) + \log_2(m)\big) + 4$	5

6.2. Verification Optimizations using Multi-Exponentiation and Batch Verification

In many of the applications discussed in Section 1.2 the verifier's runtime is of particular interest. For example, with confidential transactions every full node needs to check all confidential transactions and all associated range proofs. We therefore now present a number of optimizations for the non-interactive verifier. We present the optimizations for a single range proof but they all carry over to aggregate range proofs and the arithmetic circuit protocol.

Single multi-exponentiation. We can reduce the verifier's cryptographic operations to a single multi-exponentiation of size $2n + 2\log_2(n) + 7$. Notice that the Bulletproofs verifier only performs two checks (60) and (14). The idea is to delay exponentiation until those checks are actually performed. We, therefore, unroll the inner product argument and separately compute with which exponent each generator g_i, h_i is factored into the final g', h'. The resulting protocol is presented below with x_u being the challenge from Protocol 1, and x_j being the challenge from round j of Protocol 2. L_j and R_j are the L, R values from round j of Protocol 2:

$$\pi = \{A, S, T_1, T_2, (L_j, R_j)_{j=1}^{\log(n)} \in \mathbb{G}, \tag{88}$$
$$\tau, t, \mu, a, b \in \mathbb{Z}_p\} \tag{89}$$
Compute challenges from π: (90)
$$\{y, z, x, x_u, (x_j)_{j=1}^{\log_2(n)}\} \tag{91}$$
$$g^{t - k(y,z) + z \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle} h^{\tau_x} \cdot V^{-z^2} \cdot T_1^{-x} \cdot T_2^{-x^2} \stackrel{?}{=} 1 \tag{92}$$
$$b(i,j) = \begin{cases} 1 & \text{if the } j\text{th bit of } i-1 \text{ is } 0 \\ -1 & \text{otherwise} \end{cases} \tag{93}$$
$$\mathbf{l} = (l_1, \ldots, l_n) \in \mathbb{Z}_p^n \tag{94}$$
$$\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_p^n \tag{95}$$
$$l_i = \prod_{j=1}^{\log_2 n} x_j^{b(i,j)} \cdot a - y^{1-i} \cdot z \tag{96}$$
$$r_i = y^{1-i} \cdot \Big(\prod_{j=1}^{\log_2 n} x_j^{-b(i,j)} \cdot b - z^2 \cdot 2^{i-1} \Big) - z \tag{97}$$
$$\mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} g^{x_u \cdot t} h^\mu \cdot A^{-1} S^{-x} \prod_{j=1}^{\log_2(n)} L_j^{-x_j^2} R_j^{-x_j^{-2}} \stackrel{?}{=} 1 \tag{98}$$

We can combine the two multi-exponentiations in line (92) and (98) by using a random value $c \xleftarrow{\$} \mathbb{Z}_p$. This is because if $A^c B = 1$ for a random c then with high probability $A = 1 \wedge B = 1$.

Various algorithms are known to compute the multi-exponentiations (98) and (92) efficiently. As explained in [65], algorithms like Pippenger's [66] perform a number of group operations that scales with $O\frac{n}{\log(n)}$, i.e. sub-linearly. For realistic problem sizes these dominate verification time.

Computing scalars. A further optimization concerns the computation of the l_i and r_i values. Instead of computing $x^{(i)} = \prod_{j=1}^{\log_2 n} x_j^{b(i,j)}$ for each i, we can compute each challenge product using only one multiplication in $\mathbb{Z}_p$ by applying batch division. First we compute $x^{(1)} = (\prod_{j=1}^{\log_2 n} x_j)^{-1}$ to get the first challenge value using a single inversion. Then computing $x^{(2)} = x^{(1)} x_1^2$, $x^3 = x^{(1)} x_2^2$, and for example $x^{(7)} = x^{(3)} x_5^2$. In general in order to compute $x^{(i)}$ we let k be the next lower power of 2 of $i-1$ and compute $x^{(i)} = x^{(i-k)} \cdot x_{k+1}^2$ which takes only one additional multiplication in $\mathbb{Z}_p$ and no inversion. Further, note that the squares of the challenges are computed anyway in order to check equation (98).

Batch verification. A further important optimization concerns the verification of multiple proofs. In many applications described in 1.2 the verifier can verify multiple proofs at once. For example a Bitcoin node receiving a block of transactions needs to verify all transactions and thus range proofs in parallel. As noted above, verification boils down to a large multi-exponentiation. In fact, $2n+2$ of the generators only depend on the public parameters, and only $2\log(n)+5$ are proof-dependent. We can therefore apply batch verification [67] in order to reduce the number of expensive exponentiations. Batch verification is based on the observation that checking $g^x = 1 \wedge g^y = 1$ can be checked by drawing a random scalar α from a large enough domain and checking $g^{\alpha \cdot x + y} = 1$. With high probability over the ladder equation implies that the first two hold but the ladder only uses a single exponentiation. The same trick applies to multi-exponentiations and can save $2n$ exponentiations per additional proof. This is equivalent to the trick that is used for combining multiple exponentiations into one with the difference that the bases are equivalent. Verifying m distinct range proofs of size n now only requires a single multi-exponentiation of size $2n+2+m\cdot(2\cdot\log(n)+5)$ along with $O(m \cdot n)$ scalar operations. Note that this optimization can even be applied for circuits and proofs for different circuits if the same public parameter are used.

Even for a single verification we can take advantage of the fact that most generators are fixed in the public parameters. The verifier can precompute multiples of the generators and fast lookup tables to speed up exponentiations. The same techniques can improve the prover's complexity.

6.3. Implementation and Performance

To evaluate the performance of Bulletproofs in practice we give a reference implementation in C and integrate it into the popular library libsecp256k1 which is used in many cryptocurrency clients. libsecp256k1 uses the elliptic curve secp256k1[3] which has 128 bit security.

In their compressed form, secp256k1 points can be stored as 33 bytes. We use all of the optimizations described above, except the pre-computation of generators. The prover uses constant time operations until the computation of $\mathbf{l}$ and $\mathbf{r}$. By Theorem 1, the inner product argument does not need to hide $\mathbf{l}$ and $\mathbf{r}$ and can therefore use variable time operations. The verifier has no secrets and can therefore safely use variable time operations like the multi-exponentiations.

All experiments were performed on an Intel i7-6820HQ system throttled to 2.00 GHz and using a single thread. Less than 100 MB of memory was used in all experiments. For reference, verifying an ECDSA signature takes 86 μs on the same system. Table 2 shows that in terms of proof size Bulletproofs bring a significant improvement over the 3.8 KB proof size in [3]. A single 64-bit range proof is 688 bytes. An aggregated proof for 32 ranges is still 1 KB whereas 32 proofs from [3] would have taken up 121 KB. The cost to verify a single 64-bit range proof is 3.9 ms but using batch verification of many proofs the amortized cost can be brought down to 470 μs or 5.5 ECDSA verifications. Verifying an aggregated proof for 64 ranges takes 62 ms or 1.9 ms per range. The marginal cost of verifying an additional proof is 2.58 ms or 81 μs per range. This is less than verifying an ECDSA signature, which cannot take advantage of the same batch validation.

