Bulletproofs: The State-of-the-art Range Proofs for Confidential Transactions

Suyash Bagad

Supervised Research Exposition Report

Guide: Prof. Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology, Bombay

May 8, 2019

Abstract

Range Proofs are proofs or arguments that a secret value, which has either been encrypted or committed, lies in a particular interval. Although range proofs provide a reliable and elegant method as an essential step in verification of a confidential transaction in zero-knowledge, it is not always communication efficient. Also, most of the existing range proofs also require a trusted setup limiting the scope of it's applicability in the realm of cryptocurrencies and applied cryptography. In this report, we analyse several range proof designs with major emphasis of the state-of-the-art Bulletproofs. We illustrate how the existing range proofs are either communication inefficient or asymptotically efficient but practically large or which would require a trusted setup. Further, we analyse the superiority of Bulletproofs over the existing range proofs considering the factors mentioned above.

Bulletproofs is a newly proposed short non-interactive zero-knowledge proof protocol which does not require a trusted setup. They are very well suited for efficient range proofs on committed values. For instance, consider $x \in \mathbb{Z}$ is committed using some commitment scheme, then Bulletproofs enable proving that $x \in [0, 2^n - 1]$ using only $2\log_2(n) + 9$ group and field elements. Here, Proof verification and generation times are linear in n. Further, they also support aggregation of range proofs, let's say a party wishes to have m commitments for $x_i \in \mathbb{Z}\ \forall i \in \{1, 2, \ldots, m\}$ and would like to prove that $\forall i \in \{1, 2, \ldots, m\},\ x_i \in [0, 2^n - 1]$. Bulletproofs help them do this with just additional $\mathcal{O}(\log(m))$ group elements along with the length of a single proof. Additionally, if we have multiple parties with different individual inputs, they could potentially come up with a single proof without revealing their inputs to each other. Such a multi party computation (MPC) protocol uses either $\mathcal{O}(1)$ in rounds and $\mathcal{O}(n)$ in communication, or $\mathcal{O}(\log(n))$ in rounds and $\mathcal{O}(\log(n))$ in communication.

Bulletproofs only rely on the discrete logarithm assumption but do not demand a trusted setup. It has many applications primarily in the field of cryptocurrencies wherein communication efficiency, proof generation and verification times and a non-trusted setup are very crucial. Bulletproofs would be well-suited for the distributed and non-trusted setup of Blockchain.

Acknowledgement

The Supervised Research Exposition Project was a great learning opportunity for me which helped me delve deep into the intricacies of the beautiful field of Range Proofs for Confidential Transactions and Applied Cryptography. It was a challenging but an overwhelming experience to study several range proof designs ever since their inception to the current state-of-the-art Bulletproofs.

I would like to express my deepest appreciation and gratitude to my guide Prof. Saravanan Vijayakumaran for his guidance and constant supervision as well as for providing necessary information regarding the project. His insightful suggestions and comments time and again have helped me get a much more in-depth understanding of the field. Thanks are also due to Piyush Bagad of IIT, Kanpur for his many useful remarks, Madhura Pawar, PICT, Pune and Pranav Kulkarni of IIT, Bombay for their constant encouragement.

Suyash Bagad

Mumbai
May 3, 2019

Contents

List of Figures

List of Tables

1 Introduction

The rise of Bitcoin since 2009 has led to ever-increasing interest in cryptocurrencies. There has been significant destruction in the cryptocurrency industry with the emergence of new coins based on the Blockchain model [Fed+18]. Consequently, there has been growing research in realm of cryptocurrencies and their security. One prominent direction of research has been in designing of efficient protocols for proving the validity of cryptocurrency transactions in zero knowledge to the entity known as *miners*. In this work, we analyse the zero knowledge range proofs used in, but not restricted to, confidential transactions. Building on the techniques of Bootle et al. [Jon+16], Bulletproofs - a unique short non-interactive zero-knowledge proof was proposed by Bünz et al. [Bün+17]. We will present a detailed analysis of Bulletproofs and why it is the state-of-the-art developement in the domain of range proofs.

1.1 Background and Motivation

Modern day cryptography is based on the following primary functions [Gar98]:

- *Privacy/confidentiality*: To ensure that no one can read or access the message except the intended receiver
- *Authentication*: Proving one's identity
- *Integrity*: Ensuring that the message intended to be received by the receiver is not altered in the path
- *Non-repudiation*: A protocol for checking if a message was actually generated by the sender
- *Key exchange*: The protocol which determines how key(s) are shared between the sender and the receiver

All of the above functions form an necessary part of a cryptographic system. A well-designed system ensures all of the above functions are taken care of considering the computational bounds for carrying out any protocol.

In a confidential transaction, it is desirable to have the *confidentiality* and *anonymity*. This means that in a valid confidential transaction, the identity of the sender and receiver must be confidential and the amounts transferred must also be hidden. The idea of having such private transactions in digital currencies could be traced back to David Chaum's work on blind signatures [Dav82]. To motivate the need of *confidentiality* and *anonymity* in a digital transaction, we will consider an example in the context of Bitcoin and Blockchain.

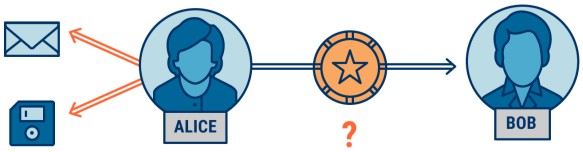
1.1.1 Bitcoin

Bitcoin was proposed by Satoshi Nakamoto in 2009. He defines it as [Nak09]:

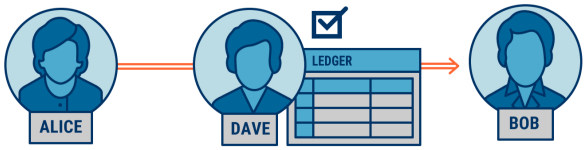
Definition 1.1 (Bitcoin) *Bitcoin is a "peer-to-peer electronic cash system" that allow(s) for online payments to be sent directly from one party to another without going through a trusted financial institution like banks.*

A ledger is a principal book of records which keeps a track of all transactions measured in terms of a monetary unit. Bitcoin is a decentralized, public ledger, decentralized because there is no trusted third party controlling the ledger and public because anyone with bitcoin can participate in the network, receive and send bitcoins, and even hold a copy of this ledger (essentially, the history of all transactions) if they want to. In that sense, the ledger is "non-trusted" and transparent to public. As opposed to such a decentralized ledger setup, traditional banks use centralized ledger system where each bank has it's own ledger visible only to the concerned user.

(a) A physical transaction



(b) A digital transaction



(c) A digital transaction with ledger

(d) A decentralized ledger

Figure 1: Understanding decentralized ledger

In reference to figure[1] 1 , in a physical transaction (figure 1()), the Alice hands Bob a physical arcade coin. Bob now has one coin, and Alice has zero. The transaction is complete. If the same transaction is to be performed digitally (figure 1(a)), Alice would send a string of some bits (corresponding to the desired amount) to Bob. Now, since it's just a string of bits, what's the guarantee that Alice won't reproduce that same string of bits in her mail or hard-disk? If this happens, it would mean that although Alice sent Bob the amount, Alice still possesses the same amount. This clearly is a violation. To tackle this, we must have a ledger or a record of all transactions between Alice and Bob. When Alice gives Bob the digital token, the ledger records the transaction (figure 1(b)). Bob has the token, and Alice does not. Let's call the one who maintains this ledger as Dave. This setup assumes that Dave is a trusted middleman for maintaining ledger. Now, what if Alice bribes Dave to erase her transaction? What if Dave decides to charge a fee that neither Alice or Bob want to pay? What happens when Alice and Bob cannot trust the third party? This results in a decentralized and public ledger system (figure 1(d)). When a lot of people have a copy of the same ledger, it becomes very difficult to tamper the system and cheat. This works because everyone is holding a copy of the same digital ledger and more the number of trusted people holding this ledger, the stronger it becomes.

1.1.2 Blockchain & Privacy

In order to ensure that such a decentralized and public ledger is secure, Bitcoin uses a technology called *Blockchain*. Blockchain technology offers a way for untrusted parties to reach a consensus on a common digital history. The World Bank defines Blockchain as follows [NKG17]:

Definition 1.2 (Blockchain) *A 'blockchain' is a particular type of data structure used in some distributed ledgers which stores and transmits data in packages called "blocks" that are connected to each other in a digital 'chain'.*

Blockchains employ cryptographic algorithms to record and synchronize data across a network in an unchangeable manner. The *state* of the Blockchain is the current status of the ledger visible to public. In case of Bitcoin, any independent observer can verify the state of the blockchain as well as the validity of all the transactions on the ledger. This is a *serious* limitation for Bitcoin and could possibly prohibit many use cases. As an example, if employees of a company were to receive their salaries in bitcoin, would they agree if their salaries were published on the public blockchain? This naturally demands for a transaction system which withholds *anonymity* and *confidentiality*.

