UCLouvain

École polytechnique de Louvain

Risk-Limiting Audit Optimization with ElectionGuard

Zero-Knowledge Arguments on Chaum-Pedersen multi-commitments

Authors: **Marie Lonfils, Alexis Vuille**
Supervisors: **Olivier Pereira, Thomas Peters**
Reader: **Edouard Cuvelier**
Academic year 2022–2023
Master [120] in Mathematical Engineering
Master [120] in Computer Science and Engineering

Contents

Abstract

This master's thesis focuses on the development of cryptographic zero-knowledge protocols within the context of ElectionGuard, an open-source voting software created by Microsoft. The primary objective is to shorten the size of the proofs that are used in risk-limiting audits to make them more efficient and practical.
Drawing inspiration from the "Bulletproofs" paper by Stanford University, we design cryptographic zero-knowledge protocols that facilitate the efficient generation of logarithmic-sized proofs.

In the context of risk-limiting audits, we consider multi-Pedersen commitments that gather the l selections cast in a ballot, $v_1, \ldots, v_l$, in one group element $V = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l}$. We designed three protocols among which the two first protocols prove in a zero-knowledge fashion that the committed votes are valid by proving that they are bits (first protocol) and satisfy the K-selection limit (second protocol), i.e. at most K choices are selected on a ballot. The third protocol allows to do partial opening of votes commitment, that is revealing one of the selections, v_j committed inside V.

This paper begins with an introduction to zero-knowledge protocols. We then present the mathematical descriptions of our protocols, emphasizing the key design concepts employed. Furthermore, we provide rigorous security guarantees for these protocols, supported by mathematical proofs for their zero-knowledge and soundness properties. Additionally, we predict the theoretical complexities of our protocols.
To validate our research, we implemented the three main zero-knowledge protocols, observing successful performance that is aligned with our theoretical predictions.

By accomplishing these objectives, our research contributes to the advancement of secure and efficient auditing in the context of voting systems and specifically within the ElectionGuard framework by reaching logarithmic proof sizes.

Acknowledgements

This thesis and the research behind it would not have been possible without the support of our supervisors Professor Pereira and Professor Peters. We would like to take this opportunity to express our gratitude towards both of them.

Part I

Introduction

This master thesis is dedicated to the elaboration of new cryptographic primitives and zero-knowledge arguments that are more data efficient in the context of Risk Limiting Audits for voting and ElectionGuard.

In Chapter 1, we present ElectionGuard an open-source software developed by Microsoft to enhance the security, transparency, and accessibility of voting.

Chapter 2 explains briefly what Risk Limiting Audits are and we explain our idea to replace ElGamal ciphertexts with multi-Pedersen commitments to have more data efficient audit trails and associated proofs to perform audits.

Chapter 3 gives some background to apprehend the cryptographic tools presented in this paper and in particular on zero-knowledge arguments, which are explained in detail in section 3.3. A reader who already has some experience with zero-knowledge proofs can skip this chapter.

Chapter 4 introduces some of the main notations used in this paper.

Chapter 5 presents the main goals of this paper. It also explains for what purposes we develop the zero-knowledge protocols presented in the paper.

Chapter 6 presents the main cryptographic building blocks used later on in the zero-knowledge proof we design. This includes the "Bulletproofs" inner product argument developed by Stanford University, University College London and Blockstream and the "Extended-Schnorr" argument which we adapted in a straightforward way from the "Bulletproofs" inner product argument.

Chapter 7 is a draft of the zero-knowledge protocol to prove that multi-Pedersen commitments are commitments of bits, while Chapter 8 is the definitive protocol which achieves logarithmic proof size.

Chapter 9 presents the K-selection proof protocol which allows to prove in a zero-knowledge fashion that multi-Pedersen commitments satisfy the selection limit, that is the number of options selected by a voter inside the commitment is within a certain range.

Chapter 10 presents a protocol that allows performing Partial Opening, which is, given a multi-Pedersen commitment of selections, proving that one of the selections of the voter inside the commitment is equal to a specific value.

We provide for all protocols, their respective complexity as well as a formal security proof.

Finally, in Chapter 11 we present briefly how we implement all the different protocols presented earlier using *Python* and we show how well our protocols perform in practice.

Chapter 1

ElectionGuard

ElectionGuard is an open-source SDK developed for secure voting by Microsoft.
According to their website: "ElectionGuard is an open source software development kit (SDK) that makes voting more secure, transparent and accessible. It is designed for election system vendors to incorporate end-to-end verifiability into their systems and any interested organization to perform and publish post-election audits." [21]

ElectionGuard incorporates cryptographic techniques to enable secure and private voting.
Some of the main key features of ElectionGuard are :

- **(i) End-to-End Encryption :** ElectionGuard employs homomorphic encryption techniques to safeguard the privacy and integrity of a voter's ballot throughout the entire election process, including the tallying stage.
- **(ii) End-to-End Verifiability :** ElectionGuard allows voters to independently verify that their vote has been accurately recorded and included in the final tally, while still maintaining ballot secrecy by providing them with a receipt after casting their vote.
- **(iii) Risk Limiting Audits :** ElectionGuard allows election authorities to conduct risk-limiting audits, which involve sampling a subset of the ballots to ensure the accuracy and integrity of the election outcome through statistical techniques.

We will mainly focus in this master thesis on **Risk Limiting Audits** and in particular we will see cryptographic techniques that allow to conduct more data efficient audits.

1.1 Contest

A contest in the context ElectionGuard is defined as a set of options (for example candidates) that can be checked by the voter together with a selection limit. [14]

1.2 Ballot

A clear form ballot in ElectionGuard is the plaintext representation of a voter's selections.
For one contest, a ballot is represented by a sequence of bits (0 or 1). The i-th bit in the ballot indicates whether or not the i-th option has been selected for not.[20]

Given one contest with a set of l options we can thus represent a ballot mathematically as $\mathbf{v} = (v_1, \ldots, v_l)$, with $v_i := \begin{cases} 1 & \text{if option } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$ for $i \in \{1, \ldots, l\}$.

For example, a plaintext ballot could have the form $\mathbf{v} = (0, 1, 0, 0, 1, 0)$ indicating that the voter has selected the second and the fifth options in a contest with 6 options, while he did not choose the first, the third, the fourth and the sixth options.

1.3 Selection Limit

As mentioned previously in every contest there is a certain selection limit that is defined.

In a contest, the selection limit is the number of selections that a voter is allowed to choose.
In most classical contests it is set to 1 as the voter can vote for at most one candidate but in some elections, it is not uncommon to have larger selection limits and the voter can for example vote for his three preferred candidates.[15]

More formally, given a contest with a set of l options, assume a ballot has the form $\mathbf{v} = (v_1, \ldots, v_l)$, with $v_i \in \{0, 1\}$ for $i \in \{1, \ldots, l\}$. Then, generally, the selection limit is defined as a natural number $K \in \mathbb{N}$ such that at most K options among the l are selected that is $< \mathbf{1}^l, \mathbf{v} >= \sum_{i=1}^{l} v_i \leq K$.
However, to be more general we can consider a lower bound $Min \in \mathbb{N}$ and an upper bound $Max \in \mathbb{N}$ such that the number of selections is between those two bounds, i.e. $\sum_{j=1}^{l} v_j \in [Min, Max]$

1.4 Use cases

To this point, ElectionGuard has been deployed in two pilot elections: a system developed by VotingWorks used a ballot-marking device coupled with a printer in Fulton, Wisconsin in February 2020, and Hart InterCivic's Verity precinct scanner used ElectionGuard in Preston, Idaho in their November 2022 general election. In this election, ElectionGuard was introduced as a pilot and was used by 109 participants without any problem. [10]

ElectionGuard is designed to support enhanced risk-limiting audits. In some use cases, it is employed with a ballot marking device with very little memory. It is thus crucial to reduce the amount of data generated by the device for each vote.

Chapter 2

Risk Limiting Audit

2.1 Concept

Risk Limiting Audit (often abbreviated RLA) is a statistical method to check the result of an election. It generally involves the sampling of cast votes and manual recount for these votes to determine the confidence level in the outcome of the election.

The goal of Risk Limiting Audits is not to deliver a 100% guarantee that the result of the election is correct but to detect with high probability if the result is wrong. Verifying with a 100% guarantee the outcome of an election would take too much time and be too costly, while RLA offers an efficient and sufficiently accurate (but not perfect) way to check the election's results. It can provide a very high level of confidence that the election results are accurate.

The size of the sample used in the Risk Limiting Audit depends on the margin of victory (the difference in the number of votes between the candidate that received the most votes and the one that received the second-most votes) for the election and the level of confidence desired. An election with a smaller margin of victory will require more samples of cast ballots to obtain the same confidence in an election than an election with a larger margin of victory.

The confidence level in the result of the election computed will determine if a full recount of the ballots is required or not. If the confidence level is high, the audit can be concluded. However, if the confidence level is too low, under some predefined threshold, a full manual recount is necessary.

The purpose of Risk Limiting Audit is thus to detect cheating and intentional modifications of ballots as well as bugs in the vote's scanner, in some software used or in any other technology used in the election.

2.2 RLA with multi-Pedersen commitments

In ElectionGuard Risk Limiting Audits use ElGamal ciphertexts of the form $(g^r, h^r g^v)$, with one ciphertext for each selection in a ballot. Subsequently, one ballot with l options requires l group elements.
The associated zero-knowledge arguments size to prove that a ballot is valid scale linearly with the number of selections and the number of votes.

In practice, it is possible to conduct Risk Limiting Audit using multi-Pedersen commitments. This practice allows for the use of a single group element per ballot and zero-knowledge arguments that scales logarithmically with the number of selections and votes (and thus more data efficient) as we shall see in this paper.
To conduct an audit with multi-Pedersen votes commitments, we will ask each voter j to commit on his selections $v_1^{(j)}, \ldots, v_l^{(j)} \in \{0, 1\}$ through the multi-Pedersen commitment $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}}$, where γ_j is the randomness used to have a perfectly hiding commitment that preserves the privacy and anonymity of the voter.

Then to conduct the RLA we will select a sample of these V_j. The size of the sample will be determined based on statistical principles and the level of confidence wanted for the audit.

Then, there exist several approaches which we will not present here, that perform statistical methods on the commitments $\{V_j\}$ to check the veracity of the result of the election.

In order to be able to conduct the audit we need to be assured that the commitments $\{V_j\}$ are commitments over valid votes ballots.
A ballot of votes $(v_1, \ldots, v_l)$ is considered to be valid if it satisfies the two followings properties :

- **(i) 0-1 Property :** all selections inside the votes ballot are bits, i.e. $v_i \in \{0,1\}\ \forall i \in \{1,\ldots,l\}$,
- **(ii) Selection Limit :** at most K selections are chosen inside the ballot, *i.e.* $\sum_{i=1}^{l} v_i \leq K$

We thus want, given a commitment $V = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l}$ of a ballot of votes of the form $(v_1, \ldots, v_l)$, to be able to prove that the ballot of votes is valid by showing that $v_i \in \{0,1\}\ \forall i \in \{1,\ldots,l\}$ and $\sum_{i=1}^{l} v_i \leq K$.
However, we would like to do it in a way that preserves the privacy and anonymity of the user. So we would like to prove the two above properties without sending in clear the plaintext ballots (i.e. the values $(v_1, \ldots, v_l)$) or even the number of selections made by a certain voter $(\sum_{i=1}^{l} v_i)$.
In particular, we want to prove that the plaintext ballot inside the commitment satisfies the **0-1 Property** and the **Selection Limit** but not leak any other information on the committed ballot of votes.

This motivates the design of the zero-knowledge protocols in this master thesis. We have one zero-knowledge protocol to prove the **0-1 Property** and another one to prove the **Selection Limit**.

2.3 Partial Opening

One thing that is useful in an election audit is to be able to perform the full opening and the partial opening of the commitments in order to be able to recount manually some of the votes.
We explain in this section briefly and roughly the concept of commitments, multi-commitments, opening and partial opening.

If you are not familiar with the concept of commitments and opening, a commitment can be seen as a locked box that you send to someone with something that you put inside the box which he cannot see and you cannot change. However, the box has a special key which you can be sent later on to open the box and see its content.

In our case, the locked box is $V = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l}$ and it contains inside the plaintext ballot $(v_1, \ldots, v_l)$.
One possible way to open the commitment is to send the randomness γ to someone who already has V, who can then get the plaintext votes by looking for the bits $(v_1, \ldots, v_l)$ such that $g_1^{v_1} \ldots g_l^{v_l} = V/h^{\gamma}$.

In our case, the commitment V is a multi-commitment that, continuing our analogy, can be seen as a locked box which contains several items (the different selections $\{v_i\}$) instead of one item and we can send the key to extract just one item of the box and not the whole content of the box.

It is useful, in order to conduct Risk Limiting Audit, to perform a partial opening that is getting only one of the values contained in the multi-commitment, which corresponds in our case to get one of the selection v_k for some given $k \in \{1,\ldots,l\}$, while the other selections v_i remain unknown for $i \neq k$.

This motivates the last protocol developed in this paper which allows to perform partial opening on commitments of the form $V = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l}$.

Part II

Preliminaries

Chapter 3

Theoretical Background

3.1 Discrete Log Relation

Given a PPT adversary $\mathcal{A}$, a security parameter $\lambda \geq 1$ and $n \geq 2$ let's define the experiment $DLogR_{\mathcal{A}}(\lambda)$ as follows :

$DLogR_{\mathcal{A}}(\lambda)$

(i) Pick group $\mathbb{G} \leftarrow Setup(1^{\lambda})$ of order q.

(ii) Pick uniformly at random generators$g_1, \ldots, g_n$ from $\mathbb{G}$ and send $g_1, \ldots, g_n$ to $\mathcal{A}$.

(iii) $\mathcal{A}(g_1, \ldots, g_n)$ outputs $a_1, \ldots, a_n \in \mathbb{Z}_p$.

(iv) Define the outcome of the experiment : $DLogR_{\mathcal{A}}(\lambda) := 1$ iif $\exists i : a_i \neq 0 \wedge \prod_{i=1}^{n} g_i^{a_i} = 1$.

The *Discrete Log Relation* assumption is that $\forall$ $\mathcal{A}$ PPT and $\forall n \geq 2$, there exists a negligible function $\epsilon(\lambda)$ such that

$$P[DLogR_{\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$$

, where the probability is taken over the random coins used in $\mathcal{A}$ as well as the random coins used in the experiment $DLogR_{\mathcal{A}}(\lambda)$ (to pick $\mathbb{G}$ and $g_1, \ldots, g_n$).

The *Discrete Log Relation* assumption will be very useful when we will do the security proofs of our protocols and in particular when we will do the witness-extended emulation proof.

3.2 Schwartz-Zippel Lemma

3.2.1 Lemma

Let F be a field and $S \subseteq F$, $|S| < +\infty$. Let $f(x_1, \ldots, x_n)$ be a polynomial of total degree $d \geq 0$ different from the zero polynomial. Then, if $r_1, \ldots, r_n$ are picked independently uniformly at random in S, we have $P[f(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|}$.

3.2.2 Particular case

We have presented for fun the general version of the Swchartz-Zippel lemma but in fact, we will be only interested in this paper in a particular case of the Swhartz-Zippel lemma where $F = \mathbb{Z}_p$, $S = F$ and $n = 1$.
In this case, the Swchartz-Zippel lemma becomes:
Let $f(x)$ be a polynomial of degree $d \geq 0$ different from the zero polynomial. Then, if r is picked uniformly at random in $\mathbb{Z}_p$, we have $P[f(r) = 0] \leq \frac{d}{|\mathbb{Z}_p|} = \frac{d}{p}$.

In this particular case of the Lemma, the proof is straightforward:
PROOF : In a field, every polynomial of degree d has at most d roots. Let's denote the (distinct) roots of f as $\{x_j\}_{j=1}^{k}$, with $k \leq d$. If r satisfies $f(r) = 0$, by definition it is a root of f, i.e. $r = x_i$ for some $i \in \{1, \ldots, k\}$. Since

r is picked uniformly at random in $\mathbb{Z}_p$, $P[r = x_i] = \frac{1}{\mathbb{Z}_p} = \frac{1}{p}\ \forall i \in \{1, \ldots, k\}$.
Then, we have $P[f(r) = 0] = P[r = x_1 \vee r = x_2 \vee \ldots \vee r = x_k] = \sum_{i=1}^{k} P[r = x_i] = \frac{k}{p} \leq \frac{d}{p}$, which thereby concludes the proof.

3.2.3 Application

We have used the Schwartz-Zippel lemma in order to design our protocols. It is very interesting conceptually, especially in order to combine verification equations.
In particular, in our protocols, we will often want at some point that a certain system of verification equations of the following form hold: $e_1(x_1, \ldots, x_n) = e_2(x_1, \ldots, x_n) = \ldots = e_d(x_1, \ldots, x_n) = 0$ (with d some fixed number independent of p), then instead of checking all these d equations separately we can check them all in once in one verification equation by picking r uniformly at random and checking instead of $e_i(x_1, \ldots, x_n) = 0$ $\forall i \in \{1, \ldots, n\}$ check whether the equation $\sum_{k=1}^{d} e_k(x_1, \ldots, x_n) r^k = 0$ hold or equivalently $f(r) = 0$, with $f(x) = \sum_{k=1}^{d} e_k(x_1, \ldots, x_n) x^k$ a polynomial of degree d.

Indeed, either the verification equations $e_k(x_1, \ldots, x_n) = 0$ hold for all $k \in \{1, \ldots, d\}$ and $f(x)$ is the zero polynomial and the verification equation $f(r) = 0$ pass. Or the verification equations $e_k(x_1, \ldots, x_n) = 0$ does not hold for some $k \in \{1, \ldots, d\}$ but then $f(x)$ is not the zero polynomial and we can apply the Schwartz-Zippel lemma to say the probability that the verification equation $f(r) = 0$ passed is negligible (less than $\frac{d}{p}$, which is negligible) since r is picked uniformly at random in $\mathbb{Z}_p$.

3.3 Zero-Knowledge Arguments

In this section, we first describe intuitively the concept of zero-knowledge arguments as well as the key properties defining it. We then give a more formal definition of the zero-knowledge argument. Finally, we speak about the different efficiency criteria to evaluate zero-knowledge arguments' practical performance.

3.3.1 Intuitive Description

A zero-knowledge argument of knowledge designs a protocol between two parties in which the first party called the prover (denoted $\mathcal{P}$) proves to the second party called the verifier (denoted $\mathcal{V}$) that a statement is true without leaking any knowledge beyond the truth of the statement. In many cases and it will be the case for all the protocols designed here, we not only require that the prover convinces the verifier that a statement is true but we also ask that the prover actually knows a witness which shows that the statement is true. More precisely, we require that if the prover manages to prove to the verifier that a statement is true, then he must necessarily know the witness that proves that the statement is actually true, or at least almost necessarily in a sense that will be formalized later on.

An argument protocol can be described very roughly as a sequence of queries done by the prover and the verifier that ends up with the verifier outputting either 1 (meaning that the verifier accepts the proof, i.e. he is convinced by the prover that the statement is true) or 0 (meaning that the verifier rejects the proof).

The correctness and the security of a zero-knowledge argument protocol are based on the three followings properties :

- **(i) Computational Witness Extended Emulation :** if at the end of the protocol, the verifier accepts the proof, then we know that except with negligible probability that the prover knows a valid witness satisfying the relation even if the prover is dishonest (does not follow the protocol).
- **(ii) Honest-Verifier Zero-Knowledge :** under the assumption that the prover is honest and the verifier is honest-but-curious (he follows the protocol but tries to learn what he can), the verifier does not learn anything about the prover's witness.
- **(iii) Completeness :** Assuming $\mathcal{P}$ follows honestly the protocol knowing a witness that satisfies the given statement, then the verifier always accepts the proof.

These three key concepts are also defined formally in the subsection below.

If the protocol satisfies properties **(i)** and **(ii)** we call it an *argument of knowledge*. If in addition, it satisfies property **(iii)** and all challenges are uniformly distributed independently of the prover's messages we call it a *perfect special honest-verifier zero-knowledge argument of knowledge*.