To aid future use of Bulletproofs we also implemented Protocol 3 for arithmetic circuits and provide a parser for circuits in the Pinocchio [9] format to the Bulletproofs format. This hooks Bulletproofs up to the Pinocchio toolchain which contains a compiler from a subset of C to the circuit format. To evaluate the implementation we analyse several circuits for hash preimages in Table 3 and Figure 1.

Specifically, a SHA256 circuit generated by jsnark[4] and a Pedersen hash function over an embedded elliptic curve similar to Jubjub[5] are benchmarked. A Bulletproof for knowing a 384-bit Pedersen hash preimage is about 1 KB and takes 69 ms to verify. The marginal cost of verifying a second proof is 4.7 ms. The SHA256 preimage proof is 1.3 KB and takes 832 ms to verify. The marginal cost of verifying additional proofs is just 58 ms. Figure 1 shows that the proving and verification time grow linearly. The batch verification first grows logarithmically and then linearly. For small circuits the logarithmic number of exponentiations dominate the cost while for larger circuit the linear scalar operations do.

3. http://www.secg.org/SEC2-Ver-1.0.pdf
4. See https://github.com/akosba/jsnark.
5. See https://z.cash/technology/jubjub.html.

TABLE 2: Range proofs: performance and proof size

Problem size	Gates	π Size (bytes)	Timing (ms) prove	verify	batch
Range proofs (range × aggregation size)					
8 bit	8	490	7	1.0	0.31
16 bit	16	556	14	1.5	0.35
32 bit	32	622	27	2.5	0.40
64 bit	64	688	54	3.9	0.47
64 bit × 2	128	754	107	6.4	0.57
per range	64	377	54	3.2	0.29
64 bit × 4	256	820	210	10.7	0.73
per range	64	205	53	2.7	0.18
64 bit × 8	512	886	416	19.7	1.02
per range	64	111	52	2.5	0.13
64 bit × 16	1024	952	825	34.0	1.56
per range	64	60	52	2.1	0.10
64 bit × 32	2048	1018	1621	62.2	2.58
per range	64	32	51	1.9	0.08

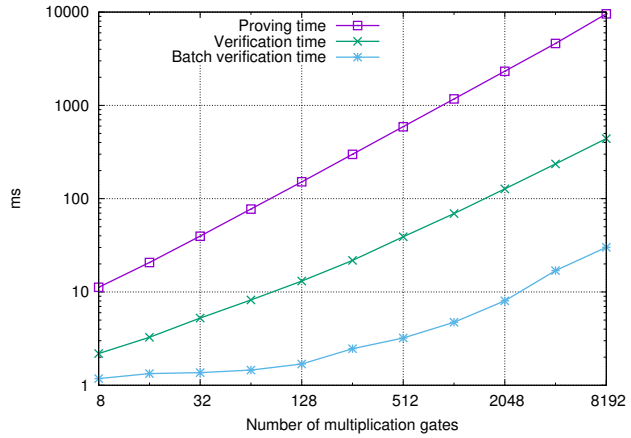
The first 4 instances are n-bit range proofs and the later ones are m aggregated 64-bit proofs and the normalized costs per range. "batch" is the marginal cost of verifying an additional proof.

TABLE 3: Protocol 3: Performance numbers and proof sizes

Problem size	Gates	π Size (bytes)	Timing (ms) prove	verify	batch
Pedersen hash preimage (input size)					
48 bit	128	864	152	13.1	1.69
96 bit	256	928	299	21.8	2.47
192 bit	512	992	599	39.1	3.21
384 bit	1024	1056	1173	69.2	4.74
768 bit	2048	1120	2318	127.3	8.04
1536 bit	4096	1184	4614	235.7	16.95
3072 bit	8192	1248	9570	439.9	30.11
Unpadded SHA256 preimage					
64 byte	25400	1376	36351	832.9	58.44

Bulletproofs for proving knowledge of x s.t. $H(x) = y$ for different sized x's. The first 7 rows are for the Pedersen hash function and the final row is for SHA256. "batch" is the marginal cost of verifying an additional proof.

Figure 1: Timings for arithmetic circuits (Pedersen hashes)



References

[1] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," Cryptology ePrint Archive, Report 2017/1066, 2017, https://eprint.iacr.org/2017/1066.

[2] G. Maxwell, "Confidential transactions," https://people.xiph.org/~greg/confidential_values.txt, 2016.

[3] A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille, "Confidential assets."

[4] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *CRYPTO*, 2013.

[5] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Advances in Cryptology - EUROCRYPT 2013*, 2013, pp. 626–645. [Online]. Available: https://doi.org/10.1007/978-3-642-38348-9_37

[6] E. Ben-Sasson, I. Ben-Tov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," 2018, https://eprint.iacr.org/2018/046.pdf.

[7] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 327–357.

[8] A. Poelstra, "Mimblewimble."

[9] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 238–252.

[10] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," *IEEE Symposium on Security and Privacy*, 2015. [Online]. Available: http://www.jbonneau.com/doc/BMCNKF15-IEEESP-bitcoin.pdf

[11] T. P. Pedersen *et al.*, "Non-interactive and information-theoretic secure verifiable secret sharing." in *Crypto*, vol. 91, no. 7. Springer, 1991, pp. 129–140.

[12] G. Maxwell and A. Poelstra, "Borromean ring signatures," http://diyhpl.us/~bryan/papers2/bitcoin/Borromean%20ring%20signatures.pdf, 2015.

[13] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[14] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world," bitcointalk.org, August 2013.

[15] O. Andreev, "Hidden in Plain Sight: Transacting Privately on a Blockchain," blog.chain.com, 2017.

[16] T. Jedusor, "Mimblewimble," 2016.

[17] G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges (full version)," IACR Cryptology ePrint Archive, Tech. Rep., 2015.

[18] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Crypto*, vol. 1. Springer, 2001, pp. 368–387.

[19] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM, 2001, pp. 116–125.

[20] D. Chaum, "Blind signatures for untraceable payments," in *CRYPTO*, 1982.

[21] J. Groth, "A verifiable secret shuffle of homomorphic encryptions," in *Public Key Cryptography*, vol. 2567. Springer, 2003, pp. 145–160.

[22] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," *Advances in Cryptology–EUROCRYPT 2008*, pp. 379–396, 2008.

[23] S. Bayer and J. Groth, "Efficient zero-knowledge argument for correctness of a shuffle," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2012, pp. 263–280.

[24] G. Wood, "Ethereum: A secure decentralized transaction ledger," http://gavwood.com/paper.pdf, 2014.

[25] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 839–858.

[26] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy." *IACR Cryptology ePrint Archive*, vol. 2017, p. 110, 2017.

[27] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," *Commun. ACM*, 2017.

[28] G. Maxwell, "Zero knowledge contingent payment. 2011," *URl: https://en. bitcoin. it/wiki/Zero_Knowledge_Contingent_Payment (visited on 05/01/2016)*.

[29] R. Canetti, B. Riva, and G. N. Rothblum, "Practical delegation of computation using multiple servers," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 445–454.

[30] B. Bünz, S. Goldfeder, and J. Bonneau, "Proofs-of-delay and randomness beacons in ethereum," *IEEE SECURITY and PRIVACY ON THE BLOCKCHAIN (IEEE S&B)*, 2017.