1.1.3 Why range proofs?

Confidential transactions in setup like that of Bitcoin have an entity known as *miner* whose primary task is to make sure that the transactions are accurate and valid. Bitcoin is backed by millions of computers across the world which are "miners" [For19]. Typically, the sender in a transaction would be the *prover* and the miner would be the *verifier*, who

[1]Image credit: `https://www.cbinsights.com/research/what-is-blockchain-technology/`

would be involved in an interactive protocol wherein the prover convinces the verifier of some claim (s)he makes.

Let's consider an example where in spite of the assumptions of a confidential transaction we would need something more over just transaction verification by miners. Let's assume a transaction where Alice sends some amount of digital currency (bitcoin, for instance) to Bob and Charlie simultaneously. Alice has $a_1 = 10\text{btc}$[2] and sends $a_2 = 6\text{btc}$, $a_3 = 4\text{btc}$ to Bob and Charlie respectively. Refer to figure 2 in which the amounts hidden (either encrypted or committed) are shown in rectangle. It is easy to check that this transaction is valid[3] if $a_1 = a_2 + a_3$. The above fact is verified by an entity called *miners*. Now, let's say Alice attempts a fraudulent transaction by setting $a_1 = 10\text{btc},\ a_2 = 14\text{btc},\ a_3 = -4 = (n-4)\text{btc}, \text{where } a_i \in \mathbb{Z}_n,\quad i = 1, 2, 3$. Here, $a_1 = a_2 + a_3$ is verified by the *miners*. However, Alice created $14 - 10 = 4\text{btc}$ just out of thin air. Clearly this is not desirable and we should have some protocol which takes care that such fraudulent transactions are not verified at any cost. The most straightforward way to avoid such transactions is to ensure that along with the verification, the amounts also need to be in a specified range. Hence, Range Proofs!

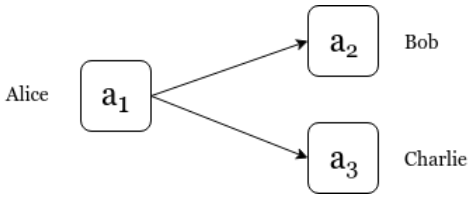


Figure 2: A simple digital currency transaction

1.2 Range Proofs

Range proofs are a form of commitment (or plaintext in case of encryption) validation that allows anyone to verify that a commitment represents an amount within a specified range, without revealing anything else about its (secret) value. They do not leak any other information about the secret value other than that the value lies in a particular range and thus are also referred as *Zero Knowledge Range Proofs*. We will later define the notion of zero-knowledge and range proofs mathematically.

Range proofs have wide-ranging applications in many cryptographic systems. For example, let's a person wishes to get a home loan and (s)he doesn't wish to reveal his/her salary in the process to the financial institution. Thus, a proof that the person's salary is sufficient to rent a house, or obtain a mortgage, without revealing the exact number is a range proof. Similarly, a proof that a person is in a country, without revealing his exact location is also a range proof. E-cash and multi-coupon systems [CHL05; CGH06] often require to prove that the secret counter j of the coins already spent is not greater than the total number J of withdrawn coins. Range proofs are also very useful for anonymous credential systems [CL04].

[2] *Bitcoin* is written to denote the overall cryptocurrency system as a whole and *bitcoin* or *btc* denotes the actual amount of currency.

[3] In this simple example, we are assuming the *miner* fees to be 0.

The one of the first ideas of range proofs was introduced by Boudot [Bou00]. It uses the idea that a committed number $x \in [a, b]$ belongs to $[a - \theta, b + \theta]$ where $\theta = 2^{t+l+1}\sqrt{b-a}$. Further, Lipmaa [Lip03] proposed a range proof which uses integer commitments and Lagrange's four-square theorem.

Theorem 1.1 (Lagrange's theorem) *Let $X \in \mathbb{Z}$. Then $X \geq 0 \iff \exists(x_1, x_2, x_3, x_4) \in \mathbb{Z}^4 : \ X = x_1^2 + x_2^2 + x_3^2 + x_4^2$*

Groth [Gro05] gave an optimized argument for integers of the form $4y + 1, y \in \mathbb{Z}$ since integers of this form only require 3 squares.

Theorem 1.2 (Groth's characterisation) *Let X be an integer which is not of the form $4^n(8k + 7)$, where $n, k \in \mathbb{N}$. Then we have*

$$X \geq 0 \iff \exists(x_1, x_2, x_3) \in \mathbb{Z}^4 : \ X = x_1^2 + x_2^2 + x_3^2 \tag{1}$$

These arguments require just $\mathcal{O}(1)$ commitments. However, each commitment is very large, as the security of the argument or protocol relies on the Strong RSA assumption. Also, these range proofs require a trusted setup.

Camenish et al. [CC08] reduced Set Membership protocol to proving knowledge of signed messages without revealing them. The verifier provides signatures on a 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. This approach uses u-ary digits of the secret value to prove that the secret value is in an interval of the form $[0, n^k - 1]$.

1.3 Bulletproofs

The name 'Bulletproofs' refers to the proposed argument being short like a bullet with bulletproof security assumptions. Bulletproofs is a zero-knowledge argument of knowledge system, to prove that a secret committed value lies in a given interval. As mentioned in section 1.1 emphasizing the fallacies of Blockchain technology, Bulletproofs are proposed to inculcate *anonymity* and *privacy* in confidential transactions.

A trusted setup implies that everyone needs to trust that the setup of the protocol was performed correctly. Bulletproofs do not require a trusted setup. Further, in a distributed system where proofs are transmitted over a network or stored for a long time, short proofs reduce overall cost. Bulletproofs are short non-interactive zero-knowledge proofs which reduce the communication cost to the order of $\mathcal{O}(\log_2(n))$. They rely only on the discrete logarithm assumption, and are made non-interactive using the Fiat-Shamir heuristic. Bulletproofs support a simple and efficient multi-party computation (MPC) protocol that allows multiple parties with secret committed values to jointly generate an aggregated range proof for all their values, without revealing their secret values to each other.

We will now discuss a few of several applications for Bulletproofs along with related work specific to these applications.

1.3.1 Confidential Transactions

Maxwell [Max16] introduced the concept of confidential transactions, where the input, output amounts in a transaction are hidden in Pedersen commitments [Ped92]. However, these confidential transactions do not enable public validation. To enable public validation, the transaction must contain a zero-knowledge proof that the sum of the committed inputs is greater than the sum of the committed outputs (since there are fees inclusive too), and that all the outputs are positive and lie in the interval $[0, 2^n]$. All existing implementations of confidential transactions are of the order $\mathcal{O}(n)$. 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 [Max16]. Bulletproofs greatly build up on this with logarithmic proof size. For transactions with multiple outputs as the one shown in figure 1, with Bulletproofs, m range proofs are merely $\mathcal{O}(m)$ additional group elements along with a single range proof.

1.3.2 Provisions

Provisions protocol allows Bitcoin exchanges to prove that they are solvent without revealing any overhead information [Dag+15]. It relies on range proofs to prevent an exchange from inserting fake accounts with negative balances. A large exchange with 2 million customers would mean the proof size to be of the order of 18 GB of which range proofs amount for 13GB. Here too, using Bulletproofs, the proof would then be dominated by one commitment per customer, with size 62 MB, resulting in a 300 times improvement.

1.3.3 Verifiable shuffles

Consider two lists of committed values $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$ where we have to prove that the second list is a permutation of the first. This problem is called a verifiable shuffle. Currently the most efficient shuffle [BG12] has size $\mathcal{O}(\sqrt{n})$. Bulletproofs can be used to create a verifiable shuffle of size $\mathcal{O}(\log(n))$. The shuffle can be implemented by sorting the two lists and then checking that they are equal (using Bulletproofs protocol for arithmetic circuits).

2 Mathematical Background

Before delving into the math of Range proofs and Bulletproofs, we review some of the necessary tools. We will start with the notations we have used in this report.