3.3.2 Formal Description

Let $(Setup, \mathcal{P}, \mathcal{V})$ be a triple of PPT algorithms, with :

- (i) $Setup$: generate the Common Reference String (C.R.S.),
- (ii) $\mathcal{P}$: prover's algorithm,
- (iii) $\mathcal{V}$: verifier's algorithm.

We denote $tr \leftarrow < \mathcal{P}(x), \mathcal{V}(y) >$, as the transcript resulting from the interaction between the prover and the verifier on inputs x and y respectively.
We write : $< \mathcal{P}(x), \mathcal{V}(y) >= b$ depending on whether the verifier accepts the proof, $b = 0$, or rejects it, $b = 1$.

In addition, let $R \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a polynomial time decidable ternary relation.
Given the C.R.S. σ and the statement u, we say w is a witness for relation R if $(\sigma, u, w) \in R$.

The triple $(Setup, \mathcal{P}, \mathcal{V})$ is an *argument of knowledge* for the relation R if it has the *Perfect Completeness* and the *Computational Witness Extended Emulation* properties defined below.

The triple $(Setup, \mathcal{P}, \mathcal{V})$ is a *Perfect Special Honest-Verifier Zero-Knowledge argument of knowledge* for the relation R if it is a *public coin argument* that satisfies the *Perfect Special Honest-Verifier Zero-Knowledge* property as we will define below.

3.3.3 Perfect Completeness

The triple $(Setup, \mathcal{P}, \mathcal{V})$ has the *Perfect Completeness* property if for all non uniform adversary $\mathcal{A}$, we have :

$$P\left[(\sigma, u, w) \notin R \vee < \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma)\right] = 1.$$

The definition may seem a bit intricate for an uninformed reader but in fact, it is very intuitive. The definition simply puts in mathematical terms the following idea: no matter what choice of statement with the associated witness an adversary makes, we have either that the witness is not a valid witness, $(\sigma, u, w) \in R$), or it is a valid witness and then the interaction between the prover and the verifier results in an accepting conversation $(< \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1)$.

In practice we will show : $P\left[< \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \in R\right] = 1$, which implies *Perfect Completeness* in all the proofs below. If you want more detail, here is formally why this is true:

$$P\left[(\sigma, u, w) \notin R \vee < \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma)\right] =$$

$$\underbrace{P\left[(\sigma, u, w) \notin R \vee < \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \notin R\right]}_{=1 \text{ since } (\sigma, u, w) \notin R} \times P\left[(\sigma, u, w) \notin R\right] +$$

$$\underbrace{P\left[(\sigma, u, w) \notin R \vee < \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \in R\right]}_{=1 \text{ as we have assumed}} \times P\left[(\sigma, u, w) \in R\right] =$$

$$P\left[(\sigma, u, w) \notin R\right] + P\left[(\sigma, u, w) \in R\right] = 1.$$

3.3.4 Computational Witness Extended Emulation

The triple $(Setup, \mathcal{P}, \mathcal{V})$ has *Computational 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 :

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

, where $\langle \mathcal{P}^*(\sigma, u, s), \mathcal{V}(\sigma, u) \rangle$ is the oracle called by $\mathcal{E}$ which permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.

In the definition, s is to be interpreted as the state of P^* including the randomness. So, whenever P^* is able to make a convincing argument when in state s, $\mathcal{E}$ can extract a witness.

Once again the definition may likely seem very intricate for uninformed readers but in fact, it is more or less intuitive. The definition simply puts in mathematical terms the following idea: no matter what choice of statement an adversary makes, if at the interaction between any dishonest prover (he can not at all follow the protocol, this is why we put $\mathcal{P}^*$ instead of $\mathcal{P}$) and an honest verifier ends up in an accepting transcript, then it must be the case that $\mathcal{P}$ knows the witness. We express the fact that $\mathcal{P}$ know the witness in the above by saying that we can construct an emulator producing a transcript indistinguishable from the true transcript but also with a witness.

3.3.5 Public Coin

An argument $(Setup, \mathcal{P}, \mathcal{V})$ is called *public coin* if the verifier chooses his message uniformly at random and independently of the messages sent by the prover, i.e. The challenges correspond to the verifier's randomness ρ.

3.3.6 Perfect Special Honest-Verifier Zero-Knowledge

A *public coin* argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ has *Perfect Special Honest-Verifier Zero-Knowledge* property (SHVZK) for relation R if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive adversaries $\mathcal{A}$ we have :

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

The intuitive idea behind the definition of SHVZK is that no matter the choice of statement and witness (u, w), as long as $(\sigma, u, w) \in R$, then the transcript of the interaction between the prover and the verifier when run on input (σ, u, w) and (σ, u, ρ) respectively does not leak any information about the witness w. We express the fact that the transcript does not leak any information about the witness by saying that the transcript could have been produced without any knowledge about the witness. We formalize this as the existence of a simulator that runs in polynomial time and produces on input (σ, u, ρ) (and not w !) a transcript tr that is indistinguishable from the true transcript resulting from a real interaction between $\mathcal{P}$ and $\mathcal{V}$.

To simplify the notation in what follows, let $C_1 := \begin{bmatrix}\sigma\leftarrow Setup(1^\lambda),\\(u,w,\rho)\leftarrow\mathcal{A}(\sigma),\\tr\leftarrow\langle\mathcal{P}(\sigma,u,w),\mathcal{V}(\sigma,u,\rho)\rangle\end{bmatrix}$ and $C_2 := \begin{bmatrix}\sigma\leftarrow Setup(1^\lambda),\\(u,w,\rho)\leftarrow\mathcal{A}(\sigma),\\tr\leftarrow\mathcal{S}(\sigma,u,\rho)\end{bmatrix}$.

In practice instead of showing $P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1\right] = P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2\right]$ we will show : $P\left[\mathcal{A}(tr)=1\,|\,C_1\right] = P\left[\mathcal{A}(tr)=1\,|\,C_2\right]$ **(1)**,
which implies *Perfect Special Honest-Verifier Zero-Knowledge* in all the proofs below. If you want more details, here is formally why this is true:

$P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1\right] = P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1,(\sigma,u,w)\in R\right]P\left[(\sigma,u,w)\in R\right] +$
$P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1,(\sigma,u,w)\notin R\right]P\left[(\sigma,u,w)\notin R\right]$ **(2)**.

Now, the first term $P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1,(\sigma,u,w)\in R\right]$ is equal to $P\left[\mathcal{A}(tr)=1\,|\,C_1\right] = P\left[\mathcal{A}(tr)=1\,|\,C_2\right]$ by **(1)**. And we can write it back as $P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2,(\sigma,u,w)\in R\right]$.
The second term $P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1,(\sigma,u,w)\notin R\right]$ is equal to 0 and so is
$P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2,(\sigma,u,w)\notin R\right]$ so the term are equal.

Now we can substitute the two expressions we just found in **(2)** to obtain $P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_1\right] =$
$P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2,(\sigma,u,w)\in R\right]P\left[(\sigma,u,w)\in R\right] +$
$P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2,(\sigma,u,w)\notin R\right]P\left[(\sigma,u,w)\notin R\right] = P\left[(\sigma,u,w)\in R\wedge\mathcal{A}(tr)=1\,|\,C_2\right]$.

3.3.7 Efficiency Criteria

The different efficiency criteria we will consider to analyse zero-knowledge arguments performances are the followings:

- **The number of communications** from the Prover $\mathcal{P}$ to $\mathcal{V}$ in the protocol.
- **The number of bases** for any exponentiation performed in the protocol by $\mathcal{P}$ or $\mathcal{V}$.
- **The number of group exponentiations** performed by $\mathcal{P}$ and $\mathcal{V}$.
- **The number of multiplications in** $\mathbb{G}$ performed by $\mathcal{P}$ and $\mathcal{V}$.
- **The number of multiplications in** $\mathbb{Z}_p$ performed by $\mathcal{P}$ and $\mathcal{V}$.

The number of communications is an important criterion from a data efficiency point of view as it will indicate in the end the size of the proof that we will need to store and/or communicate.

The number of multiplications and exponentiations are also important criteria from a time efficiency point of view as they will be the major indicators of the time required to generate and verify one proof.

3.4 Forking Lemma

Suppose that we have a $(2\mu + 1)$-move *public-coin* argument with μ challenges, $x_1, \ldots, x_\mu$ in sequence. Let $x_i \geq 1$ for $i \in [1, \mu]$. We define the $(n_1, \ldots, n_\mu)$-tree of accepting transcripts as follows.
Consider $N := \prod_{i=1}^{\mu} n_i$ accepting transcripts with challenges in the following tree format. The tree has depth μ and N leaves. The root of the tree is labelled with the statement. Each node of depth $i < \mu$ has exactly n_i children, each labelled with a distinct value of the i-th challenge x_i.

Theorem :
Let $(Setup, \mathcal{P}, \mathcal{V})$ be a $(2\mu + 1)$-move, public coin interactive protocol. Let χ be a witness extraction algorithm that succeeds with probability $1 - \mu(\lambda)$ for some negligible function $\mu(\lambda)$ 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 $(Setup, \mathcal{P}, \mathcal{V})$ has witness-extended emulation.

We will use the *Forking Lemma* to prove *Computational Witness Extended Emulation.* If we have a $(2\mu+1)-$move *public-coin* argument with μ challenges $x_1, \ldots, x_\mu$ we can simply extract the witness by considering a tree with one level for each challenge and n_i children for each node of depth $i \in [1, \mu]$ representing each a different value for the i-th challenge x_i.
This will be clearer when we will make the *Computational Witness Extended Emulation* proofs later on.

3.5 Schnorr's Protocol

The Schnorr's Protocol is a protocol that allows a prover $\mathcal{P}$ to prove in a zero-knowledge fashion that he knows the witness x for the following relation :

$$R_0 \equiv \{(g, h \in \mathbb{G}; x \in \mathbb{Z}_p) : g^x = h\}.$$

We present first this protocol because it is simple and allows us to introduce the basics of zero-knowledge protocols and associated security proofs. The proof techniques used for the more complex protocols developed later on will use the same principles even though they will be longer.

To make the link with the previous definition we gave of the ternary relation, for the Schnorr's protocol the C.R.S. is $\sigma := g$, the statement is: $u := h$ and the witness is $w := x$.
The witness $w = \mathbf{a}$ is a valid witness for relation R_0 if $(\sigma, u, w) \in R_4$ or equivalently if :
$(g, h, x) \in \{(g, h, x) : g^x = h\}$.

3.5.1 Protocol

We first give a formal written description of the protocol and then a diagram to see more visually the exchanges in the protocol.

Schnorr's Protocol

- Input : $(g, h \in \mathbb{G}, x \in \mathbb{Z}_p)$,

- $\mathcal{P}$'s input : (g, h, x),
- $\mathcal{V}$'s input : (g, h).

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 1 : $\mathcal{P}$ picks uniformly at random a blinding vector s in $\mathbb{Z}_p$ and computes $S = g^s$.
- step 2 : $\mathcal{P}$ sends S to $\mathcal{V}$.
- step 3 : $\mathcal{V}$ picks a challenge y uniformly at random in $\mathbb{Z}_p$.
- step 4 : $\mathcal{V}$ sends y to $\mathcal{P}$.
- step 5 : $\mathcal{P}$ computes his answer as : $f = x + sy$.
- step 6 : $\mathcal{P}$ sends f to $\mathcal{V}$.
- step 7 : $\mathcal{V}$ accepts the proof if the following verification equation is verified :

$$g^f = hS^y.$$

3.5.2 Diagram

Schnorr's Protocol

Prover		**Verifier**
$s \xleftarrow{\$} \mathbb{Z}_p$		
$S := g^s$		
	$\xrightarrow{S}$	
		$y \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{y}$	
$f := x + sy$		
	$\xrightarrow{f}$	
		$g^f \stackrel{?}{=} hS^y$

3.6 Theorem 1

The Schnorr's protocol proof system presented in Section 3E for relation R_0 has perfect completeness, perfect special honest-verifier zero-knowledge and computational witness extended emulation.

In the followings subsections, we will show point by point that the Schnorr's protocol satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* properties for relation R_0.

3.6.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $x \in \mathbb{Z}_p$ for the relation $R_0 \equiv \{(g, h \in \mathbb{G}; x \in \mathbb{Z}_p) : h = g^x\}$. We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will

result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma,u,w), \mathcal{V}(\sigma,u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u,w) \leftarrow \mathcal{A}(\sigma), (\sigma,u,w) \in R_0\right] = 1.$

To see this, let's look at the verification equation that the verifier $\mathcal{V}$ checks in the protocol.
It is $g^f = hS^y$. If $\mathcal{P}$ follows honestly the protocol : $f = x + sy$ with $g^x = h$. We will thus have $g^f = g^{x+sy} = g^x(g^s)^y = hS^y$. The verification will thus be verified and the conversation will be accepted by $\mathcal{V}$ thereby showing the *Perfect Completeness* property.

3.6.2 Perfect Special Honest-Verifier Zero-knowledge

To show *Perfect Special Honest-Verifier Zero-Knowledge* (SHVZK) we build explicitly an efficient simulator that given the C.R.S. and a statement ($\sigma := g \in \mathbb{G}$ and $u := h \in \mathbb{G}$ respectively) produces indistinguishable transcript (in the sense that the transcript will have identical distribution) from a transcript resulting from the true interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. That is :

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

The simulator pseudo-code is as follows :

Schnorr's Simulator

- **Input :** $(g, h \in \mathbb{G})$,
- **Output :** transcript identically distributed to a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

- **step 1 :** Picks challenge y using $\mathcal{V}$'s randomness.
- **step 2 :** Picks f uniformly at random in $\mathbb{Z}_p$.
- **step 3 :** Computes $S = (g^f h^{-1})^{y^{-1}}$.
- **step 4 :** Outputs $(S; y; f)$.

The value f produced by an honest prover interacting with an honest verifier will be a random element. Since y is picked using $\mathcal{V}$'s randomness it will be identically to a real proof with honest $\mathcal{V}$. Finally, the simulated S is fully defined by equation $g^f = hS^y$. Therefore, the transcripts produced by our simulator defined by the above pseudo-code will be identically distributed to the transcript resulting from a true interaction between an honest prover $\mathcal{P}$ and an honest verifier $\mathcal{V}$. In addition, the simulator only does simple arithmetic operations and is thus efficient, thereby showing *Perfect Special Honest-Verifier Zero-Knowledge*.

3.6.3 Computational Witness Extended-Emulation

To show *Computational Witness Extended-Emulation*, we build an efficient extractor χ that given a polynomial number of $N^{Schnorr} = 2$ transcripts with 2 different values of challenge y : y_1, y_2 of an interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ resulting in an accepting conversation for a statement h extracts a valid witness x for the relation $R_0 \equiv \{(g, h \in \mathbb{G}; x \in \mathbb{Z}_p) : h = g^x\}$.
This allows to use the *Forking Lemma* (3.4) to show the *computational witness extended emulation property* (3.3.4).

Since both conversations are accepting we have $g^{f_i} = hS^{y_i}$, $i \in \{1,2\}$.
The idea now will simply be to do linear combinations in the exponent of the two equations. We will multiply each equation $i \in \{1,2\}$ by some coefficient ν_i in the exponent.
Consider coefficients $\nu_1, \nu_2 \in \mathbb{Z}_p$ and linear combination : $\prod_{i=1}^2 \left[g^{f_i}\right]^{\nu_i} = \prod_{i=1}^2 [hS^{y_i}]^{\nu_i} \Leftrightarrow g^{\sum_{i=1}^2 f_i\nu_i} = h^{\sum_{i=1}^2 \nu_i} S^{\sum_{i=1}^2 y_i\nu_i}$.
Defining ν_1, ν_2 by: $\sum_{i=1}^2 \nu_i = 1$ and $\sum_{i=1}^2 \nu_i y_i = 0$ leads to $h = g^x$ for $x = \sum_{i=1}^2 \nu_i f_i$, which is thus a valid witness.

Using the *Forking Lemma* presented in section 3.4 this implies *computational witness extended-emulation property* (3.3.4), which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The Schnorr's protocol is in 3 moves and has one challenge $x_1 = y$, hence the parameter μ in the statement of the *Forking Lemma* is equal to 1. The corresponding tree of accepting transcripts has a depth equal to $\mu = 1$ and is just composed of the root connected to two leaves (hence $n_1 = 2$) associated with the two different values of the challenge y: y_1, y_2.
Therefore the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = n_1 = 2 = N^{Schnorr}$, which is clearly bounded above by a polynomial in the parameter λ (the constant polynomial 2 for example). In addition, χ is a polynomial time algorithm since computing the ν_i amounts to solving a system of linear equations in $\mathbb{Z}_p$: $\begin{bmatrix} y_1 & y_2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, which can be done efficiently and $x = \sum_{i=1}^{2} \nu_i f_i$, which is a polynomial time computation. Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute ν_1, ν_2, which occurs only when the determinant of the matrix $\begin{bmatrix} y_1 & y_2 \\ 1 & 1 \end{bmatrix}$ is 0, or equivalently when $y_1 - y_2$ is zero (the linear system of equations has no solution then) that is when $y_1 = y_2$, which happens only with negligible probability $(1/p)$ since y_1 and y_2 are picked uniformly at random in $\mathbb{Z}_p$.

Chapter 4

Notations

In this section, we introduce some of the notations that will be used throughout this paper.

We denote $\mathbb{G}$ as the cyclic group of prime order p, while $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ denote the ring of integers modulo p. Given a natural number $n \geq 1$, $\mathbb{Z}_p^n$ and $\mathbb{G}^n$ denote the n-dimensional vector spaces over $\mathbb{Z}$ and $\mathbb{G}$ respectively.

Group elements and thus commitments are represented by capitalized letters, while blinding terms are represented by Greek letters, for example, $S = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l} \in \mathbb{G}$ is a multi-Pedersen commitment over the values $v_1, \ldots, v_l$ with blinding factor α.

Bold font letters are used for vector elements, for instance, $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ is an n-dimensional vector with elements $(a_1, \ldots, a_n) \in \mathbb{F}$. We denote also sometimes alternatively $\mathbf{a}$ by $(a_j)_{j=1}^n$.
Given $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$, we denote $a_1 \ldots a_n$ as the product of all elements of the vector $\mathbf{a}$: $\prod_{j=1}^n a_j$.
Given a scalar $c \in \mathbb{Z}_p^n$ and a vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, we denote $c \cdot \mathbf{a}$ as the vector $(c \cdot a_i)_{i=1}^n$.

Given $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ and $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}^n$, $\mathbf{a} \circ \mathbf{b}$ denotes the element-wise product of the two vectors (also sometimes called Hadamard product): $(a_i \cdot b_i)_{i=1}^n$.
We denote the dot-product between the two vectors $\mathbf{a}$ and $\mathbf{b}$ as : $< \mathbf{a}, \mathbf{b} >= \sum_{i=1}^n a_i \cdot b_i$.
$\mathbf{a^b}$ is a shortcut notation for the product $\prod_{j=1}^n a_j^{b_j}$.