[31] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains."

[32] S. Micali, "Cs proofs," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE, 1994, pp. 436–453.

[33] J. Kilian and E. Petrank, "An efficient non-interactive zero-knowledge proof system for NP with general assumptions," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 2, no. 38, 1995. [Online]. Available: http://eccc.hpi-web.de/eccc-reports/1995/TR95-038/index.html

[34] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology - EUROCRYPT 2008*, 2008, pp. 415–432. [Online]. Available: https://doi.org/10.1007/978-3-540-78967-3_24

[35] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *IEEE Symposium on Security and Privacy*. IEEE, 2014.

[36] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *Advances in Cryptology - ASIACRYPT 2010*, 2010, pp. 321–340. [Online]. Available: https://doi.org/10.1007/978-3-642-17373-8_19

[37] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Innovations in Theoretical Computer Science 2012*, 2012, pp. 326–349. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090263

[38] ——, "Recursive composition and bootstrapping for SNARKS and proof-carrying data," in *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, 2013, pp. 111–120. [Online]. Available: http://doi.acm.org/10.1145/2488608.2488623

[39] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: nearly practical verifiable computation," *Commun. ACM*, vol. 59, no. 2, pp. 103–112, 2016. [Online]. Available: http://doi.acm.org/10.1145/2856449

[40] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology - EUROCRYPT 2016*, 2016, pp. 305–326. [Online]. Available: https://doi.org/10.1007/978-3-662-49896-5_11

[41] S. Bowe, A. Gabizon, and M. D. Green, "A multi-party protocol for constructing the public parameters of the pinocchio zk-snark," *IACR Cryptology ePrint Archive*, vol. 2017, p. 602, 2017. [Online]. Available: http://eprint.iacr.org/2017/602

[42] J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. K. Jakobsen, "Linear-time zero-knowledge proofs for arithmetic circuit satisfiability," Cryptology ePrint Archive, Report 2017/872, 2017, http://eprint.iacr.org/2017/872.

[43] E. Ben-Sasson, I. Bentov, A. Chiesa, A. Gabizon, D. Genkin, M. Hamilis, E. Pergament, M. Riabzev, M. Silberstein, E. Tromer *et al.*, "Computational integrity with a public random string from quasi-linear pcps," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 551–579.

[44] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner, "Interactive oracle proofs with constant rate and query complexity," in *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, 2017, pp. 40:1–40:15. [Online]. Available: https://doi.org/10.4230/LIPIcs.ICALP.2017.40

[45] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2087–2104.

[46] R. S. Wahby, I. Tzialla, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup."

[47] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 113–122.

[48] S. Nakamoto, "Bitcoin: A peer-to-peer electionic cash system," Unpublished, 2008.

[49] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact e-cash," in *EUROCRYPT*, 2005.

[50] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *IMC*, 2013.

[51] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating User Privacy in Bitcoin," in *Financial Cryptography*, 2013.

[52] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical decentralized coin mixing for Bitcoin," in *ESORICS*, 2014.

[53] "Monero - Private Digital Currency ," https://getmonero.org/.

[54] N. van Saberhagen, "Cryptonote v 2. 0," 2013.

[55] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2003, pp. 398–415.

[56] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 467–482.

[57] T. Sander, "Efficient accumulators without trapdoor extended abstract," *Information and Communication Security*, pp. 252–262, 1999.

[58] J. Camenisch, R. Chaabouni, and A. Shelat, "Efficient protocols for set membership and range proofs," *Advances in Cryptology-ASIACRYPT 2008*, pp. 234–252, 2008.

[59] J. C. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital sinatures (extended abstract)," in *Advances in Cryptology - EUROCRYPT '93*, 1993, pp. 274–285. [Online]. Available: https://doi.org/10.1007/3-540-48285-7_24

[60] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*, 2004, pp. 56–73. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_4

[61] R. Chaabouni, H. Lipmaa, and A. Shelat, "Additive combinatorics and discrete logarithm based range protocols," in *Information Security and Privacy - 15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Proceedings*, 2010, pp. 336–351. [Online]. Available: https://doi.org/10.1007/978-3-642-14081-5_21

[62] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *CCS '93*, 1993, pp. 62–73. [Online]. Available: http://doi.acm.org/10.1145/168588.168596

[63] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 514–532.

[64] R. Cramer and I. Damgård, "Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free?" in *CRYPTO 98*. Springer, 1998, pp. 424–441.

[65] D. J. Bernstein, J. Doumen, T. Lange, and J.-J. Oosterwijk, "Faster batch forgery identification," in *International Conference on Cryptology in India*. Springer, 2012, pp. 454–473.

[66] N. Pippenger, "On the evaluation of powers and monomials," *SIAM Journal on Computing*, vol. 9, pp. 230–250, 1980.

[67] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Advances in Cryptology — EUROCRYPT'98*, K. Nyberg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 236–250.

[68] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," in *Advances in Cryptology - EUROCRYPT 2008*, 2008, pp. 379–396. [Online]. Available: https://doi.org/10.1007/978-3-540-78967-3_22

[69] Y. Lindell, "Parallel coin-tossing and constant-round secure two-party computation," *J. Cryptology*, vol. 16, no. 3, pp. 143–184, 2003. [Online]. Available: https://doi.org/10.1007/s00145-002-0143-7

Appendix A. Zero-Knowledge Arguments of Knowledge

In this paper the common reference string will always be a public key for the Pedersen commitment scheme.

We will consider arguments consisting of three interactive algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, all running in probabilistic polynomial time. These are the common reference string generator $\mathcal{K}$, the prover $\mathcal{P}$, and the verifier $\mathcal{V}$. On input 1^λ, algorithm $\mathcal{K}$ produces a common reference string σ. The transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ depending on whether the verifier rejects, $b = 0$, or accepts, $b = 1$.

Let $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time-decidable ternary relation. Given σ, we call w a witness for a statement u if $(\sigma, u, w) \in \mathcal{R}$, and define the CRS-dependent language

$$\mathcal{L}_\sigma = \{x \mid \exists w : (\sigma, x, w) \in \mathcal{R}\}$$

as the set of statements x that have a witness w in the relation $\mathcal{R}$.