2.1 Notations

- $\mathbb{G}$: a cyclic group of prime order p,
- $\mathbb{Z}_p$: the ring of integers modulo p,
- $\mathbb{G}^n, \mathbb{Z}_p^n$: vector spaces of dimension n over $\mathbb{G}, \mathbb{Z}_p$ resp.,
- $\mathbb{Z}_p^\star := \{x \in \mathbb{Z}_p \mid gcd(x, p) = 1\}$
- $\mathbb{Z}_p^\star = \mathbb{Z}_p \setminus \{0\}$, since p is prime,
- $g, h, v, u \in \mathbb{G}$ are generators of $\mathbb{G}$,
- $C = g^a h^\alpha \in \mathbb{G}$ is a pederson commitment to a, note the commitments are capitalized and blinding factors are denoted by Greek letters,
- $x \leftarrow \mathbb{Z}_p^\star$ denotes uniform sampling of an element from $\mathbb{Z}_p^\star$,
- Bold font denotes vectors while capitalized bold font denotes matrices, $\mathbf{a} \in \mathbb{F}^n$ is a vector with elements $a_1, a_2, \ldots, a_n \in \mathbb{F}$, $\mathbf{A} \in \mathbb{F}^{n \times m}$ is a matrix with $a_{i,j}\ \forall\ (i, j) \in [1, n] \times [1, m]$
- $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n}\ a_i \cdot b_i$ denotes the inner product between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$
- $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_n \cdot b_n) \in \mathbb{F}^n$ denotes the Hadamard product between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$
- $p(X) = \sum_{i=0}^{d} \mathbf{p}_i \cdot X^i\ \in \mathbb{Z}_p^n[X],\ \mathbf{p}_i \in \mathbb{Z}_p^n$, is a vector polynomial,
- Inner product between two vector polynomials is given by

$$\langle l(X), r(X) \rangle = \sum_{i=0}^{d} \sum_{j=0}^{i} \langle \mathbf{l}_i, \mathbf{r}_j \rangle \cdot X^{i+j} \in \mathbb{Z}_p[X] \tag{2}$$

- $t(X) = \langle l(X), r(X) \rangle \implies t(x) = \langle l(x), r(x) \rangle\ \forall x \in \mathbb{Z}_p$
- Let $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$ and $\mathbf{a} \in \mathbb{Z}_p^n$, $C := \mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{n} g_i^{a_i}$.
  Here, $C \in \mathbb{G}$ is a commitment to the vector $\mathbf{a} \in \mathbb{Z}_p^n$
- We can consider the above C as the commitment to $\mathbf{a} \circ \mathbf{b}$ if we define the base generator as $g_i' = g_i^{(b_i^{-i})}$,
- $\mathbf{a} \| \mathbf{b}$ denotes concatenation of two vectors,
- We use Python notation to denote slices of vectors

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

- $\mathbf{k}^n = (1, k, k^2, \ldots, k_{n-1}) \in (\mathbb{Z}_p^\star)^n$
- We denote the relation *Relation* using the specified Input and Witness as $\{(Public\ Input;\ Witness) : Relation\}$.
- We refer to $\mathcal{A}$ as a PPT adversary which is a probabilistic Turing Machine that runs in polynomial time in the security parameter λ.

2.2 Assumptions

Definition 2.1 (Discrete Log Relation) *For all PPT adversaries $\mathcal{A}$ and for all $n \geq 2$, $\exists$ a negligible function $\mu(\lambda)$ s.t*

$$\mathbb{P}\left[\begin{array}{ll} G = Setup(1^\lambda), g_1, \ldots, g_n \leftarrow \mathbb{G}; & \\ a_1, \ldots, a_n \in \mathbb{Z}_p \leftarrow \mathcal{A}(\mathbb{G}, g_1, \ldots, g_n) & : \exists a_i \neq 0 \wedge \prod_{i=1}^n g_i^{a_i} = 1 \end{array}\right] \leq \mu(\lambda)$$

We say $\prod_{i=1}^n g_i^{a_i} = 1$ is a non trivial discrete log relation between $g_1, \ldots, g_n$. If the Discrete Log Relation assumption stands, it implies that no PPT adversary can find a non-trivial relation between randomly chosen group elements.

2.3 Commitments

Definition 2.2 (Commitments) *A non-interactive commitment consists of two PPT algorithms* (Setup, Com)*. For a message $x \in \mathbf{M}_{pp}$ (message space), the algorithm proceeds as follows:*

- *public parameters $pp \leftarrow Setup(1^\lambda)$ for security paramter λ*
- $\mathrm{Com}_{pp} : \mathbf{M}_{pp} \times \mathbf{R}_{pp} \to \mathbf{C}_{pp}$*, where $\mathbf{R}_{pp}$ is randomness space*
- $r \leftarrow \mathbf{R}_{pp}$ *and compute* $\mathbf{com} = \mathrm{Com}_{pp}(x; r)$

Definition 2.3 (Homomorphic Commitments) *A homomorphic commitment is a non-interactive commitment such that $\mathbf{M}_{pp}$, $\mathbf{R}_{pp}$, $\mathbf{C}_{pp}$ are all abelian groups, and $\forall\ x_1, x_2 \in \mathbf{M}_{pp}, r_1, r_2 \in \mathbf{R}_{pp}$, 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 2.4 (Hiding Commitment) *A commitment scheme is said to be hiding if for all PPT adversaries $\mathcal{A}$, $\exists \mu(\lambda)$, a negligible function such that,*

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

where the probability is over $b', r, Setup$ and $\mathcal{A}$. For perfectly hiding schemes, $\mu(\lambda) = 0$.

Definition 2.5 (Binding Commitment) *A commitment scheme is said to be binding if for all PPT adversaries $\mathcal{A}$, $\exists \mu(\lambda)$, a negligible function such that,*

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

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

Definition 2.6 (Pedersen Commitment) $\mathbf{M}_{pp}, \mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order p.*

- Setup*:* $g, h \leftarrow \mathbb{G}$
- $\text{Com}(x; r) = (g^x h^r)$

Definition 2.7 (Pedersen Vector Commitment) $\mathbf{M}_{pp} = \mathbb{Z}_p^n$, $\mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order p.*

- Setup*:* $\boldsymbol{g} = (g_1, \ldots, g_n), h \leftarrow \mathbb{G}$
- $\text{Com}(\boldsymbol{x} = (x_1, \ldots, x_n); r) = (h^r \boldsymbol{g}^{\boldsymbol{x}})$

The Pedersen vector commitment is perfectly hiding and computationally binding under the discrete logarithm assumption.

We describe an interactive protocol as follows:

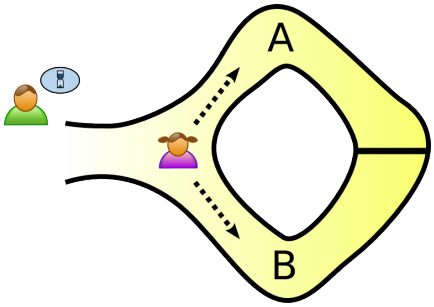
- An *interactive proof* for the decision problem π, is a the following verification protocol:
  - There are two participants, a prover and a verifier.
  - The proof consists of a specified number of rounds.
  - In the beginning, both participants get the same input.
  - In each round, the verifier challenges the prover, and the prover responds to the challenge.
  - Both the verifier and the prover can perform some private computation.
  - At the end, the verifier states whether he was convinced or not.

3 Zero-Knowledge Arguments of Knowledge

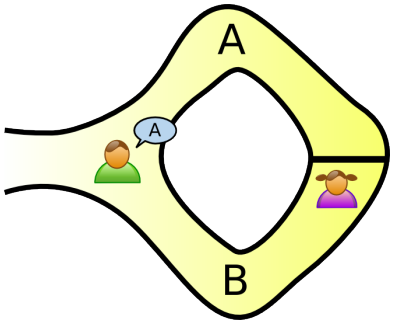
3.1 Zero-Knowledge Arguments

A protocol in which a prover convinces a verifier that a statement is true *without* revealing any information[4] about why it holds is known as a Zero-knowledge argument. An argument is a proof only if the prover is computationally bounded and some computational hardness holds. Hereafter, we use the terms *proof* and *argument* interchangeably.

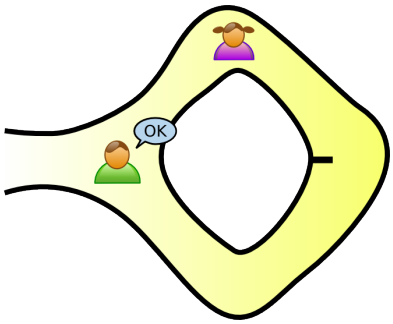
Bulletproofs are zero-knowledge arguments of knowledge. We illustrate the idea of zero-knowledge arguments of proof using the example of *Ali-Baba's secret cave* [Qui+90].[5]



(a) Peggy chooses a path uniformly from A, B without Victor knowing.