On the other hand, given $c \in \mathbb{Z}_p$ and n a natural number, we denote $\mathbf{c}^n$ as the vector containing the first n powers of c, i.e.

$$\mathbf{c}^n = (c^0, c^1, \ldots, c^{n-1}) = (c^{i-1})_{i=1}^n.$$

Finally, assume $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ is an n-dimensional vector and $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{F}^m$ is an m-dimensional vector. Then, $(\mathbf{a}||\mathbf{b})$ denotes the $(n+m)$-dimensional vector :

$$(\mathbf{a}||\mathbf{b}) = (a_1, \ldots, a_n, b_1, \ldots, b_m) \in \mathbb{F}^{n+m}$$

- For instance: using our notation, with $\mathbf{a_L^{(j)}} = (\mathbf{a_L}[1], \ldots, \mathbf{a_L}[m]), \mathbf{a_R^{(j)}} = (\mathbf{a_R}[1], \ldots, \mathbf{a_R}[m]) \in \mathbb{Z}_p^m$ and $\mathbf{g_L^{(j)}} = (\mathbf{g_L^{(j)}}[1], \ldots, \mathbf{g_L^{(j)}}[m]), \mathbf{g_R^{(j)}} = (\mathbf{g_R^{(j)}}[1], \ldots, \mathbf{g_R^{(j)}}[m]) \in \mathbb{G}^m$, for $j \in \{1, \ldots, l\}$, then the expression for the multi-Pedersen commitment you can find in Chapter 7:

$$A = h^{\alpha}(\mathbf{g_1}^{\mathbf{a_L^{(1)}}} \ldots \mathbf{g_l}^{\mathbf{a_L^{(l)}}})(\mathbf{h_1}^{\mathbf{a_R^{(1)}}} \ldots \mathbf{h_l}^{\mathbf{a_R^{(l)}}})$$

stands for :

$$A = h^{\alpha} \left[\prod_{j=1}^{l} \prod_{i=1}^{m} \mathbf{g_j}[i]^{\mathbf{a_L^{(j)}}[i]} \right] \left[\prod_{j=1}^{l} \prod_{i=1}^{m} \mathbf{h_j}[i]^{\mathbf{a_R^{(j)}}[i]} \right] \in \mathbb{G}$$

, which is a single group element..

Chapter 5

Goals

This thesis is dedicated to the design of zero-knowledge arguments for each of the following relations:

Goals

$$
\begin{aligned}
R_1 \equiv \{&(g_1, \ldots, g_l, h \in \mathbb{G}, \mathbf{V} = (V_1, \ldots, V_m) \in \mathbb{G}^m; \\
&(v_1^{(1)}, \ldots, v_1^{(m)}), \ldots, (v_l^{(1)}, \ldots, v_l^{(m)}), \gamma = (\gamma_1, \ldots, \gamma_m) \in \mathbb{Z}_p^m) : \\
&V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\}, \forall i \in [1,l], j \in [1,m]\} \\
R_2 \equiv \{&(g_1, \ldots, g_l, h \in \mathbb{G}, \mathbf{V} = (V_1, \ldots, V_m) \in \mathbb{G}^m, n \in \mathbb{N}; \\
&(v_1^{(1)}, \ldots, v_1^{(m)}), \ldots, (v_l^{(1)}, \ldots, v_l^{(m)}), \gamma = (\gamma_1, \ldots, \gamma_m) \in \mathbb{Z}_p^m) : \\
&\left[V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}} \wedge \sum_{i=1}^{l} v_i \in [0, 2^n - 1]\right]\} \\
R_3 \equiv \{&(h, g_1, \ldots, g_l \in \mathbb{G}, j \in \{1, \ldots, l\}, b \in \{0,1\}, V \in \mathbb{G}; \gamma \in \mathbb{Z}_p, (v_1, \ldots, v_l) \in \mathbb{Z}_p^{l-1}) : \\
&V = h^{\gamma} g_1^{v_1} \ldots g_l^{v_l} \wedge v_j = b\}
\end{aligned}
$$

- Relation R_1 formalizes in mathematical terms the notion that a batch of votes commitments satisfies the **0-1 Property**, that is as explained in chapter 2, the property that each of the votes on which we commit in the batch contains only values that are either a 0 or a 1.
- Relation R_2 formalizes in mathematical terms the notion that a batch of votes commitments satisfies the min-max **K-selection Property** which is, as explained in chapter 2, the property that each ballot of votes on which we commit in the batch contains at least a certain minimum of ones (selections) and at most some maximum of ones (selections).
- Relation R_3 allows doing a **Partial Opening** as explained in chapter 2. In a partial opening, the prover will send one selection of the ballot to the verifier ($b \in \{0,1\}$. The verifier verifies that the value b sent by the prover is indeed the one on which he had committed previously in the multi-commitment (V). The verification is formalized through relation R_3.

In several cases the amount of data required by the zero-knowledge proofs associated with the audit trials can be a limiting factor to conduct Risk Limiting Audits.
We want to be able to construct protocols for each of the three relations presented above that generate compact proofs. In particular, we want the proofs to have a logarithmic size.
For this, we will some of the building blocks presented in the next which allow having proofs of logarithmic size with respect to the witness size.

Chapter 6

Building Blocks

In this section, we describe the main building blocks we used in order to construct later on our cryptographic zero-knowledge protocols.

6.1 Extended-Schnorr Argument

In this section, we present a simple argument that allows a prover $\mathcal{P}$ to prove to a verifier $\mathcal{V}$ efficiently (logarithmic complexity) that he knows a witness $\mathbf{a}$ for the following relation without the zero-knowledge property :

$$R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\}.$$

Throughout this section, we assume without loss of generality, that n is a power of 2 like in the "Bulletproofs" paper [4] to make things simpler (we can simply pad the input to make n a power of 2, for example, if it is not the case). This paper is made to be more didactic, so we chose to put this protocol here because it is of "intermediate difficulty" between the basic Schnorr's protocol and the later protocols which will be more complex. In addition, it will be used as a sub-protocol in several of the protocols presented later.

For the Extended-Schnorr Argument, the C.R.S. is $\sigma := \mathbf{g}$, the statement is $u := P$ and the witness is $w := \mathbf{a}$.

We chose to give the protocol the name of "Extended-Schnorr" argument even if it is not very accurate because the Schnorr's Protocol has the zero-knowledge property, contrary to this protocol as we will see.

6.1.1 Protocol

The protocol below is greatly inspired as a lot of the protocols in this paper by the paper "Bulletproofs" [4] and in particular their inner product argument which we will also present in this paper. We use the same notations and presentations as in the "Bulletproofs" protocol.

We give a formal written description of the protocol and then a diagram to see visually all the exchanges in the protocol.

Extended-Schnorr Argument Protocol

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

- $\mathcal{P}$'s input : $(\mathbf{g}, P, \mathbf{a})$,
- $\mathcal{V}$'s input : $(\mathbf{g}, P)$.

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 0 : If $n = 1$: $\mathcal{P}$ sends $\mathbf{a}$ to $\mathcal{V}$. $\mathcal{V}$ accepts the proof if $P = \mathbf{g}^{\mathbf{a}}$, $\mathcal{V}$ rejects it otherwise.
else : go to step 1.
- step 1 : $\mathcal{P}$ computes :

- $n' = n/2$,
- $L = \mathbf{g}_{[\mathbf{n}':]}^{\mathbf{a}_{[:\mathbf{n}']}}$,
- $R = \mathbf{g}_{[:\mathbf{n}']}^{\mathbf{a}_{[\mathbf{n}':]}}$.

- step 2 : $\mathcal{P}$ sends to $\mathcal{V}$: L, R.
- step 3 : $\mathcal{V}$ picks x uniformly at random in $\mathbb{Z}_p^*$.
- step 4 : $\mathcal{V}$ sends to $\mathcal{P}$: x.
- step 5 : $\mathcal{P}$ and $\mathcal{V}$ both compute :

- $\mathbf{g}' := \mathbf{g}_{[n':]}^{x} \circ \mathbf{g}_{[:n']}^{x^{-1}} \in \mathbb{G}^{n'}$,
- $P' := L^{x^2} P R^{x^{-2}} \in \mathbb{G}$.

- step 6 : $\mathcal{P}$ computes : $\mathbf{a}' := \mathbf{a}_{[n':]} \cdot x^{-1} + \mathbf{a}_{[:n']} \cdot x \in \mathbb{Z}_p^{n'}$.
- step 7 : Recursively run the proof (i.e. go back to step 0) on input :

$$(\mathbf{g}' \in \mathbb{G}^{n'}, P' \in \mathbb{G}; \mathbf{a}' \in \mathbb{Z}_p^{n'}).$$

The "Extended-Schnorr Argument" is a recursive protocol. At each recursion step it reduces the difficulty of the argument to a simpler argument by reducing the dimension n by a factor 2 to $n' = n/2$.
The recursion ends up when the dimension n is equal to 1. In this case, the argument is easy and is of constant size by simply asking the prover $\mathcal{P}$ to send in clear $a \in \mathbb{Z}_p$ (a is not written with bold letters since it is simply a scalar for $n = 1$) to the verifier $\mathcal{V}$ and asking the verifier to check whether $P = g^a$.

Notice that since a is sent in clear the "Extended-Schnorr Argument" protocol does not have the *perfect special honest-verifier zero-knowledge property.*

6.1.2 Diagram

Extended-Schnorr's Protocol

Prover		**Verifier**
$n' := n/2$		$n' := n/2$
$L := \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}}$		
$R := \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}}$		
	$\xrightarrow{L, R}$	
		$x \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{x}$	
$\mathbf{g}' := \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}$		$\mathbf{g}' := \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}$
$P' := L^{x^2} P R^{x^{-2}}$		$P' := L^{x^2} P R^{x^{-2}}$
$\mathbf{a}' := \mathbf{a}_{[:n']} \cdot x + \mathbf{a}_{[n':]} \cdot x^{-1}$		
$\vdots$		$\vdots$
$(n' = 1)$		$(n' = 1)$
	$\xrightarrow{\mathbf{a}'}$	
		$\mathbf{g}'^{\mathbf{a}'} \stackrel{?}{=} P'$

6.1.3 Complexity

The "Extended-Schnorr" argument has the following complexities :

- **# Communications :** $2log_2(n) + 1$ elements :
  - L and R at each of the $log_2(n)$ rounds : $2log_2(n)$ elements in $\mathbb{G}$
  - $\mathbf{a}$ at the last round : 1 element in $\mathbb{Z}_p$
- **# Bases :** n bases :
  - $\mathbf{g} \in \mathbb{G}^n$
- **# Exponentiations in $\mathbb{G}$:**
  - **Prover :** $4n + 2log_2(n) - 4$ exponentiations in $\mathbb{G}$
  - **Verifier :** $2n + 2log_2(n) - 1$ exponentiations in $\mathbb{G}$
- **# Exponentiations in $\mathbb{Z}_p$**
  - **Prover:** $3log_2(n)$ exponentiations in $\mathbb{Z}_p$
  - **Verifier:** $3log_2(n)$ exponentiations in $\mathbb{Z}_p$

Table 6.1 shows the complexity to compute each element for one round of the protocol otherwise specified by *only once* which means that the element is computed only once. At the first round, $n' = \frac{n}{2}$ and n' is successively divided by 2 at each following round. Using the geometric formula $\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1}$, we obtain $\frac{n}{2} + \frac{n}{4} + \ldots + 2 + 1 = \sum_{i=0}^{log_2(n)-1} 2^i = \frac{2^{log_2(n)-1+1}-1}{2-1} = n - 1$. This formula allows us to compute the total number of exponentiations for all the rounds.

- **# Multiplications in $\mathbb{G}$:**
  - **Prover :** $3n-3$ multiplications :
    - $(n-1)-log_2(n)$ multiplications for each of L, R : $2n-2log_2(n)-2$
    - $\mathbf{g}'$: $n-1$ multiplications
    - P': $2log_2(n)$ multiplications
  - **Verifier :** $n+2log_2(n)-1$ multiplications :
    - $\mathbf{g}'$: $n-1$ multiplications
    - P': $2log_2(n)$ multiplications
- **# Multiplications in $\mathbb{Z}_p$:**
  - **Prover :** $2n-2$ multiplications for $\mathbf{a}'$
  - **Verifier :** 0 multiplication

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
L	n'		n'	
R	n'		n'	
x^{-1}		1		1
x^2		1		1
x^{-2}		1		1
g'	2n'			2n'
P'	2			2
Verif Eq			1 (only once)	
TOTAL	$4n+2log_2(n)-4$	$3log_2(n)$	$2n+2log_2(n)-1$	$3log_2(n)$

Table 6.1: Complexity of Extended-Schnorr's Protocol

6.2 Theorem 2

The "Extended-Schnorr" argument presented in section 6A for relation R_4 has perfect completeness and computational witness extended-emulation for either extracting a nontrivial discrete logarithm relation between $\mathbf{g}$ *or extracting a valid witness* $\mathbf{a}$.

In the followings subsections, we will show point by point that the "Extended-Schnorr" protocol satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* for relation R_4.

6.2.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $\mathbf{a} \in \mathbb{Z}_p^n$ for the relation $R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\}$. We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma,u,w), \mathcal{V}(\sigma,u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u,w) \leftarrow \mathcal{A}(\sigma), (\sigma,u,w) \in R_4\right] = 1$.

To see this, let's look at the verification equation that the verifier $\mathcal{V}$ checks at every recursion step.
Assume that $n = 2^\tau$: there is τ recursive rounds in the protocol until $\frac{n}{2^\tau} = 1$.

If $n = 1$, there is no recursive round, the input is $(\mathbf{g}, P; \mathbf{a})$ and $\mathcal{V}$ accepts the proof as $\mathbf{a}$ is a valid witness such that $P = \mathbf{g}^{\mathbf{a}}$.

If $n > 1$, the recursively computed input at round r is $(\mathbf{g_r}, P_r, \mathbf{a_R})$ with:

- $\mathbf{g_r} = \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]}{}^{x_r} \circ \mathbf{g_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{x_r^{-1}} = (\mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+1]}{}^{x_r} \cdot \mathbf{g_{r-1}}_{[1]}{}^{x_r^{-1}}, \ldots, \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^{r-1}}}]}{}^{x_r} \cdot \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{x_r^{-1}})$

- $P_r = L_r^{x_r^2} P_{r-1} R_r^{x_r^{-2}} = (\mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]}{}^{\mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{x_r^2}}) P_{r-1} (\mathbf{g_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]}{}^{x_r^{-2}}})$
  $= (\prod_{i=1}^{\frac{n}{2^r}} \mathbf{g_{r-1}}_{[\mathbf{i}+\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{\mathbf{a_{r-1}}_{[\mathbf{i}]}{}^{x_r^2}}) P_{r-1} (\prod_{i=1}^{\frac{n}{2^r}} \mathbf{g_{r-1}}_{[\mathbf{i}]}{}^{\mathbf{a_{r-1}}_{[\mathbf{i}+\frac{\mathbf{n}}{\mathbf{2^r}}]}{}^{x_r^{-2}}})$

- $\mathbf{a_R} = \mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} \cdot x_r^{-1} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r$

Knowing that $\mathbf{a_R} \cdot x_r = (\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} \cdot x_r^{-1} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r) \cdot x_r = \mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r^2$ and that $\mathbf{a_R} \cdot x_r^{-1} = (\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} \cdot x_r^{-1} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r) \cdot x_r^{-1} = \mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} \cdot x_r^{-2} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}$, we thus have:

$$
\begin{aligned}
\mathbf{g_r}^{\mathbf{a_R}} &= \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]}^{\mathbf{a_R} \cdot x_r} \circ \mathbf{g_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}^{\mathbf{a_R} \cdot x_r^{-1}} \\
&= \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r^2} \circ \mathbf{g_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}:]} \cdot x_r^{-2} + \mathbf{a_{r-1}}_{[:\frac{\mathbf{n}}{\mathbf{2^r}}]}} \\
&= (\mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+1]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+1]} + \mathbf{a_{r-1}}_{[1]} \cdot x_r^2} \cdot \mathbf{g_{r-1}}_{[1]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+1]} \cdot x_r^{-2}}, \ldots, \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^{r-1}}}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}]} \cdot x_r^2} \cdot \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^{r-1}}}]} \cdot x_r^{-2} + \mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^{r-1}}}]}}) \\
&= (\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]} + \mathbf{a_{r_{i-1}}}_{[\mathbf{i}]} x_r^2}) \cdot (\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]} \cdot x_r^{-2} + \mathbf{a_{r-1}}_{[\mathbf{i}]}}) \\
&= \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}^{\mathbf{a_{r_{i-1}}}_{[\mathbf{i}]} x_r^2})}_{L_r^{x_r^2}} \cdot \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}} \cdot \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\mathbf{i}]}})}_{P_{r-1}} \cdot \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]} \cdot x_r^{-2}})}_{R_r^{x_r^2}} \\
&= L_r^{x_r^2} P_{r-1} R_r^{x_r^{-2}} \\
&= P_r
\end{aligned}
$$

P_{r-1} has indeed the form $\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}} \cdot \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\mathbf{i}]}} = \prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^{r-1}}}} \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\mathbf{i}]}} = \mathbf{g_{r-1}}^{\mathbf{a_{r-1}}}$:

- $P_0 = \mathbf{g^a}$ $(\mathbf{g_0} = \mathbf{g}, \mathbf{a_0} = \mathbf{a})$
- $P_1 = (\mathbf{g_0}_{[\frac{\mathbf{n}}{\mathbf{2}}:]}^{\mathbf{a_0}_{[:\frac{\mathbf{n}}{\mathbf{2}}]}}) x_1^2 P_0 (\mathbf{g_0}_{[:\frac{\mathbf{n}}{\mathbf{2}}]}^{\mathbf{a_0}_{[\frac{\mathbf{n}}{\mathbf{2}}:]}}) x_1^{-2} = (\prod_{i=1}^{\frac{\mathbf{n}}{\mathbf{2}}} \mathbf{g}_{[\frac{\mathbf{n}}{\mathbf{2}}+\mathbf{i}]}^{\mathbf{a}_{[\frac{\mathbf{n}}{\mathbf{2}}+\mathbf{i}]}}) x_1^2 (\prod_{i=1}^{n} \mathbf{g}_{[\mathbf{i}]}^{\mathbf{a}_{[\mathbf{i}]}}) (\prod_{i=1}^{\frac{\mathbf{n}}{\mathbf{2}}} \mathbf{g}_{[\mathbf{i}]}^{\mathbf{a}_{[\mathbf{i}]}}) x_1^{-2} \longrightarrow P_0 = \mathbf{g^a}$
- $P_2 = (\mathbf{g_1}_{[:\frac{\mathbf{n}}{\mathbf{4}}]}^{\mathbf{a_1}_{[:\frac{\mathbf{n}}{\mathbf{4}}]}}) x_2^2 P_1 (\mathbf{g_1}_{[\frac{\mathbf{n}}{\mathbf{4}}:]}^{\mathbf{a_1}_{[\frac{\mathbf{n}}{\mathbf{4}}:]}}) x_2^{-2} = L_2^{x_2^2} (\prod_{i=1}^{\frac{\mathbf{n}}{\mathbf{2}}} \mathbf{g}_{[\frac{\mathbf{n}}{\mathbf{2}}+\mathbf{i}]}^{\mathbf{a}_{[\frac{\mathbf{n}}{\mathbf{2}}+\mathbf{i}]} \cdot x_1^2 + \mathbf{a}_{[\mathbf{i}]}} \cdot \mathbf{g}_{[\mathbf{i}]}^{\mathbf{a}_{[\mathbf{i}]} \cdot x_1^{-2} + \mathbf{a}_{[\mathbf{i}]}}) R_2^{x_2^{-2}} \longrightarrow P1 = \mathbf{g_1}^{\mathbf{a_1}}$
  with $\mathbf{g_1} = (\mathbf{g_0}_{[\frac{\mathbf{n}}{\mathbf{2}}+1]}^{x_1} \cdot \mathbf{g_0}_{[1]}^{x_1^{-1}}, \ldots, \mathbf{g_0}_{[\mathbf{n}]}^{x_1} \cdot \mathbf{g_0}_{[\frac{\mathbf{n}}{\mathbf{2}}]}^{x_1^{-1}})$ and $\mathbf{a_1} = \mathbf{a_0}_{[\frac{\mathbf{n}}{\mathbf{2}}:]} \cdot x_1^{-1} + \mathbf{a_0}_{[:\frac{\mathbf{n}}{\mathbf{2}}]} \cdot x_r$
- ...
- $P_r = \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]}^{\mathbf{a_{r_{i-1}}}_{[\mathbf{i}]} x_r^2})}_{L_r^{x_r^2}} \cdot \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^{r-1}}}} \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\mathbf{i}]}})}_{P_{r-1}} \cdot \underbrace{(\prod_{\mathbf{i}=1}^{\frac{\mathbf{n}}{\mathbf{2^r}}} \mathbf{g_{r-1}}_{[\mathbf{i}]}^{\mathbf{a_{r-1}}_{[\frac{\mathbf{n}}{\mathbf{2^r}}+\mathbf{i}]} \cdot x_r^{-2}})}_{R_r^{x_r^2}} \longrightarrow P_{r-1} = \mathbf{g_{r-1}^{a_{r-1}}}$

The verification equation $P_r = \mathbf{g_r}^{\mathbf{a_R}}$ is thus always verified at all rounds if P knows a valid witness $\mathbf{a}$ and if P and $\mathcal{V}$ follow honestly the protocol, which proofs the *Perfect Completeness* of the argument.