Definition 7 (Argument of Knowledge)**.** *The triple $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is called an* argument of knowledge *for relation $\mathcal{R}$ if it satisfies the following two definitions.*

Definition 8 (Perfect completeness)**.** *$(\mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

$$\mathrm{P}\left[\begin{array}{l|l} (\sigma, u, w) \notin \mathcal{R} \text{ or} & \sigma \leftarrow \mathcal{K}(1^\lambda) \\ \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) \rangle = 1 & (u, w) \leftarrow \mathcal{A}(\sigma) \end{array} \right] = 1$$

Definition 9 (Computational Witness-Extended Emulation)**.** *$(\mathcal{P}, \mathcal{V})$ has* witness-extended emulation *if for all deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for all interactive adversaries $\mathcal{A}$ there exists a negligible function $\mu(\lambda)$ such that*

$$\left| \mathrm{P}\left[\mathcal{A}(tr) = 1 \,\middle|\, \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda), (u, s) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u) \rangle \end{array} \right] - \mathrm{P}\left[\begin{array}{l|l} \mathcal{A}(tr) = 1 & \sigma \leftarrow \mathcal{K}(1^\lambda), \\ \wedge (tr \text{ is accepting} & (u, s) \leftarrow \mathcal{A}(\sigma), \\ \implies (\sigma, u, w) \in \mathcal{R}) & (tr, w) \leftarrow \mathcal{E}^{\mathcal{O}}(\sigma, u) \end{array} \right] \right| \leqslant \mu(\lambda)$$

where the oracle is given by $\mathcal{O} = \langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u) \rangle$, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. We can also define computational *witness-extended emulation by restricting to non-uniform polynomial time adversaries $\mathcal{A}$.*

We use witness-extended emulation to define knowledge-soundness as used for example in [7] and defined in [68], [69]. Informally, whenever an adversary produces an argument which satisfies the verifier with some probability, then there exists an emulator producing an identically distributed argument with the same probability, but also a witness. The value s can be considered to be the internal state of $\mathcal{P}^*$, including randomness. The emulator is permitted to rewind the interaction between the prover and verifier to any move, and resume with the same internal state for the prover, but with fresh randomness for the verifier. Whenever $\mathcal{P}^*$ makes a convincing argument when in state s, $\mathcal{E}$ can extract a witness, and therefore, we have an argument of knowledge of w such that $(\sigma, u, w) \in \mathcal{R}$.

Definition 10 (Public Coin)**.** *An argument $(\mathcal{P}, \mathcal{V})$ is called* public coin *if all messages sent from the verifier to the prover are chosen uniformly at random and independently of the prover's messages, i.e., the challenges correspond to the verifier's randomness ρ.*

An argument is zero knowledge if it does not leak information about w apart from what can be deduced from the fact that $(\sigma, x, w) \in \mathcal{R}$. We will present arguments that have special honest-verifier zero-knowledge. This means that given the verifier's challenge values, it is possible to efficiently simulate the entire argument without knowing the witness.

Definition 11 (Perfect Special Honest-Verifier Zero-Knowledge)**.** *A public coin argument $(\mathcal{P}, \mathcal{V})$ is a* perfect special honest verifier zero knowledge (SHVZK) *argument for R if there exists a probabilistic polynomial time simulator $\mathcal{S}$*

such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{l|l} (\sigma,u,w)\in\mathcal{R} & \sigma\leftarrow\mathcal{K}(1^\lambda),(u,w,\rho)\leftarrow\mathcal{A}(\sigma),\\ \text{ and } \mathcal{A}(tr)=1 & tr\leftarrow\langle\mathcal{P}(\sigma,u,w),\mathcal{V}(\sigma,u;\rho)\rangle\end{array}\right]$$

$$=\Pr\left[\begin{array}{l|l} (\sigma,u,w)\in\mathcal{R} & \sigma\leftarrow\mathcal{K}(1^\lambda),(u,w,\rho)\leftarrow\mathcal{A}(\sigma),\\ \text{ and } \mathcal{A}(tr)=1 & tr\leftarrow\mathcal{S}(u,\rho)\end{array}\right]$$

where ρ is the public coin randomness used by the verifier.

We now define range proofs, which are proofs that the prover knows an opening to a commitment, such that the committed value is in a certain range. Range proofs can be used to show that an integer commitment is to a positive number or that two homomorphic commitments to elements in a field of prime order will not overflow modulo the prime when they are added together.

Definition 12 (Zero-Knowledge Range Proof)**.** *Given a commitment scheme* $(\mathrm{Setup},\mathrm{Com})$ *over a message space* M_{pp} *which is a set with a total ordering, a Zero-Knowledge Range Proof is a protocol for the following relation:* $\{(1^\lambda,\mathrm{pp},\mathbf{com}\in\mathsf{C}_{\mathrm{pp}},l,r\in\mathsf{M}_{\mathrm{pp}};\mathrm{pp}=\mathrm{Setup}(1^\lambda)\wedge\mathbf{com}=\mathrm{Com}(x;r)\wedge x\geqslant l\wedge x\leqslant r\}$

Appendix B. A General Forking Lemma

We briefly describe the forking lemma of [7].

Suppose that we have a $(2\mu+1)$-move public-coin argument with μ challenges, $x_1,\ldots,x_\mu$ in sequence. Let $n_i\geqslant 1$ for $1\leqslant i\leqslant\mu$. Consider $\prod_{i=1}^{\mu}n_i$ accepting transcripts with challenges in the following tree format. The tree has depth μ and $\prod_{i=1}^{\mu}n_i$ leaves. The root of the tree is labeled with the statement. Each node of depth $i<\mu$ has exactly n_i children, each labeled with a distinct value of the ith challenge x_i.

This can be referred to as an $(n_1,\ldots,n_\mu)$-tree of accepting transcripts. Given a suitable tree of accepting transcripts, one can compute a valid witness for our inner-product argument, range proof, and argument for arithmetic circuit satisfiability. This is a natural generalization of special-soundness for Sigma-protocols, where $\mu=1$ and $n=2$. Combined with Theorem 5, this shows that the protocols have witness-extended emulation, and hence, the prover cannot produce an accepting transcript unless they know a witness. For simplicity in the following lemma, we assume that the challenges are chosen uniformly from $\mathbb{Z}_p$ where $|p|=\lambda$, but any sufficiently large challenge space would suffice.

Theorem 5 (Forking Lemma, [7])**.** *Let $(\mathcal{K},\mathcal{P},\mathcal{V})$ be a $(2\mu+1)$-move, public coin interactive protocol. Let $\mathcal{E}$ be a witness extraction algorithm that always succeeds in extracting a witness from an $(n_1,\ldots,n_\mu)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{\mu}n_i$ is bounded above by a polynomial in the security parameter λ. Then $(\mathcal{K},\mathcal{P},\mathcal{V})$ has witness-extended emulation.*

Appendix C. Proof of Theorem 1

Proof. Perfect completeness follows directly because Protocol 1 converts an instance for relation (1) into an instance for relation (2). Protocol 2 is trivially complete. For witness extended emulation we show that there exists an efficient extractor $\mathcal{E}$ as defined in Theorem 5. First we show how to construct an extractor $\mathcal{E}_1$ for Protocol 2 which on input $(\mathbf{g},\mathbf{h},u,P)$, either extracts a witness $\mathbf{a},\mathbf{b},c$ such that the relation holds, or discovers a non-trivial discrete logarithm relation between $\mathbf{g},\mathbf{h},u$. First note that the hardness of computing a discrete log relation between $\mathbf{g}',\mathbf{h}',u$ implies the hardness of computing one between $\mathbf{g},\mathbf{h},u$ as defined in Protocol 2. We will, therefore, use a recursive argument showing that in each step we either extract a witness or a discrete log relation. If $n=|\mathbf{g}|=1$, then the prover reveals the witness and the relation can simply be checked directly. Now, we show for each recursive step that on input $(\mathbf{g},\mathbf{h},u,P)$, we can efficiently extract a witness $\mathbf{a},\mathbf{b}$ or a non-trivial discrete logarithm relation between $\mathbf{g},\mathbf{h},u$. The extractor runs the prover to get L and R. Then, using 3 different challenges x_1,x_2,x_3, the extractor obtains $\mathbf{a}_{(1)},\mathbf{b}_{(1)},\ldots,\mathbf{a}_{(3)},\mathbf{b}_{(3)}$, such that

$$L^{x_i^2}PR^{x_i^{-2}}=\mathbf{g}^{\mathbf{a}_{(i)}}\mathbf{h}^{\mathbf{b}_{(i)}}u^{\langle\mathbf{a}_{(i)},\mathbf{b}_{(i)}\rangle}\quad\forall i\in[1,3]\tag{99}$$