(b) Victor asks her to come out of the cave from path A.



(c) If Peggy had entered from path A, she returns trivially. Otherwise, she could open the door using the secret key and return from path A.

Figure 3: Example of a zero knowledge proof

In the above example, Peggy knows the secret word used to open a mysterious door in a cave. The cave is shaped like a horse-hoe. The entrance is on one side and the magic

[4] By *any information*, we mean *any more information* in addition to what is know from *public inputs*.
[5] Figure courtesy: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`

door blocking the opposite side. Victor wants to know whether Peggy knows the secret word; but Peggy, does not want to reveal her knowledge (the secret word) to Victor or to reveal the fact of her knowledge to anyone in the world.

Peggy and Victor run the protocol described in figure 3. Provided she really does know the magic word, and the path she enters and path Victor asks her to come from are same, then it's trivial for Peggy to succeed and Victor to believe that she actually knows the secret key. Further, if the chosen path by Peggy and asked by Victor doesn't match, even then she could open the door and return from a desired path. If they were to repeat this protocol many times, say 15 times in a row, her chance of successfully "guessing" all of Victor's requests would become exponentially small (about three in a lakh).

For zero-knowledge arguments presented in this report, we will consider arguments consisting of three interactive probabilistic polynomial time algorithms $(\text{Setup}, \mathcal{P}, \mathcal{V})$. These algorithms are described by:

* Setup: $\sigma \leftarrow \text{Setup}(1^\lambda)$, σ is common reference string
* $\mathcal{P}$: prover, $\mathcal{V}$: verifier
* Transcript $tr \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle$
* $\langle \mathcal{P}, \mathcal{V} \rangle = b$, $b = 0$ if the verifier rejects or $b = 1$ accepts

Further, we define the relation $\mathcal{R}$ and the CRS-dependent language as:

$$\mathcal{R} := \{(\sigma, u, w) \in \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* : \ w \text{ is a witness for } u \mid \sigma\}$$
$$\mathcal{L}_\sigma := \{x \mid \exists w \in (\sigma, u, w) \in \mathcal{R}\}$$

So, $\mathcal{L}_\sigma$ is essentially the set of statements x that have a witness w in the relation $\mathcal{R}$.

3.2 Defining Zero-Knowledge Arguments of Knowledge

To mathematically define the notion of zero-knowledge and zero-knowledge arguments, we will provide the necessary definitions below.

Definition 3.1 (Argument of Knowledge) *The triple $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for relation $\mathcal{R}$ if it is perfectly complete and has computational witness-extended emulation.*

Definition 3.2 (Perfect completeness) *$(Setup, \mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

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

Perfect completeness implies that if a statement is actually true, then an honest verifier is convinced with probability 1 about the truth of the statement by an honest prover.

Definition 3.3 (Computational Witness-Extended Emulation) *$(Setup, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if for all deterministic polynomial time P^* there exists an expected polynomial time emulator $\mathcal{E}$ such that for all pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$ there exists a negligible function $\mu(\lambda)$ such that*

$$\left| \mathbb{P}\left[\ A_1(tr) = 1 \ \middle| \ \begin{array}{l} \sigma \leftarrow Setup(1^\lambda,) \\ (u,\ s) \leftarrow \mathcal{A}_2(\sigma), \\ tr \leftarrow \langle (\mathcal{P}^*(\sigma, u, s), V(u, s) \rangle \end{array} \right] - \mathbb{P}\left[\begin{array}{l} A_1(tr) = 1 \wedge \\ (tr \text{ accepted} \implies (\sigma, u, w) \in \mathcal{R}) \end{array} \middle| \ \begin{array}{l} \sigma \leftarrow Setup(1^\lambda,) \\ (u,\ s) \leftarrow \mathcal{A}_2(\sigma), \\ (tr,\ w) \leftarrow \mathcal{E}^{\mathcal{O}}(\sigma, u) \end{array} \right] \right| \leq \mu(\lambda)$$

where the oracle is given by $\mathcal{O} = \langle (\mathcal{P}^*(\sigma, u, s), V(u, s) \rangle$, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. We can also define com- putational witness-extended emulation by restricting to non-uniform polynomial time adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$.

Computational witness-extended emulation implies that when an adversary produces an argument to convince the verifier with some probability, then we have a corresponding emulator producing identically distributed argument with same probability, but also a witness.

Definition 3.4 (Public coin) *An argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ is called public coin if all messages sent from the verifier to the prover are chosen uniformly at random and independent of the prover's messages, i.e., the challenges correspond to the verifier's randomness ρ.*

Definition 3.5 (Zero Knowledge Argument of Knowledge) *An argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ is zero knowledge if it reveals no information about w apart from what could be deduced from the fact that $(\sigma, u, w) \in \mathcal{R}$.*

An argument of knowledge is zero knowledge if it does not leak information about w apart from what can be deduced from the fact that $(\sigma, u, w) \in \mathcal{R}$. More explicitly, we note that, a zero knowledge argument of knowledge ensures that no PPT adversary (or verifier) can ever recover w given it's relation with σ, u.

Definition 3.6 (Perfect Special Honest-Verifier Zero-Knowledge) *A public coin argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ is a perfect special honest verifier zero knowledge (SHVZK) argument of knowledge for $\mathcal{R}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$*

$$\mathbb{P}\left[(\sigma, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(tr) = 1 \;\middle|\; \begin{array}{l} \sigma \leftarrow Setup(1^\lambda,) \\ (u,\ w,\ \rho) \leftarrow \mathcal{A}_2(\sigma), \\ tr \leftarrow \langle (\mathcal{P}(\sigma, u, s), V(\sigma, u; \rho) \rangle \end{array} \right]$$

$$= \mathbb{P}\left[(\sigma, u, w) \in \mathcal{R} \wedge \mathcal{A}_1(tr) = 1 \;\middle|\; \begin{array}{l} \sigma \leftarrow Setup(1^\lambda,) \\ (u,\ w,\ \rho) \leftarrow \mathcal{A}_2(\sigma), \\ tr \leftarrow \mathcal{S}(u, \rho) \end{array} \right]$$

PSHVZK AoK implies that even if an adversary chooses a distribution over statements and witnesses, it isn't able to distinguish between simulated transcript and honestly generated transcript for $u \in \mathcal{L}_\sigma$.

Finally, we are now in a position to define Zero-Knowledge Range Proofs. As noted in sub-section 1.2, range proofs are proofs that the prover knows an opening to a commitment, such that the committed value is in a certain range.

Definition 3.7 (Zero-Knowledge Range Proof) *Given a commitment scheme (Setup, Com) over a message space M_{pp}, which is a set with a total ordering, a Zero-Knowledge Range Proof is a SHVZK argument of knowledge for the relation $\mathcal{R}_{Range}$:*

$$\mathcal{R}_{Range} : (pp, (\boldsymbol{com}, l, r), (x, \rho)) \in \mathcal{R}_{Range} \leftrightarrow \boldsymbol{com} = Com(x; \rho) \wedge x \in [l, r]$$

We now are equipped with all the mathematical machinery to dive into knowing how Bulletproofs have emerged as the state-of-the-art range proofs and have already been used in currencies like Monero [Nuz18].

4 Improved Inner-Product Argument

4.1 Building towards Inner-Product Argument

We will define Inner-Product argument which is a proof construction for proving to a verifier that we know two vectors which were committed to obtain a commitment and additionally the the inner product of those two vectors is known. We will first provide simpler protocols which aren't necessarily zero-knowledge but our ultimate aim remains to give an inner-product protocol which is efficient and zero knowledge.

The inputs to the inner-product argument are independent generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, $P \in \mathbb{G}$ and a scalar $c \in \mathbb{Z}_p$. P is a binding vector commitment to $\mathbf{a}, \mathbf{b}$. The argument lets the prover convince a verifier that the prover knows two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ such that

$$P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \qquad c = \langle \mathbf{a}, \mathbf{b} \rangle$$

The inner product argument is an efficient proof system for relation:

$$\{(\boxed{\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p}; \boxed{\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n}) : \boxed{P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle}\} \tag{3}$$

Here, The elements boxed in blue are the inputs, those in red is the relation which is known to the verifier. The content in green box are the witnesses to the relation specified above.