6.2.2 Computational Witness Extended-Emulation

To show *Computational Witness Extended-Emulation*, we build recursively an efficient extractor χ that given a polynomial number of $N^{ES} = 3^\tau = 3^{log_2(n)} = 2^{log_2(3) \cdot log_2(n)} = 2^{log_2(n^{log_2(3)})} = n^{log_2(3)} \approx n^{1.58}$ transcripts with 3 different values of the challenge $x : x_1, x_2, x_3$ at each recursion step of an interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ resulting in an accepting conversation for a statement P extracts a valid witness $\mathbf{a}$ for the relation $R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g^a}\}$.
This allows to use the *Forking Lemma* (3.4) to show the *computational witness extended emulation property* (3.3.4).

Assume n is a power of 2, i.e. $n = 2^\tau$, $\tau \in \mathbb{N}$.
We first prove we can extract a valid witness for $\tau = 0$ $(n = 1)$ for the relation R_4. Then, for $n > 1$, $\tau \geq 1$, we show that if we can extract a valid witness for $n' = 2^{\tau-1}$, then we can extract a valid witness for $n = 2n' = 2^\tau$ for relation R_4.

If $n = 1$ $(\tau = 0)$, the soundness is trivially checked since $\mathcal{P}$ sends the witness $\mathbf{a} \in \mathbb{Z}_p$ to $\mathcal{V}$ which accepts the proof if it is a valid witness, i.e. if $\mathbf{g^a} = P$. We can "extract" a valid witness: $\mathbf{a}$ which $\mathcal{P}$ send to us in clear.

If $n = 2^\tau > 1$ ($\tau \geq 1$), assume we have an extractor $\chi_{\tau-1}$ that can extract a valid witness for $n' = 2^{\tau-1}$ for relation R_4, then we construct from it an extractor χ_τ that can extract a valid witness for $n = 2n' = 2^\tau$ for relation R_4.

The extractor works by rewinding the prover as a subroutine with four different values of the challenge x: $x_1, x_2, x_3 \in \mathbb{Z}_p^*$, with $x_i \neq \pm x_j$, $i \neq j$.
Using $\chi_{\tau-1}$, we can extract witness $\mathbf{a'}$ such that $\mathbf{g'}^{\mathbf{a'}} = P'$ (recall $g' \in \mathbb{G}^{n'}$ and $\mathbf{a} \in \mathbb{Z}_p$).
Doing this, for all 3 transcripts obtained by rewinding the prover, we obtained 3 witnesses $\{\mathbf{a'}_i\}_{i=1}^3$ such that : $\mathbf{g'_i}^{\mathbf{a'}_i} = P'_i$, or equivalently : $(\mathbf{g}_{[n':]}^{x_i} \circ \mathbf{g}_{[:n']}^{x_i^{-1}})^{\mathbf{a'}_i} = L^{x_i^2} P R^{x_i^{-2}}$ since indeed P' and $\mathbf{g'}$ are defined as :

- $P' := L^{x^2} P R^{x^{-2}}$,
- $\mathbf{g'} := \mathbf{g}_{[n':]}^{x} \circ \mathbf{g}_{[:n']}^{x^{-1}}$.

.

We can make linear combinations in the exponents of the three previous equations in the exponents multiplying each of the equations i by some coefficient ν_i, with $\nu_1, \nu_2, \nu_3 \in \mathbb{Z}_p$ such that $\sum_{i=1}^3 x_i^2 \nu_i = 0$, $\sum_{i=1}^3 \nu_i = 1$ and $\sum_{i=1}^3 x_i^{-2} \nu_i = 0$.
In this way, we obtain:
$\prod_{i=1}^3 [P'_i]^{\nu_i} = \prod_{i=1}^3 \left[(\mathbf{g}_{[n':]}^{x_i} \circ \mathbf{g}_{[:n']}^{x_i^{-1}})^{\mathbf{a'}_i} \right]^{\nu_i} = \mathbf{g}_{[n':]}^{\sum_{i=1}^3 \nu_i x_i \cdot \mathbf{a'}_i} \circ \mathbf{g}_{[:n']}^{\sum_{i=1}^3 \nu_i x_i^{-1} \cdot \mathbf{a'}_i}$
$\Leftrightarrow \prod_{i=1}^3 \left[L^{x_i^2} P R^{x_i^{-2}} \right]^{\nu_i} = L^{\sum_{i=1}^3 x_i^2 \nu_i} P^{\sum_{i=1}^3 \nu_i} R^{\sum_{i=1}^3 x_i^{-2} \nu_i} = L^0 P^1 R^0 = P.$
Hence, we can write $P = \mathbf{g}^{\mathbf{a}}$, with $\mathbf{a} := (\sum_{i=1}^3 \nu_i x_i \cdot \mathbf{a'}_i || \sum_{i=1}^3 \nu_i x_i^{-1} \cdot \mathbf{a'}_i)$.

In this way χ_τ extracts a valid witness $\mathbf{a}$ for relation R_4 for $n = 2^\tau$.
Since we can extract a valid witness for $n = 1 = 2^0$, this implies that we can extract a valid witness for $n = 2 = 2^1$. Now since we can extract a valid witness for $n = 4 = 2^2$ this implies we can extract a valid witness for $n = 8 = 2^3$ and so forth. Hence, by induction, we can extract a valid witness for all $\tau \geq 0$, that is for all n power of 2.

Using the *Forking Lemma* presented in section 3.4 this implies *computational witness extended-emulation property* (3.3.4), which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The "Extended-Schnorr" protocol is in $2\tau + 1 = 2log_2(n) + 1$ moves and has $\tau = log_2(n)$ challenges: $x_1, \ldots, x_{log_2(n)}$ (one challenge at each recursion round), hence the parameter μ in the statement of the *Forking Lemma* is equal to $\tau = log_2(n)$. The corresponding tree of accepting transcripts has a depth equal to $\mu = \tau = log_2(n)$ (the depth of the tree is the recursion depth), each level of the tree corresponds to a recursion step and each node in the tree is associated to the three different values of the challenge x_i at each level (hence $n_i = 3 \ \forall i \in [1, \mu]$).
Therefore the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = \prod_{i=1}^{\mu} 3 = 3^\mu = N^{ES} \approx n^{1.58}$, which is clearly bounded above by a polynomial in the parameter λ (the polynomial n^2 for instance). In addition, χ is in polynomial time since computing the $\{\nu_i\}_{i=1}^2$ for each node in the tree amounts to solving a system of linear equations $\begin{bmatrix} 1 & 1 & 1 \\ x_1^{-2} & x_2^{-2} & x_3^{-2} \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ in $\mathbb{Z}_p$, which can be done efficiently and $x = \sum_{i=1}^3 \nu_i f_i$, which also a computation in polynomial time. Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute ν_1, ν_2, ν_3 for one of the nodes, which occurs only when the determinant of the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1^{-2} & x_2^{-2} & x_3^{-2} \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix}$ is 0, which happens only with negligible probability since each of the x_i are picked uniformly at random.[1]

[1]The determinant of the matrix is $det = (x_2^{-2} \cdot x_3^2 - x_2^2 \cdot x_3^{-2}) - (x_1^{-2} x_3^2 - x_1^2 x_3^{-2}) + (x_1^{-2} x_2^2 - x_1^2 x_2^{-2})$ and hence multiplying the equation $det = 0$ by $x_1^2 x_2^2 x_3^2$, it is equivalent to $(x_2^2 - x_3^2) x_1^4 + (x_3^4 - x_2^4) x_1^2 + (x_3^2 x_2^4 - x_2^2 x_3^4) x_1^0 = 0 \Leftrightarrow p(x_1) = 0$, where $p(X) := (x_2^2 - x_3^2) X^4 + (x_3^4 - x_2^4) X^2 + (x_3^2 x_2^4 - x_2^2 x_3^4) \in \mathbb{Z}_p[X]$. Using the *Schwartz-Zippel Lemma*, since x_1 is picked uniformly at random in $\mathbb{Z}_p$, the probability of $p(x_1)$ to be null is negligible and so is the probability of the determinant to be zero and the probability of the extractor to fail.

6.3 "Bulletproofs" Inner Product Argument

In this section, we present the "Bulletproofs" Inner Product Argument [4]. The paper [4] where it is introduced has had a huge impact on this thesis since the "Bulletproofs" Inner Product Argument is the main building block used to achieve logarithmic complexity proofs size and is an inspiration for this work.

The "Bulletproofs" Inner Product Argument is an efficient proof system for the following relation :

$$R_5 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a}, \mathbf{b} \in \mathbb{G}^n : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c =< \mathbf{a}, \mathbf{b} >)\}$$

In fact, as explained in the "Bulletproofs" paper they design the protocol for the following relation :

$$R_6 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}; \mathbf{a}, \mathbf{b} \in \mathbb{G}^n : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{<\mathbf{a},\mathbf{b}>})\}$$

Proving relation R_5 can be reduced to proving the relation R_6 using the following protocol (protocol 1) which uses the proof system for relation R_5 to build a proof system for relation R_6.

For the protocol 1 designed to prove relation R_5, the C.R.S. is $\sigma := (\mathbf{g}||\mathbf{h})$, the statement is $u := (P, c)$ and the witness is $w := (\mathbf{a}||\mathbf{b})$.

For the protocol 2 designed to prove relation R_6, the C.R.S. is $\sigma := (\mathbf{g}||\mathbf{h}||u)$, the statement is $u := P$ and the witness is $w := (\mathbf{a}||\mathbf{b})$.

6.3.1 Reduction Protocol

We first give a formal written description of the protocol and then a diagram to see visually all the exchanges in the reduction protocol.

Reduction Protocol (protocol 1) :

- **Input :** $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)$,
  - $\mathcal{P}$'s input : $(\mathbf{g}, \mathbf{h}, u, P, c; \mathbf{a}, \mathbf{b})$,
  - $\mathcal{V}$'s input : $(\mathbf{g}, \mathbf{h}, u, P, c)$.
- **Output :** $\mathcal{V}$ accepts or rejects the proof.

- **step 1 :** $\mathcal{V}$ picks x uniformly at random in $\mathbb{Z}_p^*$.
- **step 2 :** $\mathcal{V}$ sends x to $\mathcal{P}$.
- **step 3 :** $\mathcal{P}$ and $\mathcal{V}$ computes $P' := Pu^{x \cdot c}$.
- **step 4 :** Run protocol 2 on input $(\mathbf{g}, \mathbf{h}, u^x, P'; \mathbf{a}, \mathbf{b})$.

6.3.2 Diagram

Reduction Protocol

Prover		**Verifier**
		$x \xleftarrow{\$} \mathbb{Z}_p^*$
	$\xleftarrow{\quad x \quad}$	
$P' := P \cdot u^{x \cdot c}$		$P' := P \cdot u^{x \cdot c}$
	$\xleftrightarrow{\quad BP(\mathbf{g}, \mathbf{h}, u^x, P'; \mathbf{a}, \mathbf{b}) \quad}$	

6.4 "Bulletproofs" Protocol

The "Bulletproofs" protocol inner product argument to prove relation R_6 that is being used in protocol 1 to prove relation R_5 is as follows.
We first give a formal written description of the protocol and then a diagram to see visually all the exchanges in the protocol.
We assume without loss of generality that n is a power of 2 (if it is not the case we can simply pad the input with zero values for example).

Protocol

Bulletproofs Inner product Argument (protocol 2) :

- **Input :** $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)$,
  - $\mathcal{P}$'s input : $(\mathbf{g}, \mathbf{h}, u, P; \mathbf{a}, \mathbf{b})$,
  - $\mathcal{V}$'s input : $(\mathbf{g}, \mathbf{h}, u, P)$.
- **Output :** $\mathcal{V}$ accepts or rejects the proof.

- **step 1 :** if $n = 1$ go to **step 2**, else go to **step 4**:
- **step 2 :** $\mathcal{P}$ sends to $\mathcal{V}$: $a, b \in \mathbb{Z}_p$.
- **step 3 :** $\mathcal{V}$ computes $c = a \cdot b \in \mathbb{Z}_p$ and checks if $P = g^a h^b u^c \in \mathbb{G}$.
  If yes, $\mathcal{V}$ accepts the argument; otherwise $\mathcal{V}$ rejects it.
- **step 4 :** $\mathcal{P}$ computes:
  - $n' = \frac{n}{2}$
  - $c_L =< \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} >\in \mathbb{Z}_p$
  - $c_R =< \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} >\in \mathbb{Z}_p$
  - $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L} \in \mathbb{G}$
  - $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R} \in \mathbb{G}$
- **step 5 :** $\mathcal{P}$ sends to $\mathcal{V}$: L,R
- **step 6 :** $\mathcal{V}$ picks uniformly at random $x \in \mathbb{Z}_p^*$
- **step 7 :** $\mathcal{V}$ sends to $\mathcal{P}$: x
- **step 8 :** $\mathcal{P}$ and $\mathcal{V}$ compute:
  - $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$
  - $\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$
  - $P' = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$
- **step 9 :** $\mathcal{P}$ computes:
  - $\mathbf{a}' = \mathbf{a}_{[:n']} \cdot x + \mathbf{a}_{[n':]} \cdot x^{-1} \in \mathbb{Z}_p^{n'}$
  - $\mathbf{b}' = \mathbf{b}_{[:n']} \cdot x^{-1} + \mathbf{b}_{[n':]} \cdot x \in \mathbb{Z}_p^{n'}$
- **step 10 :** recursively run this protocol on input $(\mathbf{g}', \mathbf{h}', u, P'; \mathbf{a}', \mathbf{b}')$.

As we can observe the "Bulletproofs" Inner Product Argument is a recursive protocol. At each recursion step it reduces the difficulty of the argument to a simpler argument by reducing the dimension n by a factor 2 to $n' = n/2$. The recursion ends up when the dimension n is equal to 1. In this case, the argument is easy and is of constant size by simply asking the prover $\mathcal{P}$ to send in clear $a, b \in \mathbb{Z}_p$ (they are not written with bold letters since they are simply scalars when $n = 1$) to the verifier $\mathcal{V}$ and asking the verifier to check whether $P = g^a h^b u^c$.

Notice that since a, b are sent in clear the argument protocol cannot have the zero-knowledge property.

Diagram

Bulletproofs Protocol

Prover		**Verifier**
$n' := n/2$		$n' := n/2$
$c_L :=< \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} >$		
$c_R :=< \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} >$		
$L := \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L}$		
$R := \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R}$		
	$L, R \longrightarrow$	
		$x \xleftarrow{\$} \mathbb{Z}_p$
	$\longleftarrow x$	
$\mathbf{g}' := \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}$		$\mathbf{g}' := \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x}$
$\mathbf{h}' := \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}}$		$\mathbf{h}' := \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}}$
$P' := L^{x^2} P R^{x^{-2}}$		$P' := L^{x^2} P R^{x^{-2}}$
$\mathbf{a}' = \mathbf{a}_{[:n']} \cdot x + \mathbf{a}_{[n':]} \cdot x^{-1}$		
$\mathbf{b}' = \mathbf{b}_{[:n']} \cdot x^{-1} + \mathbf{b}_{[n':]} \cdot x$		
$\vdots$		$\vdots$
$(n' = 1)$		$(n' = 1)$
	$\mathbf{a}', \mathbf{b}' \longrightarrow$	
		$\mathbf{g}'^{\mathbf{a}'} \mathbf{h}'^{\mathbf{b}'} u^{c} \stackrel{?}{=} P'$

6.4.1 Theorem 3

The "Bulletproofs" inner-product argument (protocol 1) presented in section 6C for relation R_5 has perfect completeness and computational witness extended-emulation for either extracting a nontrivial discrete logarithm relation between $\mathbf{g}$ *and* $\mathbf{h}$ *or extracting a valid witness* $\mathbf{a}, \mathbf{b}$.

We refer the interested reader to the original "Bulletproofs" paper for the proof.
Notice that the "Bulletproofs" inner-product argument extractor constructed in the paper in order to prove the *Computational Witness Extended Emulation* property, which we will denote χ^{BP}, requires a polynomial number of $N^{BP}(n) = 4^{log_2(n)} = n^2$ transcripts to successfully extract the valid witness.
χ^{BP} will be used later on as a subroutine to construct the extractors to prove the *Computational Witness Extended Emulation* property for protocols that use as building block the "Bulletproofs" inner product argument.

6.4.2 Complexity

The "Bulletproofs" inner-product argument has the following complexities :

- **# Communications :** $2log_2(n) + 2$ elements:
  - L and R at each of the $log_2(n)$ rounds : $2log_2(n)$ elements in $\mathbb{G}$
  - a and b at the last round : 2 elements in $\mathbb{Z}_p$

- **# Bases :** $2n + 1$ bases :
  - $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$: $2n$ bases
  - $u \in \mathbb{G}$: 1 base
- **# Exponentiations in $\mathbb{G}$:**
  - **Prover :** $8n + 4log_2(n) - 8$ exponentiations in $\mathbb{G}$
  - **Verifier :** $4n + 2log_2(n) - 1$ exponentiations in $\mathbb{G}$
- **# Exponentiations in $\mathbb{Z}_p$**
  - **Prover:** $3log_2(n) + 2$ exponentiations in $\mathbb{Z}_p$
  - **Verifier:** $3log_2(n) + 2$ exponentiations in $\mathbb{Z}_p$

  Table 6.2 shows the complexity to compute each element for one round of the protocol otherwise specified by *only once* which means that the element is computed only once. At the first round, $n' = \frac{n}{2}$ and n' is successively divided by 2 at each following round. Using the geometric formula $\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1}$, we obtain $\frac{n}{2} + \frac{n}{4} + \ldots + 2 + 1 = \sum_{i=0}^{log_2(n)-1} 2^i = \frac{2^{log_2(n)-1+1}-1}{2-1} = n - 1$. This formula allows us to compute the total number of exponentiations for all the rounds.
- **# Multiplications in $\mathbb{G}$:**
  - **Prover :** $4n + 2log_2(n) - 4$ multiplications:
    - P' : $2log_2(n)$ multiplications
    - $2(n-1)$ multiplications for each of $\mathbf{g}', \mathbf{h}'$: $4(n-1)$ multiplications
  - **Verifier :** $2n$ multiplications:
    - $n - 1$ multiplications for each of $\mathbf{g}', \mathbf{h}'$: $2n - 2$ multiplications
    - last verification equation : 2 multiplications
- **# Multiplications in $\mathbb{Z}_p$:**
  - **Prover :** $6n - 6$ multiplications:
    - $n - 1$ multiplications for each of c_L, c_R : $2n - 2$ multiplications
    - $2(n-1)$ multiplications for each of $\mathbf{a}', \mathbf{b}'$: $4n - 4$ multiplications
  - **Verifier :** 1 multiplication in the last step's verification.

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
L	$2n' + 1$			
R	$2n' + 1$			
x^{-1}		1		1
x^2		1		1
x^{-2}		1		1
g'	2n'		2n'	
h'	2n'		2n'	
P'	2		2	
Verif Eq			2 (only once)	
u^c				1 (only once)
u'				1 (only once)
u^x		2 (only once)		
TOTAL	$8n + 4log_2(n) - 8$	$3log_2(n) + 2$	$4n + 2log_2(n) - 1$	$3log_2(n) + 2$

Table 6.2: Complexity of Bulletproofs Protocol

Part III

Protocols

Chapter 7

0-1 Proof System

7.1 Key Ideas

In this subsection, we briefly describe our key ideas when we designed the following protocols to prove the relation $R_1 \equiv \{(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m; (v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}_p^m) : V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\}, \forall i \in [1,l], j \in [1,m]\}$.