Using the same 3 challenge values for x, we compute η_1,η_2,η_3 such that

$$\sum_{i=1}^{3}\eta_i\cdot x^2=1\wedge\sum_{i=1}^{3}\eta_i=0\wedge\sum_{i=1}^{3}\eta_i\cdot x_i^{-2}=0$$

and using these η to construct linear combinations of (99) we can compute $\mathbf{a}_L,\mathbf{b}_L$ and c_L such that $L=\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{b}_L}u^{c_L}$. Repeating this process with different combinations, we can also compute $\mathbf{a}_P,\mathbf{a}_R,\mathbf{b}_P,\mathbf{b}_R,c_P$ and c_R such that

$$R=\mathbf{g}^{\mathbf{a}_R}\mathbf{h}^{\mathbf{b}_R}u^{c_R}$$
$$P=\mathbf{g}^{\mathbf{a}_P}\mathbf{h}^{\mathbf{b}_P}u^{c_P}$$

Given the previously extracted witness $(\mathbf{a}',\mathbf{b}',c')$, and the computed representations of L,P and R for each challenge x, we get

$$\begin{aligned}L^{x^2}PR^{x^{-2}}&=\mathbf{g}^{\mathbf{a}_L\cdot x^2+\mathbf{a}_P+\mathbf{a}_R\cdot x^{-2}}\cdot\mathbf{h}^{\mathbf{b}_L\cdot x^2+\mathbf{b}_P+\mathbf{b}_R\cdot x^{-2}}\\&\quad\cdot u^{c_L\cdot x^2+c_P+c_R\cdot x^{-2}}\\&=(\mathbf{g}_{[:n']}^{x^{-1}}\circ\mathbf{g}_{[n':]}^{x})^{\mathbf{a}'}\cdot(\mathbf{h}_{[:n']}^{x}\circ\mathbf{h}_{[n':]}^{x^{-1}})^{\mathbf{b}'}\cdot u^{c'}\\&=\mathbf{g}_{[:n']}^{\mathbf{a}'\cdot x^{-1}}\mathbf{g}_{[n':]}^{\mathbf{a}'\cdot x}\mathbf{h}_{[:n']}^{\mathbf{b}'\cdot x}\mathbf{h}_{[n':]}^{\mathbf{b}'\cdot x^{-1}}u^{c'}\\\implies\mathbf{a}'\cdot x^{-1}&=\mathbf{a}_{L,[:n']}\cdot x^2+\mathbf{a}_{P,[:n']}+\mathbf{a}_{R,[:n']}\cdot x^{-2}\\\wedge\quad\mathbf{a}'\cdot x&=\mathbf{a}_{L,[n':]}\cdot x^2+\mathbf{a}_{P,[n':]}+\mathbf{a}_{R,[n':]}\cdot x^{-2}\\\wedge\quad\mathbf{b}'\cdot x&=\mathbf{b}_{L,[:n']}\cdot x^2+\mathbf{b}_{P,[:n']}+\mathbf{b}_{R,[:n']}\cdot x^{-2}\\\wedge\quad\mathbf{b}'\cdot x^{-1}&=\mathbf{b}_{L,[n':]}\cdot x^2+\mathbf{b}_{P,[n':]}+\mathbf{b}_{R,[n':]}\cdot x^{-2}\\\wedge\quad c'&=c_L\cdot x^2+c_P+c_R\cdot x^{-2}\end{aligned}$$

If the implications do not hold, we directly obtain a non-trivial discrete logarithm relation between the generators

$(g_1, \ldots, g_n, h_1, \ldots, h_n, u)$. If the implications do hold, we can deduce that the following two equalities hold.

$$\begin{aligned}&\mathbf{a}_{L,[:n']} \cdot x^3 + (\mathbf{a}_{P,[:n']} - \mathbf{a}_{L,[n':]}) \cdot x \\ &+(\mathbf{a}_{R,[:n']} - \mathbf{a}_{P,[n':]}) \cdot x^{-1} - \mathbf{a}_{R,[n':]} \cdot x^{-3} = 0\end{aligned} \tag{100}$$

$$\begin{aligned}&\mathbf{b}_{L,[n':]} \cdot x^3 + (\mathbf{b}_{P,[n':]} - \mathbf{b}_{L,[:n']}) \cdot x \\ &+(\mathbf{b}_{R,[n':]} - \mathbf{b}_{P,[:n']}) \cdot x^{-1} - \mathbf{b}_{R,[:n']} \cdot x^{-3} = 0\end{aligned} \tag{101}$$

The equalities (100) and (101) hold for all 3 challenges x_1, x_2, x_3. They would hold for all challenges x if and only if

$$\mathbf{a}_{L,[:n']} = \mathbf{a}_{R,[n':]} = \mathbf{b}_{R,[:n']} = \mathbf{b}_{L,[n':]} = 0 \tag{102}$$

$$\wedge \quad \mathbf{a}_{L,[n':]} = \mathbf{a}_{P,[:n']} \wedge \mathbf{a}_{R,[:n']} = \mathbf{a}_{P,[n':]} \tag{103}$$

$$\wedge \quad \mathbf{b}_{L,[:n']} = \mathbf{b}_{P,[n':]} \wedge \mathbf{b}_{R,[n':]} = \mathbf{b}_{P,[:n']} \tag{104}$$

If, however, we find a value of $\mathbf{a}_L$,$\mathbf{a}_P$,$\mathbf{a}_R$,$\mathbf{b}_L$,$\mathbf{b}_P$, or $\mathbf{b}_R$ which is not of this form, we can directly compute one of the given form, using two of the three challenges and the equations (100) and (101). This however, directly results in two distinct representations of L, P or R, which yields a non-trivial discrete logarithm relation.

Finally, using the fact that $\mathbf{a}' = \mathbf{a}_{P,[:n']} \cdot x + \mathbf{a}_{P,[n':]} \cdot x^{-1}$ and $\mathbf{b}' = \mathbf{b}_{P,[:n']} \cdot x^{-1} + \mathbf{b}_{P,[n':]} \cdot x$ we see for all 3 challenges that:

$$\begin{aligned}\langle \mathbf{a}', \mathbf{b}' \rangle &= c' \\ &= c_L \cdot x^2 + c + c_R \cdot x^{-2} \\ &= \langle \mathbf{a}_{P,[:n']} \cdot x + \mathbf{a}_{P,[n':]} \cdot x^{-1}, \mathbf{b}_{P,[:n']} \cdot x^{-1} + \mathbf{b}_{P,[n':]} \cdot x \rangle \\ &= \langle \mathbf{a}_{P,[:n']}, \mathbf{b}_{P,[n':]} \rangle \cdot x^2 + \langle \mathbf{a}_{P,[:n']}, \mathbf{b}_{P,[:n']} \rangle \\ &+ \langle \mathbf{a}_{P,[n':]}, \mathbf{b}_{P,[n':]} \rangle + \langle \mathbf{a}_{P,[n':]}, \mathbf{b}_{P,[:n']} \rangle \cdot x^{-2}\end{aligned}$$

These equalities only hold for three distinct challenges if $\langle \mathbf{a}_P, \mathbf{b}_P \rangle = c$. Therefore, the extractor either extracts discrete logarithm relations between the generators or the witness $(\mathbf{a}_C, \mathbf{b}_C)$. Using the generalized forking lemma from [7] (see Theorem 5) we can see that the extractor uses $3^{\lceil \log_2(n) \rceil} \leqslant n^2$ challenges in total and thus runs in expected polynomial time in n and λ.