Clearly, the simplest proof system for (3) is: $\mathcal{P}$ sends $\mathcal{V}$ $(\mathbf{a}, \mathbf{b}) \in \mathbb{Z}_p^n$, requiring to send $2n$ elements to $\mathcal{V}$. We wish to build an efficient proof which requires much less elements to be exchanged. Recall, the communication cost directly affects *efficiency* of a protocol.

Note that the witness-CRS relation in (3) is essentially an `and` of two different relations. To simplify things to start with the inner-product argument, we propose to design a proof system for the relation:

$$\{(\boxed{\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}}; \boxed{\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n}) : \boxed{P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle}}\} \tag{4}$$

We will first show a protocol or a proof system for relation (4) and then prove that the same proof system gives a proof system for (3) with same complexity. Define a hash function

$$H : \mathbb{Z}_p^{2n+1} \to \mathbb{G}, \mathbf{a}_1, \mathbf{a}_1', \mathbf{b}_1, \mathbf{b}_1' \in \mathbb{Z}_p^{n/2}, c \in \mathbb{Z}_p \text{ such that}$$
$$H(\mathbf{a}_1, \mathbf{a}_1', \mathbf{b}_1, \mathbf{b}_1', c) := \mathbf{g}_{[:n/2]}^{\mathbf{a}_1} \cdot \mathbf{g}_{[n/2:]}^{\mathbf{a}_1'} \cdot \mathbf{h}_{[:n/2]}^{\mathbf{b}_1} \cdot \mathbf{h}_{[n/2:]}^{\mathbf{b}_1'} \cdot u^c \in \mathbb{G}$$

Further, we we notice that in (4), we can write P as[6]:

$$n' = n/2, \mathbf{a} = (\mathbf{a}_{[:n']}, \mathbf{a}_{[n':]}), \mathbf{b} = (\mathbf{b}_{[:n']}, \mathbf{b}_{[n':]})$$
$$P = H(\mathbf{a}_{[:n']}, \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']}, \mathbf{b}_{[n':]}, \langle \mathbf{a}, \mathbf{b} \rangle)$$

We also note that H is additively homomorphic in its inputs, i.e

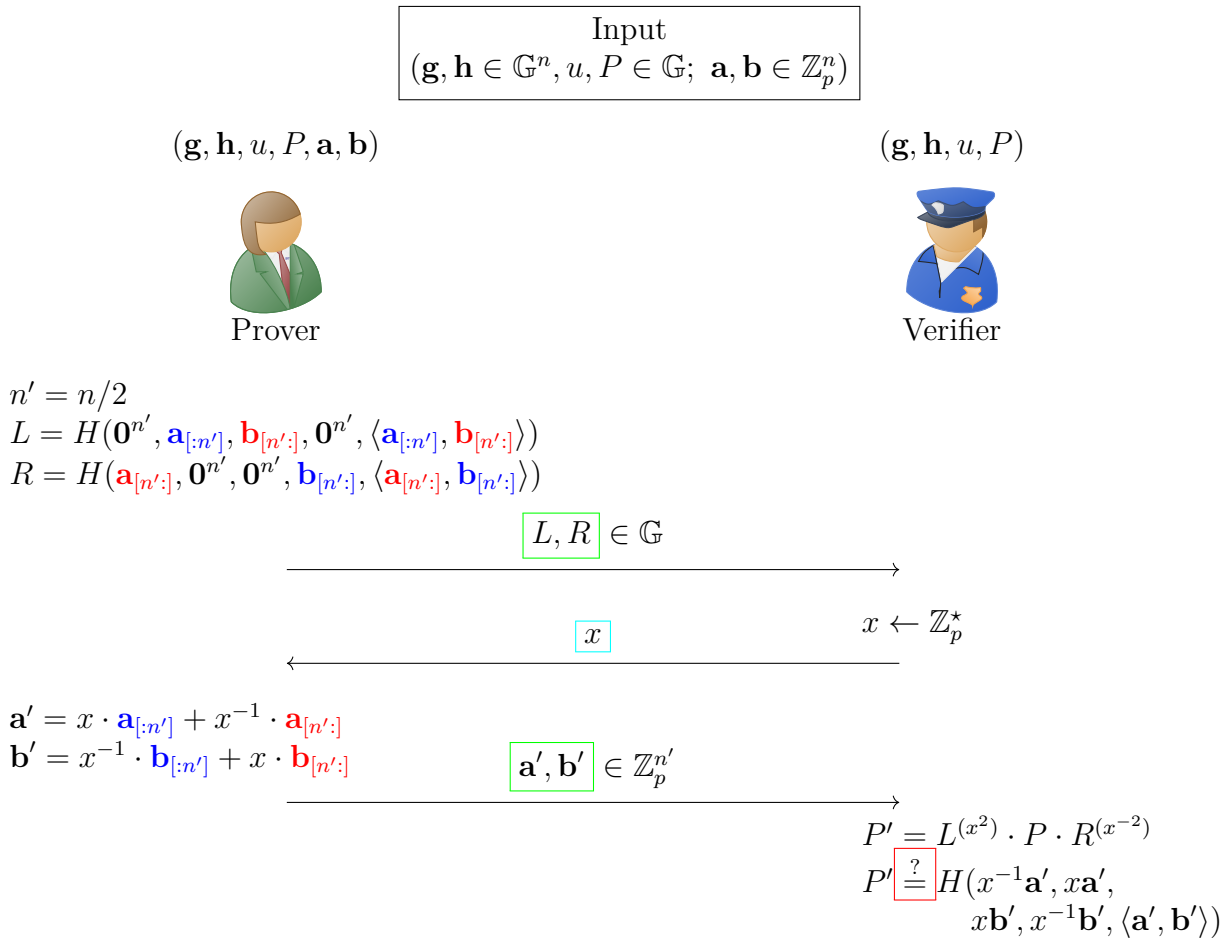
$$H(\mathbf{a}_1, \mathbf{a}'_1, \mathbf{b}_1, \mathbf{b}'_1, c_1) \cdot H(\mathbf{a}_2, \mathbf{a}'_2, \mathbf{b}_1, \mathbf{b}'_2, c_2) = [\mathbf{g}_{[:n/2]}^{\mathbf{a}_1} \cdot \mathbf{g}_{[n/2:]}^{\mathbf{a}'_1} \cdot \mathbf{h}_{[:n/2]}^{\mathbf{b}_1} \cdot \mathbf{h}_{[n/2:]}^{\mathbf{b}'_1} \cdot u^{c_1}] \cdot [\mathbf{g}_{[:n/2]}^{\mathbf{a}_2} \cdot \mathbf{g}_{[n/2:]}^{\mathbf{a}'_2} \cdot \mathbf{h}_{[:n/2]}^{\mathbf{b}_2} \cdot \mathbf{h}_{[n/2:]}^{\mathbf{b}'_2} \cdot u^{c_2}]$$

$$\begin{aligned} \implies H(\mathbf{a}_1, \mathbf{a}'_1, \mathbf{b}_1, \mathbf{b}'_1, c_1) \cdot & H(\mathbf{a}_2, \mathbf{a}'_2, \mathbf{b}_1, \mathbf{b}'_2, c_2) \\ &= [\mathbf{g}_{[:n/2]}^{\mathbf{a}_1+\mathbf{a}_2} \cdot \mathbf{g}_{[n/2:]}^{\mathbf{a}'_1+\mathbf{a}'_2} \cdot \mathbf{h}_{[:n/2]}^{\mathbf{b}_1+\mathbf{b}_2} \cdot \mathbf{h}_{[n/2:]}^{\mathbf{b}'_1+\mathbf{b}'_2} \cdot u^{c_1+c_2}] \\ &= H(\mathbf{a}_1 + \mathbf{a}_2, \mathbf{a}'_1 + \mathbf{a}'_2, \mathbf{b}_1 + \mathbf{b}_2, \mathbf{b}'_1 + \mathbf{b}'_2, c_1 + c_2) \end{aligned}$$

We now describe a protocol for the relation (4) in algorithm (1).

[6]We are denoting the first and second halves of a vector $\mathbf{a}$ by color coding to simplify visualization, $\mathbf{a}_{[:n']}$ as the first half and $\mathbf{a}_{[n':]}$ by second half.