All the ideas are greatly inspired and come in large part directly from the *"Bulletproofs"* paper.

In order to prove the relation R_1 presented above, we prove the knowledge of $\mathbf{a_L^{(j)}} = (\mathbf{a_L^{(j)}}[1], \dots, \mathbf{a_L^{(j)}}[m]) \in \mathbb{Z}_p^m$ satisfying the three following equations $\forall j \in [1,l]$:

$$\mathbf{a_L^{(j)}}[i] = v_j^{(i)}, i \in \{1, \dots, m\}, \mathbf{a_L^{(j)}} \circ \mathbf{a_R^{(j)}} = \mathbf{0}^m, \mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m.$$

These three equations are equivalent to $v_j^{(i)} \in \{0,1\}\ \forall (i,j) \in \{1, \dots, m\} \times \{1, \dots, l\}$. [1]

The protocol should not reveal the values $\{\mathbf{a_L^{(j)}}\}_{j=1}^m$ nor the values $\{\mathbf{a_R^{(j)}}\}_{j=1}^m$ because it would reveal the value of the votes and would break the confidentiality for the voters. The prover $\mathcal{P}$ will thus send to the verifier $\mathcal{V}$ a perfectly hiding commitment $A = h^\alpha(\mathbf{g_1^{a_L^{(1)}}} \dots \mathbf{g_l^{a_L^{(l)}}})(\mathbf{h_1^{a_R^{(1)}}} \dots \mathbf{h_l^{a_R^{(l)}}})$ over the values $\{\mathbf{a_L^{(j)}}\}_{j=1}^m$ and $\{\mathbf{a_R^{(j)}}\}_{j=1}^m\}$ and we will prove in a zero-knowledge fashion using commitment A that these equations are satisfied.

However, in order to use the "Bulletproofs" inner-product argument to reduce the size of the proof, we will reduce the above system of equations to a single dot product equation for each $j \in [1,l]$.

Using the *Schwartz-Zippel Lemma* presented in section 3.2.1, the above equations hold except with negligible probability if the following equations hold for all $j \in [1,l]$ for y picked uniformly at random in $\mathbb{Z}_p$:

$$\mathbf{a_L^{(j)}}[i] = v_j^{(i)}, i \in \{1, \dots, m\}, \sum_{k=1}^{m} \mathbf{a_L^{(j)}}[k] \cdot \mathbf{a_R^{(j)}}[k] y^{k-1} = 0, \sum_{k=1}^{m} (\mathbf{a_R^{(j)}}[k] \mathbf{a_L^{(j)}}[k] - 1) y^{k-1} = 0,$$

We can rewrite equivalently these equations using dot products as follows $\forall j \in [1,l]$:

$$\mathbf{a_L^{(j)}}[i] = v_j^{(i)}, i \in \{1, \dots, m\}, < \mathbf{a_L^{(j)}}, \mathbf{a_R^{(j)}} \circ \mathbf{y}^m >= 0, < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m >= 0,$$

Using the *Schwartz-Zippel Lemma* a second time, we prove that these equations hold except with negligible probability by showing the following equation holds for z picked uniformly at random $\forall j \in [1,l]$:

$$\sum_{k=1}^{m} (\mathbf{a_L^{(j)}}[k] - v_j^{(k)}) z^{k+1} + z < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m > + < \mathbf{a_L^{(j)}}, \mathbf{a_R^{(j)}} \circ \mathbf{y}^m >= 0$$

which can be expressed equivalently in one single dot product equation :

$$< \mathbf{a_L^{(j)}} - z\mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >= \sum_{k=1}^{m} z^{1+k} v_j^{(k)} + \delta(y,z)$$

, where $\delta(y,z) := (z - z^2) < \mathbf{1}^m, \mathbf{y}^m > - \sum_{k=1}^{m} z^{k+2}$ as proved in appendix A.

[1] Indeed, substituting $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$ in $\mathbf{a_L^{(j)}} \circ \mathbf{a_R^{(j)}} = \mathbf{0}^m$ leads to $\mathbf{a_L^{(j)}} \circ (\mathbf{a_L^{(j)}} - \mathbf{1}^m) = \mathbf{0}^m$ or $\mathbf{a_L^{(j)}}[i](\mathbf{a_L^{(j)}}[i] - 1) = 0\ \forall (i,j) \in \{1, \dots, m\} \times \{1, \dots, l\}$. Since $\mathbf{a_L^{(j)}} = v_j^{(i)}$, we have $v_j^{(i)}(v_j^{(i)} - 1) = 0\ \forall (i,j) \in \{1, \dots, m\} \times \{1, \dots, l\}$ which is equivalent to $v_j^{(i)} \in \{0,1\}$ $\forall (i,j) \in \{1, \dots, m\} \times \{1, \dots, l\}$ since in $\mathbb{Z}_p$ the polynomial $X(X-1) \in \mathbb{Z}_p[X]$ has only two roots : $X = 0$ and $X = 1$.

We have m equations of the form $< \mathbf{A_j}, \mathbf{B_j} >= C_j$.
Assume that the verifier $\mathcal{V}$ can compute the right-hand side term of the equation, C_j. Then, if the prover $\mathcal{P}$ sends to the verifier the two vectors representing the left-hand side and the right-hand side of the dot product ($\mathbf{A_j}$ and $\mathbf{B_j}$ respectively), the verifier could just verify the equality between the dot product and the right-hand side of the equation: $< \mathbf{A_j}, \mathbf{B_j} >= C_j$.
However, sending in clear the terms A_j and B_j breaks the zero-knowledge argument because it reveals information about the witness.
To remedy this problem, we will proceed as in the Schnorr's proof, that is we will blind these two terms using the two following blinding terms which are denoted $\mathbf{s_L^{(j)}} \in \mathbb{Z}_p^m$ and $\mathbf{s_R^{(j)}} \in \mathbb{Z}_p^m$ and a challenge $x \in \mathbb{Z}_p$, also like in the Schnorr's Protocol, which allows to send them in clear while preserving the zero-knowledge property.
The resulting blinded term corresponding to the left-hand side of the dot-product in the j-th dot product equation will be denoted $\mathbf{l^{(j)}}$ while the right-hand side one will be denoted $\mathbf{r^{(j)}}$.

We will then combine all these commitments to create a commitment over the left-hand side and the right-hand side of the dot product. The verifier then checks that this commitment denoted P is valid.

At the end of the protocol, the verifier needs to check, in **step 12**, the following equations:

$$\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >, j \in \{1, \dots, l\}$$

Picking ϕ uniformly at random in $\mathbb{Z}_p$ and using the Schwartz-Zippel lemma once again this is equivalent except with negligible probability to the following verification equation:
$\sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >) = 0$, or just written differently : $< \tilde{\mathbf{l}}, \tilde{\mathbf{r}} >= \tilde{t}$, where :
$\tilde{l} := (\phi^0 l^{(1)}, \dots, \phi^{l-1} l^{(l)})$, $\tilde{r} := (r^{(1)}, \dots, r^{(l)})$ and $\tilde{t} := \sum_{j=1}^{l} \phi^{j-1} \hat{t}^{(j)}$.

Finally, the verification of these dot product equations will suggest the usage of the "Bulletproofs" protocol to verify inner product relations to reduce the amount of data exchange.

Recall that for relation R_1, the C.R.S. is $\sigma := (g_1, \dots, g_l, h)$, the statement is $u := \mathbf{V} = (V_1, \dots, V_m)$ and the witness is $w := (v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m)$.

7.2 Proof Protocol first try

We first give a formal written description of the protocol and then a diagram to see visually all the exchanges in the protocol.

0-1 Protocol (first try)

- Input : $(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{h_1}, \dots, \mathbf{h_l}, \mathbf{V} = (V_j)_{j=1}^m \in \mathbb{G}^m; \{v_1^{(j)}, \dots, v_l^{(j)}, \gamma_j\}_{j=1}^m \subseteq \mathbb{Z}_p^{l+1})$,

- $\mathcal{P}$'s input : $(g_1, \dots, g_l, h, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{h_1}, \dots, \mathbf{h_l}, \mathbf{V}, \{v_1^{(j)}, \dots, v_l^{(j)}, \gamma_j\}_{j=1}^m)$,
- $\mathcal{V}$'s input : $(g_1, \dots, g_l, h, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{h_1}, \dots, \mathbf{h_l}, V)$

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 1 : $\mathcal{P}$ computes :

- $\mathbf{a_L^{(j)}}[i] = v_j^{(i)}$, $i \in \{1, \dots, m\}$, $\mathbf{a_L^{(j)}} = (\mathbf{a_L^{(j)}}[1], \dots, \mathbf{a_L^{(j)}}[m])$, $j \in \{1, \dots, l\}$,
- $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$, $j \in \{1, \dots, l\}$,
- $\mathcal{P}$ picks uniformly at random α, ρ in $\mathbb{Z}_p$ and $\mathbf{s_L^{(j)}}, \mathbf{s_R^{(j)}}$ in $\mathbb{Z}_p^m$, $j \in \{1, \dots, l\}$,
- $A = h^{\alpha}(\mathbf{g_1}^{\mathbf{a_L^{(1)}}} \dots \mathbf{g_l}^{\mathbf{a_L^{(l)}}})(\mathbf{h_1}^{\mathbf{a_R^{(1)}}} \dots \mathbf{h_l}^{\mathbf{a_R^{(l)}}}) \in \mathbb{G}$,

- $S = h^{\rho}(\mathbf{g_1}^{\mathbf{s_L^{(1)}}} \dots \mathbf{g_l}^{\mathbf{s_L^{(l)}}})(\mathbf{h_1}^{\mathbf{s_R^{(1)}}} \dots \mathbf{h_l}^{\mathbf{s_R^{(l)}}}) \in \mathbb{G}$.

- step 2 : $\mathcal{P}$ sends to $\mathcal{V}$: A, S.
- step 3 : $\mathcal{V}$ picks uniformly at random y, z in $\mathbb{Z}_p^*$.
- step 4 : $\mathcal{V}$ sends y, z to $\mathcal{P}$.

For the following part of the proof, given $j \in \{1, \dots, l\}$, we define $\mathbf{l^{(j)}}(X), \mathbf{r^{(j)}}(X) \in \mathbb{Z}_p^m[X]$ and $t^{(j)}(X) \in \mathbb{Z}_p[X]$ as follows :

$$\begin{aligned}
&\mathbf{l^{(j)}}(X) := \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} X \in \mathbb{Z}_p^m[X], \\
&\mathbf{r^{(j)}}(X) := \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} X) + z^2 \cdot \mathbf{z}^m \in \mathbb{Z}_p^m[X], \\
&t^{(j)}(X) :=< \mathbf{l^{(j)}}(X), \mathbf{r^{(j)}}(X) >= t_0^{(j)} + t_1^{(j)} X + t_2^{(j)} X^2 \in \mathbb{Z}_p[X].
\end{aligned}$$

- step 5 : $\mathcal{P}$ computes :

- $\mathcal{P}$ picks uniformly at random $\tau_1, \tau_2 \in \mathbb{Z}_p$,
- $t_1^{(j)} =< \mathbf{s_L^{(j)}}, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m > + < \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{s_R^{(j)}} >$, $j \in \{1, \dots, l\}$,
- $t_2^{(j)} =< \mathbf{s_L^{(j)}}, \mathbf{y}^m \circ \mathbf{s_R^{(j)}} >$, $j \in \{1, \dots, l\}$,
- $T_1 = h^{\tau_1}(g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}})$, $T_2 = h^{\tau_2}(g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}})$.

- step 6 : $\mathcal{P}$ sends to $\mathcal{V}$: T_1, T_2.
- step 7 : $\mathcal{V}$ picks uniformly at random x in $\mathbb{Z}_p^*$.
- step 8 : $\mathcal{V}$ sends x to $\mathcal{P}$.
- step 9 : $\mathcal{P}$ computes for all $j \in \{1, \dots, l\}$:

- $\mathbf{l^{(j)}} = \mathbf{l^{(j)}}(x) = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} x \in \mathbb{Z}_p^m$,
- $\mathbf{r^{(j)}} = \mathbf{r^{(j)}}(x) = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m \in \mathbb{Z}_p^m$,
- $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >\in \mathbb{Z}_p$,
- $\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{j=1}^m z^{1+j} \gamma_j$,
- $\mu = \alpha + \rho x \in \mathbb{Z}_p$.

- step 10 : $\mathcal{P}$ sends to $\mathcal{V}$: $\tau_x, \mu, \{\hat{t}^{(j)}, \mathbf{l^{(j)}}, \mathbf{r^{(j)}}\}_{j=1}^l$.
- step 11 : $\mathcal{V}$ computes:

- $\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$, $i \in \{1, \dots, m\}$, $\mathbf{h'_j} = (\mathbf{h'_j}[1], \dots, \mathbf{h'_j}[l])$, $j \in [1, l]$.
- $P = AS^x(\mathbf{g}_1 \dots \mathbf{g}_l)^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \in \mathbb{G}$.

- step 12 : $\mathcal{V}$ checks the following identities :

- $(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}}) h^{\tau_x} \stackrel{?}{=} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$.
- $P \stackrel{?}{=} h^{\mu}(\mathbf{g_1}^{\mathbf{l^{(1)}}} \dots \mathbf{g_l}^{\mathbf{l^{(l)}}})((\mathbf{h'_1})^{\mathbf{r^{(1)}}} \dots (\mathbf{h'_l})^{\mathbf{r^{(l)}}})$.

- $\hat{t}^{(j)} \stackrel{?}{=} < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$, $j \in \{1, \dots, l\}$.

However, in order to reduce the amount of data exchanged in this zero-knowledge argument protocol, instead of sending $\{\hat{t}^{(j)}, \mathbf{l^{(j)}}, \mathbf{r^{(j)}}\}_{j=1}^{l}$ at **step 10 :**, we only send $\{\hat{t}^{(j)}\}_{j=1}^{l}$ and we will change **step 10**, **step 11** and **step 12** as described below.
Notice that it is important that $\mathcal{P}$ first sends $\{\hat{t}^{(j)}\}_{j=1}^{l}$ before he receives the challenge ϕ from $\mathcal{V}$. Otherwise he could cheat by adapting $\{\hat{t}^{(j)}\}_{j=1}^{l}$ to the value ϕ.
The new **step 10**, **step 11** and **step 12** are as follows :

- step 10 : $\mathcal{P}$ sends to $\mathcal{V}$: $\tau_x, \mu, \{\hat{t}^{(j)}\}_{j=1}^{l}$.
- step 11 : $\mathcal{V}$ computes:

- $\mathcal{V}$ picks ϕ uniformly at random in $\mathbb{Z}_p$
- $\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$, $i \in \{1, \dots, m\}$, $\mathbf{h'_j} = (\mathbf{h'_j}[1], \dots, \mathbf{h'_j}[m])$, $j \in [1, l]$.
- $\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$, $i \in \{1, \dots, m\}$, $\mathbf{g'_j} = (\mathbf{g'_j}[1], \dots, \mathbf{g'_j}[m])$, $j \in [1, l]$.
- $P = AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$.

- step 12 : $\mathcal{V}$ checks the following identity :

- $(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}})h^{\tau_x} \stackrel{?}{=} \mathbf{V}^{\mathbf{z}^m \cdot z^2}(g_1 \dots g_l)^{\delta(y,z)}T_1^x T_2^{x^2}$.

At this point, to conclude the proof, $\mathcal{V}$ only needs to verify :

- $P \stackrel{?}{=} h^{\mu}(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{l^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{l^{(l)}}})((\mathbf{h'_1})^{\mathbf{r^{(1)}}} \dots (\mathbf{h'_l})^{\mathbf{r^{(l)}}})$.
- $\sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1} \stackrel{?}{=} < (\mathbf{l^{(j)}}\phi^{j-1})_{j=1}^{l}, (\mathbf{r^{(j)}})_{j=1}^{l} >$, $j \in \{1, \dots, l\}$.

Which is equivalent to proving the following identity :
$\Big[Ph^{-\mu} = (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{l^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{l^{(l)}}})((\mathbf{h'_1})^{\mathbf{r^{(1)}}} \dots (\mathbf{h'_l})^{\mathbf{r^{(l)}}}) \wedge \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1} = < (\mathbf{l^{(j)}}\phi^{j-1})_{j=1}^{l}, (\mathbf{r^{(j)}})_{j=1}^{l} >\Big]$,
which suggests the usage of the "Bulletproofs" inner-product argument.
- step 13 : $\mathcal{V}$ sends ϕ to P.
- step 14 : $\mathcal{P}$ and $\mathcal{V}$ engage in the "Bulletproofs" inner-product argument for the following relation :

$$R_5 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g^a h^b} \wedge c = < \mathbf{a}, \mathbf{b} >\}$$

, on input :

- $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $\mathbf{h} := (\mathbf{h'_1} || \dots || \mathbf{h'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $P := Ph^{-\mu} \in \mathbb{G}$,
- $c := \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1} \in \mathbb{Z}_p$,
- $\mathbf{a} := (\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1}) \in \mathbb{Z}_p^{(m \cdot l)}$,
- $\mathbf{b} := (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) \in \mathbb{Z}_p^{(m \cdot l)}$,

for $n = (m \cdot l)$.

We use the "Bulletproofs" inner product argument in order to reduce the number of communications. Indeed, recall the "Bulletproofs" inner product argument has a logarithmic communication complexity.
However, the "Bulletproofs" requires to have an input that is a power of 2 but it is not a problem if $m \cdot l$ is not a

power of 2 we can simply pad the input so to make it a power of 2.

It is worth noting that even though the "Bulletproofs" inner product argument leak information about $\mathbf{a} := (\mathbf{l^{(1)}}\phi^0||\dots||\mathbf{l^{(l)}}\phi^{l-1})$ and $\mathbf{b} := (\mathbf{r^{(1)}}||\dots||\mathbf{r^{(l)}})$ this is not a problem for the zero-knowledge property because $\mathbf{l}$ and $\mathbf{r}$ are blinded and will not give any information about the witness $(v_1^{(j)},\dots,v_l^{(j)})_{j=1}^m$.