We now show that using Protocol 1 we can construct an extractor $\mathcal{E}$ that extracts a valid witness for relation (2). The extractor uses the extractor $\mathcal{E}_1$ of Protocol 2. On input $(\mathbf{g}, \mathbf{h}, u, P, c)$ $\mathcal{E}$ runs the prover with on a challenge x and uses the extractor $\mathcal{E}_1$ to get witness $\mathbf{a}, \mathbf{b}$ such that: $P \cdot u^{x \cdot c} = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^{x \cdot \langle \mathbf{a}, \mathbf{b} \rangle}$. Forking the $\mathcal{P}$, supplying him with a challenge x' and rerunning the extractor $\mathcal{E}_1$ yields a second witness $(\mathbf{a}', \mathbf{b}')$. Again the soundness of Protocol 2 implies that $P \cdot u^{x' \cdot c} = \mathbf{g}^{\mathbf{a}'} \mathbf{h}^{\mathbf{b}'} u^{x' \cdot \langle \mathbf{a}', \mathbf{b}' \rangle}$. From the two witnesses, we can compute:

$$u^{(x-x') \cdot c} = \mathbf{g}^{\mathbf{a}-\mathbf{a}'} \mathbf{h}^{\mathbf{b}-\mathbf{b}'} u^{x \cdot \langle \mathbf{a}, \mathbf{b} \rangle - x' \cdot \langle \mathbf{a}', \mathbf{b}' \rangle}$$

Unless $\mathbf{a} = \mathbf{a}'$ and $\mathbf{b} = \mathbf{b}'$ we get a not trivial discrete log relation between $\mathbf{g}, \mathbf{h}$ and u. Otherwise we get $u^{(x-x') \cdot c} = u^{(x-x') \cdot \langle \mathbf{a}, \mathbf{b} \rangle} \implies c = \langle \mathbf{a}, \mathbf{b} \rangle$. Thus, $(\mathbf{a}, \mathbf{b})$ is a valid witness for relation (2). Since $\mathcal{E}$ forks the prover once, and uses the efficient extractor $\mathcal{E}_1$ twice, it is also efficient. This shows that the protocol has witness extended emulation. □

Appendix D.
Proof of Theorem 2

Proof. Perfect completeness follows from the fact that $t_0 = k(y,z) + z \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle + z^2 \cdot \langle \mathbf{z}^m, \mathbf{v} \rangle$ for all valid witnesses. To prove perfect honest-verifier zero-knowledge we construct a simulator that produces a distribution of proofs for a given statement $(g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^{n \cdot m}, \mathbf{V} \in \mathbb{G}^m)$ that is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. The simulator chooses all proof elements and challenges uniformly at random from their respective domains or computes them directly as described in the protocol. S and T_1 are computed according to the verification equations, i.e.:

$$S = (h^{-\mu} \cdot A \cdot \mathbf{g}^{-z\mathbf{l}} \cdot \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} - \mathbf{r}} \prod_{j=1}^{m} \mathbf{h'}_{[(j-1) \cdot m : j \cdot m]}^{z^{j+1} \cdot \mathbf{2}^n})^{-x^{-1}}$$

$$T_1 = (h^{-\tau_x} g^{k(y,z) + z \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle - t} \cdot \mathbf{V}^{z^2 \cdot \mathbf{z}^m} \cdot T_2^{x^2})^{-x^{-1}}$$

Finally, the simulator runs the inner-product argument with the simulated witness $(\mathbf{l}, \mathbf{r})$. All elements in the proof are either independently randomly distributed or their relationship is fully defined by the verification equations. The inner product argument remains zero knowledge as we can successfully simulate the witness, thus revealing the witness or leaking information about it does not change the zero-knowledge property of the overall protocol. The simulator runs in time $O(\mathcal{V} + \mathcal{P}_{\mathsf{InnerProduct}})$ and is thus efficient.

In order to prove special soundness, we construct an extractor $\mathcal{E}$ as follows. The extractor $\mathcal{E}$ runs the prover with n different values of y, $(Q+1)$ different values of z, and 7 different values of the challenge x. This results in $14 \cdot (Q+1) \cdot n$ valid proof transcripts. The extractor $\mathcal{E}$ first runs the extractor $\mathcal{E}_{\mathsf{InnerProduct}}$ for the inner-product argument to extract a witness $\mathbf{l}, \mathbf{r}$ to the inner product argument such that $\mathbf{g}^{\mathbf{l}} \mathbf{h}^{\mathbf{r}} = P \wedge \langle \mathbf{l}, \mathbf{r} \rangle = t$. Using this witness and 3 valid transcripts with different x challenges, $\mathcal{E}$ can compute linear combinations of (63) in order to extract $\alpha, \rho, \mathbf{a}_L, \mathbf{a}_R, \mathbf{s}_L, \mathbf{s}_R$ such that $A = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$, as well as $S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$.

If for any other set of challenges (x, y, z) the extractor can compute a different representation of A or S, then this yields a non-trivial discrete logarithm relation between independent generators $h, \mathbf{g}, \mathbf{h}$ which contradicts the discrete logarithm assumption.

Using these representations of A and S, as well as $\mathbf{l}$ and $\mathbf{r}$, we then find that for all challenges x, y and z

$$\begin{aligned}\mathbf{l} &= \mathbf{a}_L - z \cdot \mathbf{1}^{n \cdot m} r + \mathbf{s}_L \cdot x \\ \mathbf{r} &= \mathbf{y}^{n \cdot m} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s}_R \cdot x) \\ &+ \sum_{j=1}^{m} z^{1+j} \cdot 0^{(j-1) \cdot n} || \mathbf{2}^n || 0^{(m-j) \cdot n}\end{aligned}$$

If these equalities do not hold for all challenges and $\mathbf{l}, \mathbf{r}$ from the transcript, then we have two distinct representations of the same group element using a set of independent generators. This would be a non-trivial discrete logarithm relation.