4.2 Inner-Product Argument



Algorithm 1: Protocol for relation (4)

The verifier is convinced because indeed the LHS of the last check by $\mathcal{V}$ can be written as:

$$L^{x^2} \cdot P \cdot R^{x^{-2}} = H(\mathbf{a}_{[:n']} + x^{-2}\mathbf{a}_{[n':]}, x^2\mathbf{a}_{[:n']} + \mathbf{a}_{[n':]}, x^2\mathbf{b}_{[n':]} + \mathbf{b}_{[:n']}, \mathbf{b}_{[n':]} + x^{-2}\mathbf{b}_{[:n']}, \langle \mathbf{a}', \mathbf{b}' \rangle)$$

The key features of this approach are listed below.

- This proof system requires sending $n + 2$ elements.
- To extract a valid witness $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ from a successful prover, we need to rewind the prover three times. After the prover sends L, R we rewind the prover three times to

obtain three tuples $(x_i, \mathbf{a}'_i, \mathbf{b}'_i)$ for $i = 1, 2, 3$ such that for all distinct x_i:

$$L^{(x_i^2)} \cdot P \cdot L^{(-x_i^2)} = H(x^{-1}\mathbf{a}'_i, x_i\mathbf{a}'_i, x_i\mathbf{b}', x_i^{-1}\mathbf{b}'_i, \langle \mathbf{a}'_i, \mathbf{b}'_i \rangle) \tag{5}$$

- We can now find $\nu_1, \nu_2, \nu_3 \in \mathbb{Z}_p$ such that,

$$\sum_{i=1}^{3} x_i^2 \nu_i = 0, \quad \sum_{i=1}^{3} \nu_i = 1, \quad \sum_{i=1}^{3} x_i^{-2} \nu_i = 0$$

- Thus, $\mathbf{a}, \mathbf{b}$ can be found by computing:

$$\mathbf{a} = \sum_{i=1}^{3} (\nu_i \cdot x_i^{-1}\mathbf{a}'_i \parallel \nu_i \cdot x_i^1 \mathbf{a}'_i) \in \mathbb{Z}_p^n$$
$$\mathbf{b} = \sum_{i=1}^{3} (\nu_i \cdot x_i^{-1}\mathbf{b}'_i \parallel \nu_i \cdot x_i^1 \mathbf{b}'_i) \in \mathbb{Z}_p^n$$

- Can we do still improve? Observe that the test in the last check by $\mathcal{V}$ is equivalent to:

$$P' \stackrel{?}{=} \left(\mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}\right)^{\mathbf{a}'} \cdot \left(\mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}}\right)^{\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle}$$

- Thus the prover can recursively engage in an inner-product argument for P' with respect to generators:

$$(\mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}, \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}}, u)$$

- We hope to get a total communication of Protocol 2 to be only $2\lceil \log_2(n) \rceil$ elements in $\mathbb{G}$ plus 2 elements in $\mathbb{Z}_p$. Let's see how we go about doing it in the next section.

4.3 Recursive Inner-Product Argument

Input
$(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G};\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)$

Prover		Verifier
$(\mathbf{g}, \mathbf{h}, u, P, \mathbf{a}, \mathbf{b})$		$(\mathbf{g}, \mathbf{h}, u, P)$
		
if $(n = 1)$:	$a, b \in \mathbb{Z}_p$ →	
		$c = a \cdot b$ $P \stackrel{?}{=} g^a h^b u^c$
else $(n > 1)$: $n' = n/2$ $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$ $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$ $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L} \in \mathbb{G}$ $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R} \in \mathbb{G}$		
	$L, R \in \mathbb{G}$ →	
		$x \leftarrow \mathbb{Z}_p^\star$
	← x	
$\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}'$ $\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}'$ $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$		$\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}'$ $\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}'$ $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$
$\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$ $\mathbf{b}' = x^{-1} \cdot \mathbf{b}_{[:n']} + x \cdot \mathbf{b}_{[n':]} \in \mathbb{Z}_p^{n'}$		

Run Algorithm (1) with input:
$((\mathbf{g}', \mathbf{h}', u, P';\ \mathbf{a}', \mathbf{b}'))$

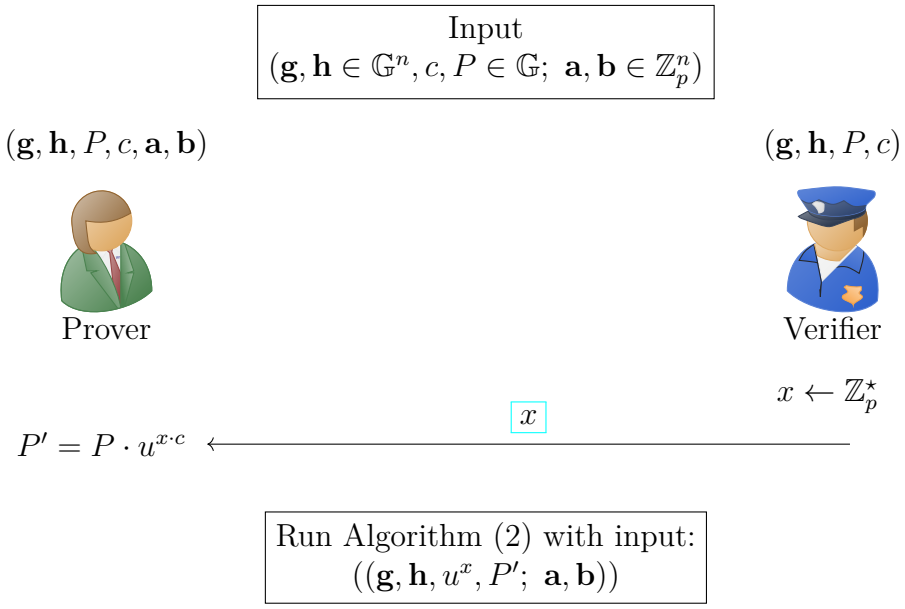
Algorithm 2: Improved Inner-Product Argument

In the above recursive inner-product protocol, the resulting depth is of the order of $\log_2(n)$. The total communication in this protocol is $2 \times \lceil \log_2(n) \rceil$ elements in $\mathbb{G}$ and 2 elements in $\mathbb{Z}_p$, i.e the prover sends the following in the specified order:

$$(L_1, R_1), \ldots, (L_{\log_2(n)}, R_{\log_2(n)}), a, b$$

Now, coming back to the relation (3), we now are in an position to design a proof system for (3). The inner product protocol for the relation (3) is presented in Protocol

(3).



Algorithm 3: Proof system for Relation (3)

We now state the Inner-Product Argument which shows that Protocol (3) is a proof system for (3).

Theorem 4.1 (Inner-Product Argument) *The argument presented in Algorithm (3) for the relation (3) has perfect completeness and statistical witness-extended-emulation for either extracting a non-trivial discrete logarithm relation between $\boldsymbol{g}, \boldsymbol{h}, u$ or extracting a valid witness $\boldsymbol{a}, \boldsymbol{b}$.*

Proof: The proof for Theorem 1 is given in Appendix B in [Bün+17].

5 Logarithmic Range Proof Protocol

Reducing range proof size from linear to logarithmic order is the reason why Bulletproofs are the state-of-the-art today. In this section, we review the proposed logarithmic range proof protocols.

5.1 Inner-Product Range Proof

We wish to design a proof system for the following relation which is equivalent to the range proof relation

$$\{(g, h \in \mathbb{G}, V, n; v, \gamma \in \mathbb{Z}_p) : V = g^v h^\gamma \wedge v \in [0, 2^n - 1]\} \tag{6}$$

Let $\mathbf{a}_L = (a_1, \ldots, a_n) \in \{0,1\}^n$ be the vector containing the bits of v, $v = \langle \mathbf{a}_L, \mathbf{2}^n \rangle$. Prover $\mathcal{P}$ convinces the verifier that $v \in [0, 2^n - 1]$ by proving that:

* It knows $\mathbf{a}_L \in \mathbb{Z}_p^n,\ v, \gamma \in \mathbb{Z}_p$ such that $V = g^v h^\gamma$
* $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v, \quad \text{and} \quad \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n \quad \text{and} \quad \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$

To do so, we take a random linear combination (chosen by the verifier) of the constraints to use inner-product argument. Note that $\langle \mathbf{b}, \mathbf{y}^n \rangle = 0, y \in \mathbb{Z}_p \implies \mathbf{b} = \mathbf{0}^n$. For randomly chosen $y, z \in \mathbb{Z}_p$, we can write:

$$z^2 \cdot \langle \mathbf{a}_L, \mathbf{2}^n \rangle + z \cdot \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle = z^2 \cdot v \tag{7}$$

$$\implies \left\langle \boxed{\mathbf{a}_L - z \cdot \mathbf{1}^n}, \boxed{\mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n} \right\rangle = z^2 + \delta(y, z) \tag{8}$$

$$\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$$

Here, $\delta(y, z) \in \mathbb{Z}_p$ is a quantity that the verifier can easily calculate since $y, z \in \mathbb{Z}_p^\star$ are the challenges generated by the verifier himself. So now, if the prover could send to the verifier the two vectors in the inner product in (8), then the verifier could check (8) using the commitment V to v. Thus the verifier would be convinced that $v \in [0, 2^n - 1]$. But there's an issue in this approach. The verifier can easily extract $\mathbf{a}_L$ from the first vector which the prover sent for calculation of inner product.