0-1 Protocol (first part)

Prover		**Verifier**
$\mathbf{a_L^{(j)}} = (v_j^{(1)},\dots,v_j^{(m)})$		
$\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$		
$\mathbf{s_L^{(j)}}, \mathbf{s_R^{(j)}} \xleftarrow{\$} \mathbb{Z}_p^m$		
$A = h^\alpha(\mathbf{g_1^{a_L^{(1)}}} \dots \mathbf{g_l^{a_L^{(l)}}})(\mathbf{h_1^{a_R^{(1)}}} \dots \mathbf{h_l^{a_R^{(l)}}})$		
$S = h^\alpha(\mathbf{g_1^{s_L^{(1)}}} \dots \mathbf{g_l^{s_L^{(l)}}})(\mathbf{h_1^{s_R^{(1)}}} \dots \mathbf{h_l^{s_R^{(l)}}})$		
	$\xrightarrow{A,S}$	
		$y, z \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{y,z}$	
$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p^*$		
$T_1 = h^{\tau_1} g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}}$		
$T_2 = h^{\tau_2} g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}}$		
	$\xrightarrow{T_1,T_2}$	
		$x \xleftarrow{\$} \mathbb{Z}_p^*$
	$\xleftarrow{x}$	
$\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} x$		
$\mathbf{r^{(j)}} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m$		
$\hat{t}^{(j)} = < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$		
$\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{j=1}^{m} z^{1+j}\gamma_j$		
$\mu = \alpha + \rho x$		
	$\xrightarrow{\tau_x, \mu, (\hat{t}^{(i)})_{i=1}^l}$	
		$(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}}) h^{\tau_x} \stackrel{?}{=} \mathbf{V}^{\mathbf{z}^m \cdot z^2} \cdot (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$

0-1 Protocol (part 2)

Prover		**Verifier**
		$\phi \xleftarrow{\$} \mathbb{Z}_p^*$
	$\xleftarrow{\phi}$	
$\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$		$\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$
$\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$		$\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$
$P = AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} \cdot$		$P = AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} \cdot$
$(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$		$(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$
$\mathbf{g} := (\mathbf{g'_1} \| \dots \| \mathbf{g'_l})$		$\mathbf{g} := (\mathbf{g'_1} \| \dots \| \mathbf{g'_l})$
$\mathbf{h} := (\mathbf{h'_1} \| \dots \| \mathbf{h'_l})$		$\mathbf{h} := (\mathbf{h'_1} \| \dots \| \mathbf{h'_l})$
$c := \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$		$c := \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$
$\mathbf{a} := (\mathbf{l^{(1)}} \phi^0 \| \dots \| \mathbf{l^{(l)}} \phi^{l-1})$		
$\mathbf{b} := (\mathbf{r^{(1)}} \| \dots \| \mathbf{r^{(l)}})$		
$BULLETSPROOFS(\mathbf{g}, \mathbf{h}, P, c; \mathbf{a}, \mathbf{b})$		
$\longleftrightarrow$		

7.3 Complexity

The 0-1 zero-knowledge argument protocol presented above has the following complexities :

- **# Communications :** $l + 2log_2(ml) + l + 8$ elements of which
  - $2log_2(ml) + 4$ **Group elements :**
    - A, S, T_1, T_2 : 4 elements in $\mathbb{G}$
    - "Bulletproofs" inner-product arguments with $n = ml$: $2log_2(ml)$ elements in $\mathbb{G}$
  - $l + 4$ $\mathbb{Z}_p$ **elements :**
    - $\hat{t}$: l elements in $\mathbb{Z}_p$
    - τ_x, μ : 2 elements in $\mathbb{Z}_p$
    - "Bulletproofs" inner-product arguments with $n = ml$: 2 elements in $\mathbb{Z}_p$
- **# Bases :** $2ml + 2l + 2$ bases :
  - $\mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{h_1}, \dots, \mathbf{h_l} \in \mathbb{G}^m$: $2ml$ bases
  - $g_1, \dots, g_l, h_1, \dots, h_l \in \mathbb{G}$: $2l$ bases
  - $h, u \in \mathbb{G}$: 2 bases
- **# Exponentiations in $\mathbb{G}$:**
  - **Prover :** $16lm + 2l + m + 4log_2(lm) - 2$ exponentiations in $\mathbb{G}$
  - **Verifier :** $8lm + 2log_2(lm) + 1$ exponentiations in $\mathbb{G}$

  The details are shown in Table 7.1.
- **# Exponentiations in $\mathbb{Z}_p$:**
  - **Prover :** $3log_2(lm) + 7$ exponentiations in $\mathbb{Z}_p$
  - **Verifier :** $3log_2(lm) + 7$ exponentiations in $\mathbb{Z}_p$

The details are shown in Table 7.1.

- **# Multiplications in $\mathbb{G}$:** $10ml + 2log_2(ml) + m + 6l + 2 : 2ml + 2l - 2$
  - **Prover:** $10ml + 2l + 2log_2(ml) - 3$ multiplications :
    - $2ml$ for each of A,S: $4ml$ multiplications
    - l for each of T_1, T_2 : $2l$ multiplications
    - P : $2ml + 1$ multiplications
    - "Bulletproofs" inner-product argument with $n = ml$: $4ml + 2log_2(ml) - 4$ multiplications
  - **Verifier:** $4ml + 1$ multiplications :
    - P : $2ml + 1$ multiplications
    - "Bulletproofs" inner-product argument with $n = ml$: $2ml$ multiplications
- **# Multiplications in $\mathbb{Z}_p$:**
  - **Prover:** $6ml - 7l + m$ multiplications:
    - l : ml multiplications
    - r : $5ml - 7l + m - 1$ multiplications
    - μ : 1 multiplication
  - **Verifier:** 1 multiplication:
    - "Bulletproofs" inner-product argument with $n = ml$: 1 multiplication

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
A	2lm+1			
S	2lm+1			
$\{z_j\}_{j=1}^{m+2}$		1		1
T_1	l+1			
T_2	l+1			
x^2		1		1
y^{-1}		1		1
ϕ		1		1
δ		1		1
P	2lm+2		2lm+2	
τ_x		l		
h'	lm		lm	
g'	lm		lm	
BP(lm)	$8lm + 4log_2(lm) - 8$	$3log_2(lm) + 2$	$4lm + 2log_2(lm) - 1$	$3log_2(lm) + 2$
TOTAL	$16lm+2l+4log_2(lm)-2$	$3log_2(lm) + 7$	$8lm + 2log_2(lm) + 1$	$3log_2(lm) + 7$

Table 7.1: Complexity of 0-1 Protocol

As desired the number of exponentiation and multiplications is linear in lm but unfortunately, the size of the elements exchanged is proportional to l which does not allow efficient Risk Limiting Audits.

7.4 Theorem 4

Under the Discrete Log Relation assumption, the 0-1 protocol presented in section 7C for relation R_1 has perfect completeness, perfect special honest-verifier zero-knowledge and computational witness extended emulation.

In the followings subsections, we will show point by point that the 0-1 protocol described above satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* for relation R_1.

7.4.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $\{\gamma_j\}_{j=1}^m, \{v_i^{(j)}\}_{i=1,j=1}^{l,m}$ for the relation $R_1 \equiv \{(g_1,\dots,g_l,h \in \mathbb{G}, \mathbf{V}=(V_1,\dots,V_m) \in \mathbb{G}^m;$
$(v_1^{(1)},\dots,v_1^{(m)}),\dots,(v_l^{(1)},\dots,v_l^{(m)}),\gamma=(\gamma_1,\dots,\gamma_m) \in \mathbb{Z}_p^m) : V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\},$
$\forall i \in [1,l], j \in [1,m]\}$.
We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma,u,w), \mathcal{V}(\sigma,u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u,w) \leftarrow \mathcal{A}(\sigma), (\sigma,u,w) \in R_1\right] = 1$.

To see this, let's look at the verification equations that the verifier $\mathcal{V}$ checks in the protocol :

$$\begin{aligned}
\mathbf{V}^{\mathbf{z}^m \cdot z^2}(g_1\dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} &= (h^{\gamma_1} g_1^{v_1^{(1)}} \dots g_l^{v_l^{(1)}}, \dots, h^{\gamma_m} g_1^{v_1^{(m)}} \dots g_l^{v_l^{(m)}})^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} \\
&\quad (h^{\tau_1}(g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}}))^x (h^{\tau_2}(g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}}))^{x^2} \\
&= (h^{\gamma_1 z^2} g_1^{v_1^{(1)} z^2} \dots g_l^{v_l^{(1)} z^2} \dots h^{\gamma_m z^{m+2}} g_1^{v_1^{(m)} z^{m+2}} \dots g_l^{v_l^{(m)} z^{m+2}}) g_1^{\delta(y,z)} \dots g_l^{\delta(y,z)} \\
&\quad h^{\tau_1 x} g_1^{t_1^{(1)} x} \dots g_l^{t_1^{(l)} x} h^{\tau_2 x^2} g_1^{t_2^{(1)} x^2} \dots g_l^{t_2^{(l)} x^2} \\
&= g_1^{\sum_{k=1}^m v_1^{(k)} z^{k+1} + \delta(y,z) + t_1^{(1)} x + t_2^{(1)} x^2} \dots g_l^{\sum_{k=1}^m v_l^{(k)} z^{k+1} + \delta(y,z) + t_1^{(l)} x + t_2^{(l)} x^2} h^{\tau_1 x + \tau_2 x^2 + \sum_{k=1}^m \gamma_k z^{k+1}} \\
&= g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}} h^{\tau_x}
\end{aligned}$$

since $\mathcal{P}$ has followed the protocol and thus:

$\hat{t}^{(j)} = \sum_{k=1}^m v_j^{(k)} z^{k+1} + \delta(y,z) + t_1^{(j)} x + t_2^{(j)} x^2$, $j \in \{1,\dots,l\}$ and $\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{k=1}^m \gamma_k z^{k+1}$.

We also have :

$$\begin{aligned}
P &= AS^x (\mathbf{g_1'}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g_l'}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h_1'} \dots \mathbf{h_l'})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= \left(h^\alpha (\mathbf{g_1}^{\mathbf{a_L}^{(1)}} \dots \mathbf{g_l}^{\mathbf{a_L}^{(l)}})(\mathbf{h_1}^{\mathbf{a_R}^{(1)}} \dots \mathbf{h_l}^{\mathbf{a_R}^{(l)}})\right) \left(h^\rho (\mathbf{g_1}^{\mathbf{s_L}^{(1)}} \dots \mathbf{g_l}^{\mathbf{s_L}^{(l)}})(\mathbf{h_1}^{\mathbf{s_R}^{(1)}} \dots \mathbf{h_l}^{\mathbf{s_R}^{(l)}})\right)^x \mathbf{g_1'}^{-z\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g_l'}^{-z\phi^{l-1} \cdot \mathbf{1}^m} \\
&\quad (\mathbf{h_1'} \dots \mathbf{h_l'})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= \left(h^\alpha (\mathbf{g_1'}^{\phi^0 \mathbf{a_L}^{(1)}} \dots \mathbf{g_l'}^{\phi^{l-1} \mathbf{a_L}^{(l)}})(\mathbf{h_1'}^{\mathbf{y}^m \circ \mathbf{a_R}^{(1)}} \dots \mathbf{h_l'}^{\mathbf{y}^m \circ \mathbf{a_R}^{(l)}})\right) \left(h^\rho (\mathbf{g_1'}^{\phi^0 \mathbf{s_L}^{(1)}} \dots \mathbf{g_l'}^{\phi^{l-1} \mathbf{s_L}^{(l)}})(\mathbf{h_1'}^{\mathbf{y}^m \circ \mathbf{s_R}^{(1)}} \dots \mathbf{h_l'}^{\mathbf{y}^m \circ \mathbf{s_R}^{(l)}})\right)^x \\
&\quad \mathbf{g_1'}^{-z\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g_l'}^{-z\phi^{l-1} \cdot \mathbf{1}^m} (\mathbf{h_1'} \dots \mathbf{h_l'})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= h^{(\alpha + \rho x)} \mathbf{g_1'}^{\phi^0 (\mathbf{a_L}^{(1)} - z \cdot \mathbf{1}^m + \mathbf{s_L}^{(1)} x)} \dots \mathbf{g_l'}^{\phi^{l-1} (\mathbf{a_L}^{(l)} - z \cdot \mathbf{1}^m + \mathbf{s_L}^{(l)} x)} \mathbf{h_1'}^{\mathbf{y}^m \circ (\mathbf{a_R}^{(1)} + \mathbf{s_R}^{(1)} x + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m} \dots \\
&\quad \mathbf{h_l'}^{\mathbf{y}^m \circ (\mathbf{a_R}^{(l)} + \mathbf{s_R}^{(l)} x + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m} \\
&= h^\mu (\mathbf{g_1'}^{\phi^0 \cdot \mathbf{l}^{(1)}} \dots \mathbf{g_l'}^{\phi^{l-1} \cdot \mathbf{l}^{(l)}})((\mathbf{h_1'})^{\mathbf{r}^{(1)}} \dots (\mathbf{h_l'})^{\mathbf{r}^{(l)}})
\end{aligned}$$

Where the last equality follows by :

$$\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} x$$

and

$$\mathbf{r^{(j)}} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m, j \in \{1,\dots,l\}$$

Moreover, the prover $\mathcal{P}$ follows the protocol honestly (by assumption) and computes $\hat{t}^{(j)}$ as $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \implies \sum_{j=1}^l \phi^{j-1} \hat{t}^{(j)} = \sum_{j=1}^l \phi^{j-1} < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$.
Hence, since the "Bulletproofs" argument is complete and also since $\mathcal{P}$ runs it honestly knowing the valid witness, $\mathcal{V}$ will accept the "Bulletproofs" inner-product argument. The "Bulletproofs" inner product argument accepted together with the verification equation $(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}}) h^{\tau_x} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$ satisfied implies that $\mathcal{V}$ will accept the proof and thereby shows the *Perfect Completeness* of the protocol.

7.4.2 Perfect Special Honest-Verifier Zero-Knowledge

To show *Perfect Special Honest-Verifier Zero-Knowledge* (SHVZK) we build explicitly an efficient simulator that given the C.R.S. and a statement ($g_1,\dots,g_l,h \in \mathbb{G}$ and $\mathbf{V}=(V_1,\dots,V_m) \in \mathbb{G}$ respectively) produces

indistinguishable transcript (in the sense that the transcript will have identical distribution) from a transcript resulting from the true interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. That is :

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

The simulator pseudo-code is as follows :

0-1 Simulator

- Input : $(g_1,\dots,g_l,h,u,\mathbf{g_1},\dots,\mathbf{g_l},\mathbf{h_1},\dots,\mathbf{h_l},\mathbf{V})$.
- Output : transcript identically distributed to a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

- step 1 : Picks challenges x,y,z,ϕ uniformly at random in $\mathbb{Z}_p^*$.
- step 2 : Compute $\mathbf{h'_j}[i]=(\mathbf{h_j}[i])^{y^{-i+1}}$, $i\in\{1,\dots,m\}$, $\mathbf{h'_j}=(\mathbf{h'_j}[1],\dots,\mathbf{h'_j}[m])$, $j\in[1,l]$.
- step 3 : Compute $\mathbf{g'_j}[i]=(\mathbf{g_j}[i])^{\phi^{-j+1}}$, $i\in\{1,\dots,m\}$, $\mathbf{g'_j}=(\mathbf{g'_j}[1],\dots,\mathbf{g'_j}[m])$, $j\in[1,l]$.
- step 4 : Picks μ,τ_x uniformly at random in $\mathbb{Z}_p^*$.
- step 5 : Picks $\mathbf{l^{(j)}},\mathbf{r^{(j)}}$ uniformly at random in $\mathbb{Z}_p^m$, $j\in\{1,\dots,l\}$.
- step 6 : Computes $\hat{t}^{(j)}=<\mathbf{l^{(j)}},\mathbf{r^{(j)}}>$, $j\in\{1,\dots,l\}$.
- step 7 : $c:=\sum_{j=1}^{l}\hat{t}^{(j)}\phi^{j-1}$.
- step 8 : Picks A uniformly at random in $\mathbb{G}$.
- step 9 : Computes

$$S=\left(h^{-\mu}(\mathbf{g'_1}^{-\phi^0\mathbf{l^{(1)}}}\dots\mathbf{g'_l}^{-\phi^{l-1}\mathbf{l^{(l)}}})(\mathbf{h'_1}^{-\mathbf{r^{(1)}}}\dots\mathbf{h'_l}^{-\mathbf{r^{(l)}}})A(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\right)^{-x^{-1}}$$

- step 10 : Picks T_2 uniformly at random in $\mathbb{G}$.
- step 11 : Compute $T_1=\left(\mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_2^{x^2}(g_1^{-\hat{t}^{(1)}}\dots g_l^{-\hat{t}^{(l)}})h^{-\tau_x}\right)^{-x^{-1}}$.
- step 12 : Compute $P=AS^x(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$.
- step 13 : Given the witness $\mathbf{a}:=(\mathbf{l^{(j)}}\phi^{j-1})_{j=1}^{l}$ and $\mathbf{b}:=(\mathbf{r^{(j)}})_{j=1}^{l}$ run the protocol 2 (6.4) on input $((\mathbf{g'_1}||\dots||\mathbf{g'_l}),(\mathbf{h'_1}||\dots||\mathbf{h'_l}),P,c;\mathbf{a},\mathbf{b})$ to get transcript S^{inner} of the inner-product argument.
- step 14 : Output : $(A,S;y,z;T_1,T_2;x;\tau_x,\mu,\{\hat{t}^{(j)}\}_{j=1}^{l};\phi;S^{inner})$.

An honest prover interacting with an honest verifier will produce independent random values $A,\mathbf{l^{(j)}},\mathbf{r^{(j)}},\mu,\tau_x$ given $\alpha,\rho,\tau_1,\tau_2,x,y,z$ are chosen independently and randomly. $\mathcal{P}$ and $\mathcal{V}$ will produce values $\hat{t}^{(j)}=<\mathbf{l^{(j)}},\mathbf{r^{(j)}}>$, $j\in\{1,\dots,l\}$.
The simulated commitment S is fully defined by :
$AS^x(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}=h^{\mu}(\mathbf{g'_1}^{\phi^0\cdot\mathbf{l^{(1)}}}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{l^{(l)}}})((\mathbf{h'_1})^{\mathbf{r^{(1)}}}\dots(\mathbf{h'_l})^{\mathbf{r^{(l)}}})$, which is ensured by computing S accordingly. Given that $\mathcal{P}$ follows the protocol honestly, T_1,T_2 are perfectly hiding multi-Pedersen commitments and are thus random group elements. Given $\hat{t}$ and τ_x, the internal relation between T_1 and T_2 is fully defined by the equation $(g_1^{\hat{t}^{(1)}}\dots g_l^{\hat{t}^{(l)}})h^{\tau_x}=\mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_1^xT_2^{x^2}$.

S^{inner} is ran knowing the true witness for this sub-protocol, $(\mathbf{l^{(1)}}\phi^0||\dots||\mathbf{l^{(l)}}\phi^{l-1}),(\mathbf{r^{(1)}}||\dots||\mathbf{r^{(l)}})$ such that :
$\bar{t}=<(\mathbf{l^{(1)}}\phi^0||\dots||\mathbf{l^{(l)}}\phi^{l-1}),(\mathbf{r^{(1)}}||\dots||\mathbf{r^{l)}})>$ and $AS^x(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$
$=(\mathbf{g_1}^{-\phi^0}||\dots||\mathbf{g_l}^{\phi^{l-1}})^{(\mathbf{l^{(1)}}\phi^0||\dots||\mathbf{l^{(l)}}\phi^{l-1})}(\mathbf{h'_1}||\dots||\mathbf{h'_l})^{(\mathbf{r^{(1)}}||\dots||\mathbf{r^{l)}})}$ so the transcript produced for the first "Bulletproofs" inner product argument inside the protocol will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

We can thus conclude that the transcript obtained by the simulator pseudo-code is distributed identically to the transcript of a true protocol interaction between honest $\mathcal{P}$ and $\mathcal{V}$ with independently and uniformly selected challenges.
This shows our protocol proof has the *Perfect Special Honest-Verifier Zero-Knowledge* property.

7.4.3 Computational Witness Extended-Emulation

In this section, in order to prove the *Computational Witness Extended-Emulation* property, we construct an efficient extractor χ that uses χ^{BP}, the extractor of the "Bulletproofs" inner-product argument as a subroutine and a polynomial number of $N^{0-1-FT} := 3 \cdot m \cdot (m+2) \cdot l \cdot N^{BP}(ml) \cdot = O(m^4 l^3)$ transcripts with 3 different values of the challenge x : x_1, x_2, x_3, m different values of the challenge y : $\{y_j\}_{j=1}^{m}$, $m+2$ different values of the challenge z : $\{z_j\}_{j=1}^{m+2}$, l different values of the challenge ϕ : $\{\phi_i\}_{i=1}^{l}$ and $(ml)^2$ for the "Bulletproofs" inner product argument to extract a valid witness for relation $R_1 \equiv \{(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m; (v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}_p^m) : V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\}, \forall i \in [1,l], j \in [1,m]\}$.

This allows to prove, using the Forking Lemma (3.4), the *Computational Witness Extended Emulation* (3.3.4) property.

We will use the *Discrete Log Relation* assumption, which was presented in section 3.1.

The following proof may seem a bit long and intricate but it is not very complicated. All that is required is to build the extractor χ.
Informally the extractor will work as follows: χ first uses the "Bulletproofs" inner product argument extractor, χ^{BP} to extract the witnesses $(\phi^0 \mathbf{l^{(1)}} || \dots || \phi^{l-1} \mathbf{l^{(l)}})$ and $(\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}})$.
Then we combine different transcripts to open the commitments S, A, T_1, T_2 and finally $\{V_j\}_{j=1}^{m}$, which leads to a candidate witness for relation R_1. We then use the verification equations and the fact that $< (\phi^0 \mathbf{l^{(1)}} || \dots || \phi^{l-1} \mathbf{l^{(l)}}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) >= \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$ to show that the extracted witness satisfy the relation R_1.