For given values of y and z, the extractor $\mathcal{E}$ now takes 3 transcripts with different x's and uses linear combinations of equation (61) to compute $\tau_1, \tau_2, t_1, t_2, v, \gamma$ such that

$$T_1 = g^{t_1} h^{\tau_1} \wedge T_2 = g^{t_1} h^{\tau_2} \wedge g^v h^\gamma = \prod_{j=1}^{m} \mathbf{h}'^{z^{j+1} \cdot \mathbf{2}^n}_{[(j-1)\cdot m : j \cdot m]}$$

Repeating this for m different z challenges, we can compute $(v_j, \gamma_j)_{j=1}^m$ such that $g^{v_j} h^{\gamma_j} = V_j \; \forall j \in [1, m]$. If for any transcript $k(y,z) + z \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle + \sum_{j=1}^m z^{j+2} \cdot \langle \mathbf{v}_j, \mathbf{2}^n \rangle + t_1 \cdot x + t_2 \cdot x^2 \neq t$ then this yields a violation of the binding property of the Pedersen commitment, i.e. a discrete log relation between g and h. If not, then for all y, z challenges and 3 distinct challenges $X = x_j, j \in [1, 3]$:

$$\sum_{i=0}^{2} t_i \cdot X^i - p(X) = 0$$

with $t_0 = k(y,z) + z \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle + \sum_{j=1}^m z^{j+2} \cdot \langle \mathbf{v}_j, \mathbf{2}^n \rangle$ and
$p(X) = \sum_{i=0}^2 p_i \cdot X^i = \langle l(X), r(X) \rangle$. Since the polynomial $t(X) - p(X)$ is of degree 2, but has at least 3 roots (each challenge x_j), it is necessarily the zero polynomial, i.e. $t(X) = \langle l(X), r(X) \rangle$.
Since this implies that $t_0 = p_0$, the following holds for all y, z challenges:

$$\begin{array}{l} z \cdot \langle \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \rangle + \sum_{j=1}^m z^{j+2} \cdot \langle \mathbf{v}_j, \mathbf{2}^n \rangle + k(y,z) \\ = \langle \mathbf{a}_L, \mathbf{y}^{n \cdot m} \circ \mathbf{a}_R \rangle + z \cdot \langle \mathbf{a}_L - \mathbf{a}_R, \mathbf{y}^{n \cdot m} \rangle \\ + \sum_{j=1}^m z^{j+1} \langle \mathbf{a}_{L,[(j-1)\cdot m : j \cdot m]}, \mathbf{2}^n \rangle \\ + k(y,z) \in \mathbb{Z}_p \end{array}$$

Using $n \cdot m$ y challenges and $m + 2$ z challenges we can infer the following.

$$\begin{array}{rl} \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^{n \cdot m} & \in \mathbb{Z}_p^{n \cdot m} \\ \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{n \cdot m} & \in \mathbb{Z}_p^{n \cdot m} \\ v_j = \langle \mathbf{a}_{L,[(j-1)\cdot m : j \cdot m]}, \mathbf{2}^n \rangle & \in \mathbb{Z}_p \forall j \in [1, m] \end{array}$$

The first two equations imply that $\mathbf{a}_L \in \{0,1\}^{n \cdot m}$. The last equation imply that $v_j \in [0, 2^{n-1}]$ for all j. Since $g^{\mathbf{v}} h^\gamma = \mathbf{V}$ we have that $(\mathbf{v}, \boldsymbol{\gamma})$ is valid witness for relation (64). The extractor rewinds the prover $3 \cdot (m+1) \cdot n \cdot O(n^2)$ times. Extraction is efficient and polynomial in λ because $n, m = O(\lambda)$. □

Appendix E. Proof of Theorem 3

Proof. Perfect completeness follows from the fact that

$$\begin{aligned} t_2 &= k(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O \rangle \\ &= k(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \rangle \end{aligned} \quad (105)$$

whenever the prover knows a witness to the relation and is honest. To prove perfect honest-verifier zero-knowledge we construct a simulator that produces a distribution of proofs for a given statement

$$\left(\begin{array}{l} g, h \in \mathbb{G}, \mathbf{g}, \mathbf{h} \in \mathbb{G}^n, \mathbf{V} \in \mathbb{G}^m, \\ (\mathbf{w}_{L,q}, \mathbf{w}_{R,q}, \mathbf{w}_{O,q})_{q=1}^Q \in \mathbb{Z}_p^{n \times 3}, (\mathbf{w}_{V,q})_{q=1}^Q \in \mathbb{Z}_p^m, \mathbf{c} \in \mathbb{Z}_p^Q \end{array} \right)$$

that is indistinguishable from valid proofs produced by an honest prover interacting with an honest verifier. The simulator acts as followZ:

$$x, y, z, \mu, \tau_x \xleftarrow{\$} \mathbb{Z}_p \quad (106)$$

$$\mathbf{l}, \mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^n \quad (107)$$

$$t = \langle \mathbf{l}, \mathbf{r} \rangle \quad (108)$$

$$A_I, A_O \xleftarrow{\$} \mathbb{G} \quad (109)$$

$$S = \left(\begin{array}{l} A_I^x \cdot A_O^{x^2} \cdot \mathbf{g}^{-\mathbf{l}} \mathbf{h}'^{-\mathbf{y}^n - \mathbf{r}} \cdot \\ x W_L^x \cdot W_R^x \cdot h^{-\mu} \end{array} \right)^{-x^{-3}} \quad (110)$$

$$T_3, T_4, T_5, T_6 \xleftarrow{\$} \mathbb{G} \quad (111)$$

$$T_1 = \left(\begin{array}{l} h^{-\tau_x} g^{x^2 \cdot (k(y,z) + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{c} \rangle) - t} \\ \cdot \mathbf{V}^{x^2 \cdot (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_V)} \cdot \prod_{i=3}^6 T_i^{x^i} \end{array} \right)^{-x^{-1}} \quad (112)$$

$$\text{Output: } (A_I, A_O, S; y, z; T_1, (T_i)_3^6; x; \tau_x, \mu, t, \mathbf{l}, \mathbf{r}) \quad (113)$$

The values $A_I, A_O, \mathbf{l}, \mathbf{r}, \mu, \tau_x$ produced by an honest prover interacting with an honest verifier are random independent elements, i.e. if $\mathbf{s}, \rho, \alpha, \tau_1, (\tau_i)_3^6, \rho$ as well as x, y, z are chosen independently and randomly. t is the inner product of $\mathbf{l}, \mathbf{r}$ as in any verifying transcript. The simulated S is fully defined by equations (85). The honestly produced T are perfectly hiding commitments and as such random group elements. Their internal relation given t and τ_x is fully defined by equation (83), which is ensured by computing T_1 accordingly. Therefore, the transcript of the proof is identically distributed to an honestly computed proof with uniformly selected challenges. The simulator runs in time $O(\mathcal{V})$ and is thus efficient.
In order to prove special soundness we construct an extractor $\mathcal{E}$ as follows. The $\mathcal{E}$ runs the prover with n different y, $(Q+1)$ different z and 7 different x challenges. This results in $14 \cdot (Q+1) \cdot n$ valid proof transcripts. $\mathcal{E}$ takes 3 valid transcripts for $x = x_1, x_2, x_3$ and fixed y and z. From the transmitted $\mathbf{l}, \mathbf{r}, t$ for each combination of challenges, $\mathcal{E}$ can compute η_1, η_2, η_3 such that

$$\sum_{i=1}^3 \eta_i \cdot x_i = 1 \wedge \sum_{i=1}^3 \eta_i \cdot x^2 = \sum_{i=1}^3 \eta_i \cdot x_i^3 = 0$$