To address this issue, by introducing two additional blinding terms $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_p^n$ to blind these vectors. Thus, we define two linear vector polynomials $l(X), r(X) \in \mathbb{Z}_p^n[X]$ and $t(X) \in \mathbb{Z}_p$ as follows:

$$\begin{aligned} l(X) &= \boxed{\mathbf{a}_L - z \cdot \mathbf{1}^n} + s_L \cdot X \\ r(X) &= \boxed{\mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n} + s_R \cdot X) + \boxed{z^2 \cdot \mathbf{2}^n} \\ t(X) &= \langle l(X), r(X) \rangle = \boxed{t_0} + t_1 \cdot X + t_2 \cdot X^2 \end{aligned}$$

The constant terms of $l(X)$ and $r(X)$ are the required inner product vectors in (8). Now, the prover can publish $l(x)$ and $r(x)$ for one $x \in \mathbb{Z}_p$. The constant term of $t(X)$, denoted t_0, is the result of the inner product. The prover needs to convince the verifier that this t_0 satisfies $t_0 = z^2 \cdot v + \delta(y, z)$. To so do, the prover commits to the remaining coefficients of $t(X)$, $t_1, t_2 \in \mathbb{Z}_p$.

Input
$((\mathbf{g}, \mathbf{h} \in \mathbb{G}, V, n;\ v, \gamma \in \mathbb{Z}_p))$

Prover		Verifier
$(\mathbf{g}, \mathbf{h}, V, n, v, \gamma)$		$(\mathbf{g}, \mathbf{h}, V, n)$
$\mathbf{a}_L \in \{0,1\}^n$ s.t. $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$		
$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \in \mathbb{Z}_p^n$		
$\alpha \leftarrow \mathbb{Z}_p$		
$A = h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$		
$\mathbf{s}_L, \mathbf{s}_R \leftarrow \mathbb{Z}_p^n$		
$\rho \leftarrow \mathbb{Z}_p$		
$S = h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$	$\boxed{A, S} \in \mathbb{G}$	
	$\boxed{x, y, z}$	$x, y, z \leftarrow \mathbb{Z}_p^\star$
$\tau_1, \tau_2 \leftarrow \mathbb{Z}_p$		
$T_i = g^{t_i} h^{\tau_i},\ i \in \{1, 2\}$	$\boxed{T_1, T_2} \in \mathbb{G}$	
$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x \ \in \mathbb{Z}_p^n$		
$\mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \ \in \mathbb{Z}_p^n$		
$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$		
$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma \in \mathbb{Z}_p$		
$\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$		
	$\boxed{\tau_x, \mu, \hat{t}} \in \mathbb{Z}_p, \boxed{\mathbf{l}, \mathbf{r}} \in \mathbb{Z}_p^n$	
		$h'_i = h_i^{(y^{-i+1})} \in \mathbb{G}, \forall i$
		$g^{\hat{t}} h^{\tau_x} \stackrel{\boxed{?}}{=} V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2}$
		$P = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} \in \mathbb{G}$
		$P \stackrel{\boxed{?}}{=} h^\mu \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$
		$\hat{t} \stackrel{\boxed{?}}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$

Algorithm 4: Inner-Product Range Proof

The above protocol is logarithmic in size. Some key observations on this protocol are:

- In this protocol, $\mathcal{P}$ transmits $(\mathbf{l}, \mathbf{r})$, whose size is linear in n.
- The total communication cost in this protocol is $2n+4$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p$.
- The verifier, to check if the received $\mathbf{l}, \mathbf{r}$ are actually $l(x), r(x)$ and also check if $t(x) = \langle \mathbf{l}, \mathbf{r} \rangle$, first, switches the generators of the commitment from $\mathbf{h} \in \mathbb{G}^n$ to $\mathbf{h}' = \mathbf{h}^{(\mathbf{y}^{-n})}$.
- Now, A becomes a commitment to $(\mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n)$ w.r.t $(\mathbf{g}, \mathbf{h}', h)$. Similarly, S is now a vector commitment to $(\mathbf{s}_L, \mathbf{s}_R \circ \mathbf{y}^n)$.
- If all three conditions marked by red rectangle result in a positive answer to the verifier, (s)he accepts and thus prover succeeds.

Theorem 5.1 (Range Proof) *The range proof presented above has perfect completeness, perfect special honest verifier zero-knowledge, and computational witness extended emulation.*

Proof: The range proof is a special case of the aggregated range proof with $m = 1$. Refer Appendix C in [Bün+17] for the proof of the theorem 5.1.

With this useful protocol at our hand, we move onto next interesting aspect of Bulletproofs.

5.2 Logarithmic Range Proof

In the above protocol, we had the transfer of $\mathbf{l} \in \mathbb{Z}_p^n$ and $\mathbf{r} \in \mathbb{Z}_p^n$ resulting in proof size proporional to $2n$. For a proof whose size is logarithmic in n, we can eliminate the transfer of $\mathbf{l}$ and $\mathbf{r}$ using the inner-product argument. We also observe that vectors $(\mathbf{l}, \mathbf{r})$ are not secret and hence a protocol that only provides soundness[7] is sufficient.

We observe that verifying lines last and second-last checks of protocol (4) is the same as verifying that the witness $(\mathbf{l}, \mathbf{r})$ satisfies the inner product relation (3) on public input $(\mathbf{g}, \mathbf{h}', Ph^{-\mu}, \hat{t})$, where P commitment to two vectors $\mathbf{l}, \mathbf{r} \in \mathbb{Z}_p^n$ whose inner product is $\hat{t}$.

$$P \stackrel{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$$

$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$$

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

Thus, using the inner product argument, the total communication cost reduces down to $2\lceil \log_2(n) \rceil + 2 + 2$ elements in $\mathbb{G}$ and 5 elements in $\mathbb{Z}_p$.

[7] *Soundness* is the property of only being able to prove "true" things. *Completeness* is the property of being able to prove all true things. [Mat13]

5.3 Aggregating Logarithmic Proofs

In many of the range proof applications described in, 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 as well as the fees to be paid to *miners* for verification. In such a case, we would need to have a range proof for m values individually. So if each of the outputs can be expressed in $[0, 2^n - 1]$, the communication cost would be

$$m \times [(2\lceil \log_2(n) \rceil + 4) + 3]$$

Can we do any better? Let's see. We present a proof system for the relation:

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

Now, $\mathbf{a}_L$ is the concatenation of all of the bits for every v_j. Thus, we have the modified equations of $l(X), r(X)$

$$\begin{aligned} l(X) &= (\mathbf{a}_L - z \cdot \mathbf{1}^{nm}) + \mathbf{s}_L \cdot X \in \mathbb{Z}_p^{nm}[X] \\ r(X) &= \mathbf{y}^{nm} \circ (\mathbf{a}_R + z \cdot \mathbf{1}^{nm} + \mathbf{s}_R \cdot X) + \Sigma_{j=1}^{m} z^{1+j} \cdot (\mathbf{0}^{(j-1)n} \| \mathbf{2}^n \| \mathbf{0}^{(m-j)n}) \\ \tau_x &= \tau_1 \cdot x + \tau_2 \cdot x^2 + \Sigma_{j=1}^{m} z^{1+j} \cdot \gamma_j \end{aligned}$$

Thus the communication cost in this case would be:

$$[(2\lceil \log_2(m \times n) \rceil + 4) + 3] = [(2\lceil \log_2(n) \rceil + 4) + 3] + 2\lceil \log_2(m) \rceil$$

This aggregated range proof uses $2\lceil \log_2(nm) \rceil + 4$ elements in $\mathbb{G}$ as opposed to $m\ (2\lceil \log_2(n) \rceil + 4)$ for m independent range proofs.

Theorem 5.2 (Aggregate Range Proof) *The aggregate range proof presented above has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.*

Proof: The proof for Theorem 5.2 is presented in Appendix C in [Bün+17].