We denote in this section :

- $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $\mathbf{h} := (\mathbf{h'_1} || \dots || \mathbf{h'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $\overline{P} := P h^{-\mu} \in \mathbb{G}$,
- $c := \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1} \in \mathbb{Z}_p$.
- $\phi = (\phi^0, \dots, \phi^{l-1}) \in \mathbb{Z}_p$.

Our extractor χ first runs the extractor of the "Bulletproofs" inner-product argument, χ^{BP}, on each of the transcripts to obtain the witnesses $(\tilde{\mathbf{l}}, \tilde{\mathbf{r}}) = \left((\tilde{\mathbf{l}}^{(\mathbf{1})} || \dots || \tilde{\mathbf{l}}^{(\mathbf{l})}), (\tilde{\mathbf{r}}^{(\mathbf{1})} || \dots || \tilde{\mathbf{r}}^{(\mathbf{l})})\right) \in \mathbb{Z}_p^{(m \cdot l)} \times \mathbb{Z}_p^{(m \cdot l)}$ such that: $\mathbf{g}^{\tilde{\mathbf{l}}} \mathbf{h}^{\tilde{\mathbf{r}}} = \overline{P} \wedge < \tilde{\mathbf{l}}, \tilde{\mathbf{r}} >= c$.
Now, consider two valid transcripts with different values of the challenge x: $x_1, x_2 \in \mathbb{Z}_p^*$ and their inner-product argument witnesses $\tilde{\mathbf{l}}_\mathbf{1}, \tilde{\mathbf{r}}_\mathbf{1}, \tilde{\mathbf{l}}_\mathbf{2}, \tilde{\mathbf{r}}_\mathbf{2}$. We have : $\mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}} = \overline{P}_i$, which is equivalent to :
$\mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}} h^{\mu_i} = A S^{x_i} (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}, i \in \{1,2\}$.
We combine these two equations to open the commitments A and S and extract their committed values as follows.
Let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 0, \sum_{i=1}^{2} \nu_i x_i = 1$. Then, we can compute $\prod_{i=1}^{2} (P_i)^{\nu_i}$ in two ways:

- (i) $\prod_{i=1}^{2} (P_i)^{\nu_i} = \prod_{i=1}^{2} \left(h^{\mu_i} \mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}} \right)^{\nu_i} = h^{\sum_{i=1}^{2} \nu_i \mu_i} \mathbf{g}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}}$
  $= h^{\sum_{i=1}^{2} \nu_i \mu_i} (\mathbf{g'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(1)}} \dots \mathbf{g'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{l})}}) (\mathbf{h'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(1)}} \dots \mathbf{h'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(\mathbf{l})}})$.

- (ii) $\prod_{i=1}^{2} (P_i)^{\nu_i} = \prod_{i=1}^{2} \left(A S^{x_i} (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \right)^{\nu_i}$
  $= A^{\sum_{i=1}^{2} \nu_i} S^{\sum_{i=1}^{2} \nu_i x_i} (\mathbf{g'_1}^{\phi^0 \sum_{i=1}^{2} \nu_i \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \sum_{i=1}^{2} \nu_i \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1}^{\sum_{i=1}^{2} \nu_i \cdot \mathbf{1}^m} \dots \mathbf{h'_l}^{\sum_{i=1}^{2} \nu_i \cdot \mathbf{1}^m})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$
  $= A^0 S^1 (\mathbf{g'_1}^{\mathbf{0}^m} \dots \mathbf{g'_l}^{\mathbf{0}^m})^{-z} (\mathbf{h'_1}^{\mathbf{0}^m} \dots \mathbf{h'_l}^{\mathbf{0}^m})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$
  $= S$.

We thus have : $S = \prod_{i=1}^{2} (P_i)^{\nu_i} = h^{\sum_{i=1}^{2} \nu_i \mu_i} (\mathbf{g'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(1)}} \dots \mathbf{g'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{l})}}) (\mathbf{h'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(1)}} \dots \mathbf{h'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(\mathbf{l})}})$.
Hence, we can write $S = h^{\rho} (\mathbf{g_1}^{\mathbf{s_L}^{(1)}} \dots \mathbf{g_l}^{\mathbf{s_L}^{(\mathbf{l})}}) (\mathbf{h_1}^{\mathbf{s_R}^{(1)}} \dots \mathbf{h_l}^{\mathbf{s_R}^{(\mathbf{l})}})$, where :

- $\rho = \sum_{i=1}^{2} \nu_i \mu_i$,
- $\mathbf{s_L^{(j)}} = \phi^{-j+1} (\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{j})})$, $j \in \{1, \dots, l\}$ and

- $\mathbf{s_R^{(j)}} = \mathbf{y}^{-m}(\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i})$, $j \in \{1, \dots, l\}$.

Now similarly, let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 1, \sum_{i=1}^{2} \nu_i x_i = 0$. Then, once again we can do the same computations to get:

$$\begin{aligned}\prod_{i=1}^{2}(P_i)^{\nu_i} &= h^{\sum_{i=1}^{2}\nu_i\mu_i}(\mathbf{g'_1}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(1)}} \dots \mathbf{g'_l}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(l)}})(\mathbf{h'_1}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(1)}} \dots \mathbf{h'_l}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(l)}}) \\ &= A^{\sum_{i=1}^{2}\nu_i} S^{\sum_{i=1}^{2}\nu_i x_i}(\mathbf{g'_1}^{\phi^0\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1}\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m} \dots \mathbf{h'_l}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m} \\ &= A(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\end{aligned}$$

Hence, we can write $A = h^{\alpha}(\mathbf{g_1}^{\mathbf{a_L^{(1)}}} \dots \mathbf{g_l}^{\mathbf{a_L^{(l)}}})(\mathbf{h_1}^{\mathbf{a_R^{(1)}}} \dots \mathbf{h_l}^{\mathbf{a_R^{(l)}}})$, where :

- $\alpha = \sum_{i=1}^{2} \nu_i\mu_i$,
- $\mathbf{a_L^{(j)}} = \sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{j})}\phi^{-j+1} + z \cdot \mathbf{1}^m$, $j \in \{1, \dots, l\}$ and
- $\mathbf{a_R^{(j)}} = \mathbf{y}^{-m}\left(\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(\mathbf{j})} - z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m\right)$, $j \in \{1, \dots, l\}$.

Notice that since A and S are provided before the choice of challenges x, y, z, ϕ, we can assume these two expressions hold for any challenge x, y, z, ϕ provided later to the prover in the transcripts considered by our extractor.

Now, we can use the 2 expressions we just found for A and S and substitute them in the equation:
$h^{\mu}\mathbf{g'_1}^{\tilde{\mathbf{l}}^{(1)}} \dots \mathbf{g'_l}^{\tilde{\mathbf{l}}^{(l)}}\mathbf{h'_1}^{\tilde{\mathbf{r}}^{(1)}} \dots \mathbf{h'_l}^{\tilde{\mathbf{r}}^{(l)}} = AS^x(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$.

In this way, we obtain the equation:
$h^{\mu}\mathbf{g'_1}^{\tilde{\mathbf{l}}^{(1)}} \dots \mathbf{g'_l}^{\tilde{\mathbf{l}}^{(l)}}\mathbf{h'_1}^{\tilde{\mathbf{r}}^{(1)}} \dots \mathbf{h'_l}^{\tilde{\mathbf{r}}^{(l)}} = \left(h^{\alpha}(\mathbf{g'_1}^{\phi^0\mathbf{a_L^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1}\mathbf{a_L^{(l)}}})((\mathbf{h'_1}^{\mathbf{y}^m\circ\mathbf{a_R^{(1)}}} \dots \mathbf{h'_l}^{\mathbf{y}^m\circ\mathbf{a_R^{(l)}}})\right)$
$\left(h^{\rho x}(\mathbf{g'_1}^{\phi^0\mathbf{s_L^{(1)}}x} \dots \mathbf{g'_l}^{\phi^{l-1}\mathbf{s_L^{(l)}}x})((\mathbf{h'_1}^{\mathbf{y}^m\circ\mathbf{s_R^{(1)}}x} \dots \mathbf{h'_l}^{\mathbf{y}^m\circ\mathbf{s_R^{(l)}}x})\right)(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$

Now using the *Discrete Log Relation* assumption we can infer the following except with negligible probability:

- (i) $\mu = \alpha + \rho x$,
- (ii) $\tilde{\mathbf{l}}^{(\mathbf{j})} = \phi^{j-1}(\mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(j)}}x)$, $j \in \{1, \dots, l\}$,
- (iii) $\tilde{\mathbf{r}}^{(\mathbf{j})} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2 \cdot \mathbf{z}^m$, $j \in \{1, \dots, l\}$.

We can thus write $\tilde{\mathbf{l}}^{(\mathbf{j})} = \phi^{j-1}\mathbf{l}^{(\mathbf{j})}$ and $\tilde{\mathbf{r}}^{(\mathbf{j})} = \mathbf{r}^{(\mathbf{j})}$, where $\mathbf{l}^{(\mathbf{j})} = \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(j)}}x$, (1)
and $\mathbf{r}^{(\mathbf{j})} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2 \cdot \mathbf{z}^m$, $j \in \{1, \dots, l\}$, (2).

Now, recall that $\tilde{\mathbf{l}}, \tilde{\mathbf{r}}$ satisfy : $< \tilde{\mathbf{l}}, \tilde{\mathbf{r}} >= c := \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1}$ or just written differently :
$< (\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) >= \sum_{j=1}^{l} < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \phi^{j-1} = \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1}$.

This last equality holds for each of our transcripts and thus in particular for all l different challenges ϕ. Notice that since, from our early discussion, the expressions of S and A are independent of the value of the challenge ϕ, we can conclude that the $\mathbf{a_R^{(j)}}$, $\mathbf{a_L^{(j)}}$ and consequently the $\mathbf{l}^{(\mathbf{j})}$ and $\mathbf{r}^{(\mathbf{j})}$ are also independent of the value of the challenge ϕ. Hence, if we define $p(\phi) := \sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$, it is a polynomial of degree $l-1$ since the term with highest degree will be ϕ^{l-1} and all coefficients $(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$ are independent of ϕ.
Denote $\Phi = \{\phi_k\}_{k=1}^{l}$, the set of these l different challenges ϕ; then we have :
$\sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >) = 0 \quad \forall \phi \in \{\phi_k\}_{k=1}^{l}$. Since we assume $\phi_i \neq \phi_j$, $i \neq j$, this implies that the polynomial $p(\phi) = \sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$ has (at least) l distinct roots $\{\phi_k\}_{k=1}^{l}$. The only polynomial of degree $l-1$ with more than l distinct roots is the polynomial 0, with 0 coefficients. Hence, $p(\phi)$ is the 0 polynomial and thereby all his coefficients are 0, that is:
$\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >= 0 \Leftrightarrow \hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \forall j \in \{1, \dots, l\}$. (3)

Now, given the challenges z, y, ϕ, we consider transcripts with 3 different challenges $x : x_1, x_2, x_3$ and the verification equation : $(g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}})h^{\tau_{x_i}} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2}$, that must be verified $\forall i \in \{1,2,3\}$.
Let $\{\nu_j\}_{j=1}^3 \subset \mathbb{Z}_p$ such that $\sum_{j=1}^3 \nu_j = 0, \sum_{j=1}^3 \nu_j x_j = 1, \sum_{j=1}^3 \nu_j x_j^2 = 0$, then we can compute :

$$\begin{aligned}
\prod_{i=1}^3 \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}})h^{\tau_{x_i}}\right)^{\nu_i} &= (g_1^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(l)}})h^{\sum_{i=1}^3 \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^3 \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2}\right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^3 \nu_i \cdot z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\sum_{i=1}^3 \nu_i \delta(y,z)} T_1^{\sum_{i=1}^3 \nu_i x_i} T_2^{\sum_{i=1}^3 \nu_i x_i^2} \\
&= T_1
\end{aligned}$$

Hence, we can write $T_1 = h^{\tau_1} g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}}$, where $\tau_1 = \sum_{i=1}^3 \nu_i \tau_{x_i}$, $t_1^{(j)} = \sum_{i=1}^3 \nu_i \hat{t}_i^{(j)}$, $j \in \{1, \dots, l\}$.

Similarly, let $\{\nu_j\}_{j=1}^3 \subset \mathbb{Z}_p$ such that $\sum_{j=1}^3 \nu_j = 0, \sum_{j=1}^3 \nu_j x_j = 0, \sum_{j=1}^3 \nu_j x_j^2 = 1$, then we can compute:

$$\begin{aligned}
\prod_{i=1}^3 \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}})h^{\tau_{x_i}}\right)^{\nu_i} &= (g_1^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(l)}})h^{\sum_{i=1}^3 \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^3 \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2}\right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^3 \nu_i z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\sum_{i=1}^3 \nu_i \delta(y,z)} T_1^{\sum_{i=1}^3 \nu_i x_i} T_2^{\sum_{i=1}^3 \nu_i x_i^2} \\
&= T_2
\end{aligned}$$

Hence, we can write $T_2 = h^{\tau_2} g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}}$, where $\tau_2 = \sum_{i=1}^3 \nu_i \tau_{x_i}$, $t_2^{(j)} = \sum_{i=1}^3 \nu_i \hat{t}_i^{(j)}$, $j \in \{1, \dots, l\}$.

Finally, let $\{\nu_j\}_{j=1}^3 \subset \mathbb{Z}_p$ such that $\sum_{j=1}^3 \nu_j = 1, \sum_{j=1}^3 \nu_j x_j = 0, \sum_{j=1}^3 \nu_j x_j^2 = 0$, then we can compute :

$$\begin{aligned}
\prod_{i=1}^3 \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}})h^{\tau_{x_i}}\right)^{\nu_i} &= (g_1^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^3 \nu_i \hat{t}_i^{(l)}})h^{\sum_{i=1}^3 \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^3 \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2}\right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^3 \nu_i z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\sum_{i=1}^3 \nu_i \delta(y,z)} T_1^{\sum_{i=1}^3 \nu_i x_i} T_2^{\sum_{i=1}^3 \nu_i x_i^2} \\
&= \mathbf{V}^{z^2 \cdot \mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)}
\end{aligned}$$

Hence, we can write $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} = h^{\gamma} g_1^{v_1} \dots g_l^{v_l}$, where $\gamma = \sum_{i=1}^3 \nu_i \tau_{x_i}$ and $v_j = \sum_{i=1}^3 \nu_i \hat{t}_i^{(j)} - \delta(y,z)$, $j \in \{1, \dots, l\}$.

Now, given y, x, ϕ, consider m transcripts with different challenges $\{z_j\}_{j=1}^m$. From our previous discussion, we can write $\mathbf{V}^{z_j^2 \cdot \mathbf{z_j}^m} = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$.
Given $j \in \{1, \dots, m\}$, let $\{\nu_i\}_{i=1}^m \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^m \nu_i z_i^{1+j} = 1$ and $\sum_{i=1}^m \nu_i z_i^{1+k} = 0 \ \forall k \in (\{1, \dots, m\} \backslash \{j\})$.
In other words, $\{\nu_i\}_{i=1}^m$ are such that given $k \in \{1, \dots, m\}$: $\sum_{i=1}^m z_i^{k+1} \nu_i = \Gamma(k,j) := \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{otherwise.} \end{cases}$
In what follows below, we denote V_j as the j-th component of the m-dimensional vector V, i.e. $\mathbf{V} = (V_j)_{j=1}^m = (V_1, \dots, V_m)$.
Then, we can compute the product $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i}$ as :

- $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^m \left(\prod_{k=1}^m V_k^{z_i^{k+1}}\right)^{\nu_i} = \prod_{k=1}^m \left[\prod_{i=1}^m \left(V_k^{z_i^{k+1}}\right)^{\nu_i}\right] = \prod_{k=1}^m \left(V_k^{\sum_{i=1}^m z_i^{k+1} \nu_i}\right)$
  $= \prod_{k=1}^m V_k^{\Gamma(k,j)} = V_j^{\Gamma(j,j)} = V_j$, since $V_k^{\Gamma(k,j)} = 1 \ \forall k \neq j$.

- $\prod_{i=1}^{m}\left(\mathbf{V}^{z_i^2\mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^{m}\left(h^{\gamma_i}g_1^{v_1^{(i)}}\dots g_l^{v_l^{(i)}}\right)^{\nu_i} = h^{\sum_{i=1}^{m}\gamma_i\nu_i}g_1^{\sum_{i=1}^{m}v_1^{(i)}\nu_i}\dots g_l^{\sum_{i=1}^{m}v_l^{(i)}\nu_i}$

Hence given any $j \in \{1,\dots,m\}$, we can write $V_j = h^{\overline{\gamma}_j}g_1^{\overline{v}_1^{(j)}}\dots g_l^{\overline{v}_l^{(j)}}$, where $\overline{\gamma}_j = \sum_{i=1}^{m}\gamma_i\nu_i$ and $\overline{v}_k^{(j)} = \sum_{i=1}^{m}v_k^{(i)}\nu_i$, $k \in \{1,\dots,l\}$.

We have succeed to extract a candidate witness $\{(\overline{\gamma}_j, v_1^{(j)},\dots,v_l^{(j)})\}$ such that $h^{\overline{\gamma_j}}g_1^{v_1^{(j)}}\dots g_l^{v_l^{(j)}} = V_j$.
It remains to show that $\{(\overline{\gamma}_j, v_1^{(j)},\dots,v_l^{(j)})\}$ satisfy the relation R_1, that is :
$v_i^{(j)} \in \{0,1\}\ \forall(i,j) \in \{1,\dots,l\}\times\{1,\dots,m\}$.
We show this by recombining all the expressions we found earlier for the commitments and exploiting the different verification equations.

Now, given any transcript with fixed challenges x, y, z, ϕ we can substitute the expressions we just found for $(V_j)_{j=1}^m$, T_1 and T_2 in the equation, $(g_1^{\hat{t}^{(1)}}\dots g_l^{\hat{t}^{(l)}})h^{\tau_x} = \mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_1^xT_2^{x^2}$ to get :

$$\begin{aligned}
(g_1^{\hat{t}^{(1)}}\dots g_l^{\hat{t}^{(l)}})h^{\tau_x} &= \mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_1^xT_2^{x^2}\\
&= \prod_{j=1}^{m}V_j^{z^{j+1}}(g_1\dots g_l)^{\delta(y,z)}(h^{\tau_1}g_1^{t_1^{(1)}}\dots g_l^{t_1^{(l)}})^x(h^{\tau_2}g_1^{t_2^{(1)}}\dots g_l^{t_2^{(l)}})^{x^2}\\
&= \prod_{j=1}^{m}(h^{\overline{\gamma}_j}g_1^{\overline{v}_1^{(j)}}\dots g_l^{\overline{v}_l^{(j)}})^{z^{j+1}}(g_1\dots g_l)^{\delta(y,z)}h^{\tau_1x}g_1^{t_1^{(1)}x}\dots g_l^{t_1^{(l)}x}h^{\tau_2x^2}g_1^{t_2^{(1)}x^2}\dots g_l^{t_2^{(l)}x^2}\\
&= (h^{\sum_{j=1}^{m}\overline{\gamma}_jz^{j+1}}g_1^{\sum_{j=1}^{m}\overline{v}_1^{(j)}z^{j+1}}\dots g_l^{\sum_{j=1}^{m}\overline{v}_l^{(j)}z^{j+1}})g_1^{\delta(y,z)}\dots g_l^{\delta(y,z)}h^{\tau_1x+\tau_2x^2}g_1^{t_1^{(1)}x+t_2^{(1)}x^2}\dots g_l^{t_1^{(l)}x+t_2^{(l)}x^2}\\
&= h^{\tau_1x+\tau_2x^2+\sum_{j=1}^{m}\overline{\gamma}_jz^{j+1}}g_1^{\sum_{j=1}^{m}\overline{v}_1^{(j)}z^{j+1}+\delta(y,z)+t_1^{(1)}x+t_2^{(1)}x^2}\dots g_l^{\sum_{j=1}^{m}\overline{v}_l^{(j)}z^{j+1}+\delta(y,z)+t_1^{(l)}x+t_2^{(l)}x^2}
\end{aligned}$$

Now using the *Discrete Log Relation* assumption we can infer the following except with negligible probability :

- (i) $\tau_x = \tau_1x + \tau_2x^2 + \sum_{j=1}^{m}\overline{\gamma}_jz^{j+1}$,
- (ii) $\hat{t}^{(j)} = \sum_{k=1}^{m}\overline{v}_j^{(k)}z^{k+1} + \delta(y,z) + t_1^{(j)}x + t_2^{(j)}x^2$, $j \in \{1,\dots,l\}$.