Using these η's to compute linear combinations of equation (85), $\mathcal{E}$ computes $\alpha \in \mathbb{Z}_p, \mathbf{a}_L, \mathbf{a}_R \in \mathbb{Z}_p^n$ such that $h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} = A_I$. If for any other set of challenges (x, y, z) the extractor can compute a different $\alpha', \mathbf{a}'_L, \mathbf{a}'_R$ such that $h^{\alpha'} \mathbf{g}^{\mathbf{a}'_L} \mathbf{h}^{\mathbf{a}'_R} = A_I = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$, then this yields a non-trivial discrete log relation between independent generators $h, \mathbf{g}, \mathbf{h}$ which contradicts the discrete log assumption. Similarly, the extractor can use the same challenges and Equation (85) to compute unique $\beta, \rho \in \mathbb{Z}_p, \mathbf{a}_{O,L}, \mathbf{a}_{O,R}, \mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_p^n$ such that $h^\beta \mathbf{g}^{\mathbf{a}_{O,L}} \mathbf{h}^{\mathbf{a}_{O,R}} = A_O$ and $h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} = S$.
Using Equation (85), we can replace A_I, A_O, S with the

computed representations and read $\mathbf{l}, \mathbf{r}, t$ from the transcripts. We then find that for all challenges x, y, z:

$$\mathbf{l} = \mathbf{a}_L \cdot x + \mathbf{a}_{O,L} \cdot x^2 + \mathbf{y}^{-n} \circ (\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_R) \cdot X + \mathbf{s}_L \cdot x^3$$
$$\mathbf{r} = \mathbf{y}^n \circ \mathbf{a}_R \cdot x - \mathbf{y}^n + \mathbf{z}_{[1:]}^{Q+1} \cdot (\mathbf{W}_L \cdot x + \mathbf{W}_O) + \mathbf{y}^n \circ \mathbf{a}_{O,R} \cdot x^2 + \mathbf{y}^n \circ \mathbf{s}_R \cdot x^3$$
$$t = \langle \mathbf{l}, \mathbf{r} \rangle$$

If these equalities do not hold for all challenges and $\mathbf{l}, \mathbf{r}$ from the transcript, then we have two distinct representations of the same group element using a set of independent generators. This would be a non-trivial discrete log relation. We now show that t_2 indeed has the form described in (105). For a given y, z the extractor takes 7 transcripts with different x's and uses linear combinations of equation (83) to compute $(\tau_i, t_i), i \in [1, 3, \ldots, 6]$ such that $T_i = g^{t_i} h^{\tau_i}$. Note that the linear combinations have to cancel out the other $T_i^{x^i}$ terms as well as $(\mathbf{v}^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_V})^{x^2}$. Using these (τ_i, t_i) we can compute v, γ such that $g^v h^\gamma = \mathbf{V}^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_V}$. Repeating this for m different z challenges, we can compute $(v_j, \gamma_j)_{j=1}^m$ using linear combinations of $g^v h^\gamma = \mathbf{V}^{\mathbf{z}_{[1:]}^{Q+1} \cdot \mathbf{W}_V}$ such that $g^{v_j} h^{\gamma_j} = V_j \forall j \in [1, m]$. This will however only succeed if the weight vectors $\mathbf{w}_{V,j}$ are linearly independent, i.e if the matrix $\mathbf{W}_V$ has rank m. This necessarily implies that $Q \geqslant m$. If for any transcript $t_1 \cdot x + \sum_{i=3}^6 t_i \cdot x^i + x^2 \cdot (\langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \rangle + k(y, z)) \neq t$ then this yields a violation of the binding property of the Pedersen commitment, i.e. a discrete log relation between g and h. If not, then for all y, z challenges and 7 distinct challenges $x = x_j, j \in [1, 7]$:

$$\sum_{i=1}^{6} t_i \cdot x - p(x) = 0 \tag{114}$$

with $t_2 = \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \rangle + k(y, z)$ and $p(x) = \sum_{i=1}^6 p_i \cdot x^i = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$. Since the polynomial $t(x) - p(x)$ is of degree 6, but has at least 7 roots (each challenge x_j), it is necessarily the zero polynomial, i.e. $t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$. Finally, we show that this equality implies that we can extract a witness $(\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{Z}_p^n, \mathbf{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m)$ which satisfies the relation.

The quadratic coefficient of p is:

$$p_2 = \langle \mathbf{a}_L, \mathbf{y}^n \circ \mathbf{a}_R \rangle - \langle \mathbf{a}_{O,L}, \mathbf{y}^n \rangle + \langle \mathbf{z}_{[1:]}^{Q+1}, \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_{R,q} \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_{O,L} \rangle + k(y, z) \in \mathbb{Z}_p$$

The polynomial equality implies that any challenge y, z, $p_2 = t_2$. Using a fixed y and $(Q+1)$ different z challenges we can infer that all coefficients of $p_2(z) - t_2(z)$ have to be zero. Using n different y challenges, i.e. $n \cdot (Q+1)$ total transcripts we can infer the following equalities:

$$\mathbf{a}_L \circ \mathbf{a}_R - \mathbf{a}_{O,L} = \mathbf{0}^n \in \mathbb{Z}_p^n \tag{115}$$
$$\mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_{O,L} = \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \in \mathbb{Z}_p^Q \tag{116}$$

From equation (115) we can directly infer that $\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_{O,L}$. Equations (116) are exactly the linear constraints on the circuit gates.

Defining $\mathbf{a}_O = \mathbf{a}_{O,L}$, we can conclude that $(\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{v}, \boldsymbol{\gamma})$ is indeed a valid witness. The extractor rewinds the prover $14 \cdot (Q+1) \cdot n$ times. Extraction is efficient and polynomial in λ because $n, Q = O(\lambda)$. □

Appendix F.
Proof of Theorem 4

Proof. Completeness follows from the completeness of the underlying protocols. Zero-knowledge follows from the fact that $\mathbf{l}$ and $\mathbf{r}$ can be efficiently simulated, and because the simulator can simply run Protocol 2 given the simulated witness $(\mathbf{l}, \mathbf{r})$. The protocol also has a knowledge-extractor, as the extractor of the range proof can be extended to extract $\mathbf{l}$ and $\mathbf{r}$ by calling the extractor of Protocol 2. The extractor uses $O(n^3)$ valid transcripts in total, which is polynomial in λ if $n = O(\lambda)$. The extractor is thus efficient and either extracts a discrete logarithm relation or a valid witness. However, if the generators $\mathbf{g}, \mathbf{h}, g, h$ are independently generated, then finding a discrete logarithm relation between them is as hard as breaking the discrete log problem. If the discrete log assumption holds in $\mathbb{G}$ then a computationally bounded $\mathcal{P}$ cannot produce discrete-logarithm relations between independent generators. The proof system is therefore computationally sound. □

Acknowledgements

We thank Shashank Agrawal for coming up with the Bulletproof name (short like a bullet with bulletproof security assumptions). We thank Peter Dettmann for pointing out the batch inversion trick. We thank Sean Bowe for various optimizations applicable to arithmetic circuits for Pedersen hash functions. Further we thank Philip Hayes and the anonymous reviewers for helpful corrections. This work was supported by NSF, DARPA, a grant from ONR, and the Simons Foundation.