5.4 Fiat-Shamir Heuristic for Non-Interactive Proof

The verifier is a public coin verifier, as all the honest verifier's messages are random elements from $\mathbb{Z}_p^\star$. We can therefore convert the protocol into a non-interactive protocol that is secure and full zero-knowledge in the random oracle model using the Fiat-Shamir heuristic [BR93]. To do so, we just need to replace all random challenges by hashes of the transcript up to that point. For example, $y = \mathrm{H}(A, S)$ and $z = \mathrm{H}(A, S, y)$. For a non-trusted setup we can use such a hash function to generate the public parameters.

5.5 MPC Protocol for Bulletproofs

In several of the applications, the prover could potentially consist of multiple parties (say, m) who each want to generate a single range proof. (Note the difference from aggregating range proofs). For example, 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. Refer to [Max13] to know more about such applications. The question for us is, how could we leverage the advantage provided by the Bulletproofs in terms of communication cost and overall precision.

The MPC protocol works as follows:

- Assign a set of distinct generators $(\mathbf{g}^{(k)}, \mathbf{h}^{(k)})_{k=1}^{m}$ to each party
- Define $\mathbf{g} = (\mathbf{g}^{(1)} \| \mathbf{g}^{(2)} \| \dots \| \mathbf{g}^{(m)})$, $g_i = g_{\lceil i/m \rceil}^{((i-1) \bmod m+1)}$
- Define $\mathbf{h}, \mathbf{h}^{(k)}$ similarly

As described in algorithm (4), in each of it's three rounds, each party generates its part of the proof, i.e.

$$\boxed{A^{(k)}, S^{(k)}}; \boxed{T_1^{(k)}, T_2^{(k)}}; \boxed{\tau_x^{(k)}, \mu^{(k)}, \hat{t}^{(k)}, \mathbf{l}^{(k)}, \mathbf{r}^{(k)}}$$

These shares are then sent to a dealer who simply adds them homomorphically to generate the respective proof component, i.e. $A = \prod_{k=1}^{m} A^{(k)}$ and $\tau_x = \sum_{k=1}^{m} \tau_x^{(k)}$. Then, 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. Finally, the inner product argument is run and the final proof is generated.

6 Theoretical Performance

Bulletproofs have a significant advantage when providing multiple range proofs at once. The proof size for the protocol (4) only grows by an additive logarithmic factor when conducting m range proofs, while all other solutions grow at multiplicative order in m.

	# $\mathbb{G}$ elements	# $\mathbb{Z}_p$ elements
Σ protocol [CD98]	mn	$3mn + 1$
Poelstra [ass17]	$0.63mn$	$1.26mn + 1$
Bulletproofs	$2(\log_2(n) + \log_2(m)) + 4$	5

Table 1: Range proof size for m proofs and range $[0, 2^n - 1]$.

We reviewed in detail the sub-protocols which finally led us to understand Bulletproofs. The following table helps us summarise their communication cost and also compares it with the existing protocols.

	# $\mathbb{G}$ elements	# $\mathbb{Z}_p$ elements
Simplest Inner-product protocol	$2n$	0
Inner-product argument, Section (4.2)	n	2
Recursive Inner-Product Argument , Section (4.3)	$2(\lceil \log_2(n) \rceil)$	2
Inner-Product range proof using inner-product argument , Section (5.1)	$2n + 4$	3
Inner-Product range proof using recursive Inner-Product Argument, Section (5.2)	$2(\lceil \log_2(n) \rceil) + 4$	5
Aggregate Range Proof using m individual range proofs , Section (5.3)	$m \times \{2(\lceil \log_2(n) \rceil) + 4\}$	$5m$
Aggregate Range Proof using inner-product argument, Section (5.3)	$2\{\lceil \log_2(n) \rceil + \lceil \log_2(m) \rceil\}$ $+4$	5

Table 2: Proof sizes for different inner product and ARP protocols.

References

[ass17] Confidential assets. *Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille*. 2017. URL: `https://blockstream.com/bitcoin17-final41.pdf` (visited on 21/05/2013).

[BG12] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 263–280. ISBN: 978-3-642-29011-4.

[Bou00] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 431–444. ISBN: 978-3-540-45539-4.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: ACM Press, 1993, pp. 62–73.

[Bün+17] Benedikt Bünz et al. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *Cryptology ePrint Archive, Report 2017/1066*, 2017. URL: `https://eprint.iacr.org/2017/1066`.

[CC08] Jan Camenisch, Rafik Chaabouni, and abhi shelat abhi. "Efficient Protocols for Set Membership and Range Proofs". In: *Advances in Cryptology - ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 234–252. ISBN: 978-3-540-89255-7.

[CD98] Ronald Cramer and Ivan Damgård. "Zero-Knowledge Proofs for Finite Field Arithmetic; or: Can Zero-Knowledge Be for Free?" In: *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '98. London, UK, UK: Springer-Verlag, 1998, pp. 424–441. ISBN: 3-540-64892-5. URL: `http://dl.acm.org/citation.cfm?id=646763.706339`.

[CGH06] Sébastien Canard, Aline Gouget, and Emeline Hufschmitt. "A Handy Multi-coupon System". In: *Applied Cryptography and Network Security*. Ed. by Jianying Zhou, Moti Yung, and Feng Bao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 66–81. ISBN: 978-3-540-34704-0.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. "Compact E-Cash". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 302–321. ISBN: 978-3-540-32055-5.

[CL04] Jan Camenisch and Anna Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps". In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matt Franklin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–72. ISBN: 978-3-540-28628-8.

[Dag+15] Gaby G. Dagher et al. "Provisions: Privacy-preserving Proofs of Solvency for Bitcoin Exchanges". In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1008.

[Dav82] Chaum David. "Blind Signatures for Payments". In: *Springer-Verlag* (1982). URL: http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF.

[Fed+18] Amir Feder et al. "The Rise and Fall of Cryptocurrencies". In: *Workshop on the Economics of Information Security* (June 2018). URL: http://par.nsf.gov/biblio/10066236.

[For19] Luke Fortney. *Investopedia*. 2019. URL: https://www.investopedia.com/terms/b/bitcoin-mining.asp (visited on 02/03/2019).

[Gar98] Kessler Gary C. "An Overview of Cryptography". In: *Handbook on Local Area Networks* (Sept. 1998). URL: https://www.garykessler.net/library/crypto.html.

[Gro05] Jens Groth. "Non-interactive Zero-Knowledge Arguments for Voting". In: *Applied Cryptography and Network Security*. Ed. by John Ioannidis, Angelos Keromytis, and Moti Yung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 467–482. ISBN: 978-3-540-31542-1.

[Jon+16] Bootle Jonathan et al. "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting". In: *Springer Berlin Heidelberg*, 2016, pp. 327–357. ISBN: 978-3-662-49896-5. URL: https://www.iacr.org/archive/eurocrypt2016/96650200/96650200.pdf.

[Lip03] Helger Lipmaa. "On Diophantine Complexity and Statistical Zero-Knowledge Arguments". In: *IACR Cryptology ePrint Archive*. 2003.

[Mat13] Luke Mathieson. *The difference between soundness and completeness*. 2013. URL: https://philosophy.stackexchange.com/questions/6992/the-difference-between-soundness-and-completeness (visited on 24/05/2013).

[Max13] Greg Maxwell. *CoinJoin: Bitcoin privacy for the real world*. 2013. URL: bitcointalk.org (visited on 24/17/2013).

[Max16] Greg Maxwell. *Confidential transactions*. 2016. URL: https://people.xiph.org/~greg/confidential_values.txt (visited on 02/11/2019).

[Nak09] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: http://bitcoin.org/bitcoin.pdf.

[NKG17] Harish Natarajan, Solvej Karla Krause, and Helen Luskin Gradstein. "Distributed Ledger Technology (DLT) and blockchain (English)". In: *FinTech note; no. 1. Washington, D.C. : World Bank Group* (2017). URL: http://documents.worldbank.org/curated/en/177911513714062215/Distributed-Ledger-Technology-DLT-and-blockchain.

[Nuz18] Lucas Nuzzi. *Monero Becomes Bulletproof*. 2018. URL: https://medium.com/digitalassetresearch/monero-becomes-bulletproof-f98c6408babf (visited on 04/17/2019).

[Ped92] Torben Pryds Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology — CRYPTO '91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140. ISBN: 978-3-540-46766-3.

[Qui+90] Jean-Jacques Quisquater et al. "How to Explain Zero-Knowledge Protocols to Your Children". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 628–631. ISBN: 978-0-387-34805-6.