We can rewrite the last expression as : $\hat{t}^{(j)} = t_0^{(j)} + t_1^{(j)}x + t_2^{(j)}x^2$ (4), $j \in \{1,\dots,l\}$, where
$t_0^{(j)} = \sum_{k=1}^{m}\overline{v}_j^{(k)}z^{k+1} + \delta(y,z)$. For the rest of the proof, it remains to show that
$t_0^{(j)} =< \mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m, \mathbf{y}^m\circ(\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m) + z^2\cdot\mathbf{z}^m > \forall j \in \{1,\dots,l\}$.
Once we have this we can show that that $\{(\gamma_j, v_1^{(j)},\dots,v_l^{(j)})\}$ is a valid witness for relation R_1.
Indeed, recall that in section 7.1 we have mentioned that the equation $< \mathbf{a_L^{(j)}} - z\mathbf{1}^m, \mathbf{y}^m\circ(\mathbf{a_R^{(j)}} + z\mathbf{1}^m) + z^2\cdot\mathbf{z}^m >= \sum_{k=1}^{m}z^{1+k}v_j^{(k)} + \delta(y,z)$ implies except with negligible probability the three following equations $\forall j \in [1,l]$: $\mathbf{a_L^{(j)}}[i] = v_j^{(i)}, i \in \{1,\dots,m\}, \mathbf{a_L^{(j)}}\circ\mathbf{a_R^{(j)}} = \mathbf{0}^m, \mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$, which proves $v_i^{(j)} \in \{0,1\}$.

For this, given $j \in [1,l]$, define the two linear vectors polynomials $p_L^{(j)}(X), p_R^{(j)}(X) \in \mathbb{Z}_p^m[X]$ and the two quadratic scalar polynomials $p^{(j)}(X), t^{(j)}(X) \in \mathbb{Z}_p[X]$ that will depend of the challenge x as follows:

- (i) $p_L^{(j)}(X) = \mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m + \mathbf{s_L^{(j)}}X$,
- (ii) $p_R^{(j)}(X) = \mathbf{y}^m\circ(\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m + \mathbf{s_R^{(j)}}X) + z^2\cdot\mathbf{z}^m$,
- (iii) $p^{(j)}(X) =< p_L^{(j)}(X), p_R^{(j)}(X) >= p_0^{(j)} + p_1^{(j)}X + p_2^{(j)}X^2$,
- (iv) $t^{(j)}(X) = t_0^{(j)} + t_1^{(j)}X + t_2^{(j)}X^2$.

Now, recall that given a transcript with challenges x, y, z, ϕ, we have shown earlier (3) : $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$ and by (4) we have : $\hat{t}^{(j)} = t^{(j)}(x)$ so that $t^{(j)}(x) =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$.
Moreover, from (1) and (2) we have the following: $\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m + \mathbf{s_L^{(j)}}x = p_L^{(j)}(x)$ and
$\mathbf{r^{(j)}} = \mathbf{y}^m\circ(\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2\cdot\mathbf{z}^m = p_R^{(j)}(x)$.

This implies in particular that for all challenges z, y, ϕ and 3 different challenges x_1, x_2, x_3, we have $t^{(j)}(x_i) =< p_L^{(j)}(x_i), p_R^{(j)}(x_i) >= p^{(j)}(x_i)$, $i \in \{1,2,3\}$. This can be expressed equivalently as the fact that the scalar quadratic polynomial $\sum_{k=0}^{2}(t_k^{(j)} - p_k^{(j)})X^k \in \mathbb{Z}_p[X]$ admits at least the 3 following roots: (x_1, x_2, x_3). Since the only quadratic polynomial in $\mathbb{Z}_p$ with more than 2 roots is the zero polynomial, we can conclude that $t_k^{(j)} - p_k^{(j)} = 0 \Leftrightarrow t_k^{(j)} = p_k^{(j)}\ \forall k \in \{0,1,2\}$.
In particular, $t_0^{(j)} = p_0^{(j)} \Leftrightarrow \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + \delta(y,z) =< \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >$.

Now, using the proof in appendix A, this can be expressed equivalently as :

$$\sum_{k=1}^{m}(\mathbf{a_L^{(j)}}[k] - v_j^{(k)})z^{k+1} + z < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m > + < \mathbf{a_L^{(j)}}, \mathbf{a_R^{(j)}} \circ \mathbf{y}^m >= 0$$

Since this holds for $m+2$ different challenges z and m different challenges y (using again the reasoning that a polynomial of degree m with more than m roots is the 0 polynomial) we can infer the following :

- (i) $\mathbf{a_L^{(j)}} \circ \mathbf{a_R^{(j)}} = \mathbf{0}^m$,
- (ii) $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$,
- (iii) $v_j^{(k)} = \mathbf{a_L^{(j)}}[k]\ \forall k \in \{1, \dots, l\}$.

(i) and (ii) together imply that $\mathbf{a_L^{(j)}} \in \{0,1\}^m$ while (iii) implies $v_i^{(j)} \in \{0,1\}\ \forall i \in \{1, \dots, l\}$.
Since all this final reasoning holds for any $j \in \{1, \dots, m\}$ and $V_j = h^{\overline{\gamma_j}} g_1^{\overline{v}_1^{(j)}} \dots g_l^{\overline{v}_l^{(j)}}\ \forall j \in \{1, \dots, m\}$, we have extracted a valid witness $(\overline{v_1}, \dots, \overline{v_l}, \overline{\gamma})$ for the relation:

$$R_1 = \{(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m; v_1, \dots, v_l, \gamma \in \mathbb{Z}_p^m) :$$
$$V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\}, \forall i \in [1,l], j \in [1,m]\}.$$

Using the *Forking Lemma* presented in section 3.4 this implies the *Computational Witness Extended Emulation* property (3.3.4) which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The 0-1 protocol is in $6 + (2log_2(ml) + 1) = 7 + 2log_2(ml)$ moves and has $3 + log_2(ml)$ challenge $(y,z), x, \phi$ and $log_2(ml)$ challenges from the "Bulletproofs" inner product argument. Hence the parameter μ in the statement of the *Forking Lemma* is equal to $3 + log_2(ml)$. The corresponding tree of accepting transcripts has a depth equal to $\mu = 3 + log_2(ml)$. The node in the first level has $n_1 = m(m+2)$ children in the second level of the tree corresponding to the $m(m+2)$ pairs of challenges (y,z) for the first challenge $(\{y_j\}_{j=1}^{m} \times \{z_j\}_{j=1}^{m+2})$. Each node in the second level has $n_2 = 3$ children each corresponding to a different value of the challenge x $(\{x_1, x_2, x_3\})$. Each node in the third level has $n_3 = l$ children corresponding to the l different challenges ϕ $(\{\phi_i\}_{i=1}^{l})$. Finally, each node in the fourth level of our tree is connected to the root of a tree of accepting transcripts of the "Bulletproofs" inner product argument, which is a tree of depth $log_2(ml)$ with 4 children per node (see the "Bulletproofs" paper), hence $n_i = 4$, $i \geq 4$
Therefore, the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = \prod_{i=1}^{3+log_2(ml)} n_i = n_1 \cdot n_2 \cdot n_3 \prod_{i=4}^{3+log_2(ml)} n_i = m(m+2) \cdot 3 \cdot l \cdot \prod_{i=1}^{log_2(ml)} n_{i+4} = 3l(m^2+2m) \prod_{i=1}^{log_2(ml)} n_{i+4} = 3l(m^2+2m) \prod_{i=1}^{log_2(ml)} 4 = 3lm^2 4^{log_2(ml)} = 3l(m^2+2m)(ml)^2 = 3l^3(m^4+2m^3) = N^{0-1-FT}(ml)$, which is clearly bounded above by a polynomial in the parameter λ (the constant polynomial $3l^3(m^4+2m^3)$ for instance). In addition, χ is in polynomial time since computing the ν_i amounts each time to solving a system of linear equations in $\mathbb{Z}_p$ for example for the opening of the commitment A: $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, which can be done efficiently. All other operations are also basic polynomial time computations. Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute the ν_i to open a commitment, which occurs only when the determinant of the matrix is 0 (the linear system of equations has no solution then), which happens only with negligible probability. For example for the opening of the commitment A, the matrix corresponding to the linear system of equation in $\mathbb{Z}_p$ is $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$. The determinant is $x_1 - x_2$. It is zero when $x_1 = x_2$, which occurs with negligible probability $(1/p)$ since x_1 and x_2 are picked uniformly at random in $\mathbb{Z}_p$.

Chapter 8

Logarithmic 0-1 Proof

8.1 Key Ideas

The problem of the 0-1 protocol system is the term linear in l in its data complexity $O(l+log_2(m\cdot l))$. What creates this linear in l complexity term is the communication of $\{\hat{t}^{(j)}\}_{j=1}^{l}$. In order to reduce this communication a good idea is to pick a new base $(h_1,\ldots,h_l)$ and instead of checking the identity $(g_1^{\hat{t}^{(1)}}\ldots g_l^{\hat{t}^{(l)}})h^{\tau_x} = \mathbf{V}^{\mathbf{z}^m\cdot z^2}(g_1\ldots g_l)^{\delta(y,z)}T_1^x T_2^{x^2}$, one can check $(g_1,\ldots,g_l)^{(\hat{t}^{(1)},\ldots,\hat{t}^{(l)})}(h_1,\ldots,h_l)^{(\phi^0,\ldots,\phi^{l-1})} = \overline{P}$, where :
$\overline{P} := h^{-\tau_x}\mathbf{V}^{\mathbf{z}^m\cdot z^2}(g_1\ldots g_l)^{\delta(y,z)}T_1^x T_2^{x^2}(h_1,\ldots,h_l)^{(\phi^0,\ldots,\phi^{l-1})}$ so that we can replace the previous checking by a (second) "Bulletproofs" over the relation :

$$R_5 \equiv \{(\mathbf{g},\mathbf{h}\in\mathbb{G}^n, P\in\mathbb{G}, c\in\mathbb{Z}_p; \mathbf{a},\mathbf{b}\in\mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c =< \mathbf{a},\mathbf{b} >\}$$

, on input :

- $\mathbf{g} := (g_1,\ldots,g_l)\in\mathbb{G}^l$,
- $\mathbf{h} := (h_1,\ldots,h_l)\in\mathbb{G}^l$,
- $P := \overline{P}\in\mathbb{G}$,
- $c := \sum_{j=1}^{l}\hat{t}^{(j)}\phi^{j-1}\in\mathbb{Z}_p$,
- $\mathbf{a} := (\hat{t}^{(1)},\ldots,\hat{t}^{(l)})$,
- $\mathbf{b} := (\phi^0,\ldots,\phi^{l-1})$,

for $n = l$, which allows to get a total proof in complexity $O(log_2(l\cdot m))$.

However, this could allow the prover to cheat by computing earlier T_1 or any other commitment earlier in the proof with a committed value in the base $(h_1,\ldots,h_l)$. This is where we use our "Schnorr's extended argument" to show that $\mathcal{P}$ knows the witness $(\hat{t}^{(1)},\ldots,\hat{t}^{(l)})$ such that $h^{-\tau_x}\mathbf{V}^{\mathbf{z}^m\cdot z^2}(g_1\ldots g_l)^{\delta(y,z)}T_1^x T_2^{x^2} = (g_1,\ldots,g_l)^{(\hat{t}^{(1)},\ldots,\hat{t}^{(1)})}$. Notice that our "Schnorr's extended argument" protocol does not have the zero-knowledge property but it is not a concern since the $(\hat{t}^{(1)},\ldots,\hat{t}^{(l)})$ are blinded and do not leak any information. Recall anyway that in our previous protocol we had the zero-knowledge property while we were sending these values in clear.

8.2 0-1 Argument (Protocol 3)

We first give a formal written description of the protocol and then a diagram to see visually all the exchanges in the 0-1 protocol built to have proofs of logarithmic size.

0-1 Protocol

- Input : $(g_1,\ldots,g_l,h_1,\ldots,h_l,h,u\in\mathbb{G},\mathbf{g_1},\ldots,\mathbf{g_l},\mathbf{h_1},\ldots,\mathbf{h_l},\mathbf{V}=(V_j)_{j=1}^m\in\mathbb{G}^m;\{v_1^{(j)},\ldots,v_l^{(j)},\gamma_j\}_{j=1}^m\subseteq\mathbb{Z}_p^{l+1})$,

- $\mathcal{P}$'s input : $(g_1,\ldots,g_l,h_1,\ldots,h_l,h,u,\mathbf{g_1},\ldots,\mathbf{g_l},\mathbf{h_1},\ldots,\mathbf{h_l},\mathbf{V},\{v_1^{(j)},\ldots,v_l^{(j)},\gamma_j\}_{j=1}^m)$,

- $\mathcal{V}$'s input : $(g_1, \ldots, g_l, h_1, \ldots, h_l, h, u, \mathbf{g_1}, \ldots, \mathbf{g_l}, \mathbf{h_1}, \ldots, \mathbf{h_l}, \mathbf{V})$

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 1 : $\mathcal{P}$ computes :

- $\mathbf{a_L^{(j)}}[i] = v_j^{(i)}$, $i \in \{1, \ldots, m\}$, $\mathbf{a_L^{(j)}} = (\mathbf{a_L^{(j)}}[1], \ldots, \mathbf{a_L^{(j)}}[m])$, $j \in \{1, \ldots, l\}$,
- $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$, $j \in \{1, \ldots, l\}$,
- Pick uniformly at random α, ρ in $\mathbb{Z}_p$ and $\mathbf{s_L^{(j)}}, \mathbf{s_R^{(j)}}$ in $\mathbb{Z}_p$, $j \in \{1, \ldots, l\}$,
- $A = h^{\alpha}(\mathbf{g_1}^{\mathbf{a_L^{(1)}}} \ldots \mathbf{g_l}^{\mathbf{a_L^{(l)}}})(\mathbf{h_1}^{\mathbf{a_R^{(1)}}} \ldots \mathbf{h_l}^{\mathbf{a_R^{(l)}}}) \in \mathbb{G}$,
- $S = h^{\rho}(\mathbf{g_1}^{\mathbf{s_L^{(1)}}} \ldots \mathbf{g_l}^{\mathbf{s_L^{(l)}}})(\mathbf{h_1}^{\mathbf{s_R^{(1)}}} \ldots \mathbf{h_l}^{\mathbf{s_R^{(l)}}}) \in \mathbb{G}$.

- step 2 : $\mathcal{P}$ sends to $\mathcal{V}$: A, S.
- step 3 : $\mathcal{V}$ picks uniformly at random $y, z \in \mathbb{Z}_p^*$.
- step 4 : $\mathcal{V}$ sends y, z to $\mathcal{P}$.

At this state of the proof, given $j \in \{1, \ldots, l\}$ we define $\mathbf{l^{(j)}}(X), \mathbf{r^{(j)}}(X) \in \mathbb{Z}_p^m[X]$ and $t^{(j)}(X) \in \mathbb{Z}_p[X]$ as follows :

$$\begin{aligned}
&\mathbf{l^{(j)}}(X) := \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}}X \in \mathbb{Z}_p^m[X],\\
&\mathbf{r^{(j)}}(X) := \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}}X) + z^2 \cdot \mathbf{z}^m \in \mathbb{Z}_p^m[X],\\
&t^{(j)}(X) :=< \mathbf{l^{(j)}}(X), \mathbf{r^{(j)}}(X) >= t_0^{(j)} + t_1^{(j)}X + t_2^{(j)}X^2 \in \mathbb{Z}_p[X].
\end{aligned}$$

- step 5 : $\mathcal{P}$ computes :

- Pick uniformly at random $\tau_1, \tau_2 \in \mathbb{Z}_p$.
- $t_1^{(j)} =< \mathbf{s_L^{(j)}}, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m > + < \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{s_R^{(j)}} \circ \mathbf{y}^m >$, $j \in \{1, \ldots, l\}$,
- $t_2^{(j)} =< \mathbf{s_L^{(j)}}, \mathbf{y}^m \circ \mathbf{s_R^{(j)}} >$, $j \in \{1, \ldots, l\}$,
- $T_1 = h^{\tau_1}(g_1^{t_1^{(1)}} \ldots g_l^{t_1^{(l)}})$, $T_2 = h^{\tau_2}(g_1^{t_2^{(1)}} \ldots g_l^{t_2^{(l)}})$.

- step 6 : $\mathcal{P}$ sends to $\mathcal{V}$: T_1, T_2.
- step 7 : $\mathcal{V}$ picks uniformly at random x in $\mathbb{Z}_p^*$.
- step 8 : $\mathcal{V}$ sends x to $\mathcal{P}$.
- step 9 : $\mathcal{P}$ computes for all $j \in \{1, \ldots, l\}$:

- $\mathbf{l^{(j)}} = \mathbf{l^{(j)}}(x) = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}}x \in \mathbb{Z}_p^m$,
- $\mathbf{r^{(j)}} = \mathbf{r^{(j)}}(x) = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2 \cdot \mathbf{z}^m \in \mathbb{Z}_p^m$,
- $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >\in \mathbb{Z}_p$,
- $\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{j=1}^m z^{1+j}\gamma_j \in \mathbb{Z}_p$,

- $\mu = \alpha + \rho x \in \mathbb{Z}_p$.

- step 10 : $\mathcal{P}$ sends to $\mathcal{V}$: τ_x, μ.
- step 11 : $\mathcal{V}$ picks ϕ uniformly at random in $\mathbb{Z}_p^*$.
- step 12 : $\mathcal{V}$ sends ϕ to $\mathcal{P}$.
- step 13 : $\mathcal{P}$ computes $\overline{t} = \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$.
- step 14 : $\mathcal{P}$ sends $\overline{t}$ to $\mathcal{V}$.
- step 15 : $\mathcal{P}$ and $\mathcal{V}$ compute:

- $\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$, $i \in \{1, \dots, m\}$, $\mathbf{h'_j} = (\mathbf{h'_j}[1], \dots, \mathbf{h'_j}[m])$, $j \in [1, l]$.
- $\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$, $i \in \{1, \dots, m\}$, $\mathbf{g'_j} = (\mathbf{g'_j}[1], \dots, \mathbf{g'_j}[m])$, $j \in [1, l]$.
- $P = AS^x (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$.
- $\overline{P} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} (h_1, \dots, h_l)^{(\phi^0, \dots, \phi^{l-1})}$

- step 16 : $\mathcal{P}$ and $\mathcal{V}$ engage in two "Bulletproofs" inner-product arguments for the following relation :

$$R_5 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c =< \mathbf{a}, \mathbf{b} >\}$$

, first on input :

- $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $\mathbf{h} := (\mathbf{h'_1} || \dots || \mathbf{h'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $P := P h^{-\mu} \in \mathbb{G}$,
- $c := \overline{t} \in \mathbb{Z}_p$,
- $\mathbf{a} := (\mathbf{l^{(1)}} \phi^0 || \dots || \mathbf{l^{(l)}} \phi^{l-1}) \in \mathbb{Z}_p^{(m \cdot l)}$,
- $\mathbf{b} := (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) \in \mathbb{Z}_p^{(m \cdot l)}$,

for $n = (m \cdot l)$,
and then on input :

- $\mathbf{g} := (g_1, \dots, g_l) \in \mathbb{G}^l$,
- $\mathbf{h} := (h_1, \dots, h_l) \in \mathbb{G}^l$,
- $P := \overline{P} \in \mathbb{G}$,
- $c := \overline{t} \in \mathbb{Z}_p$,
- $\mathbf{a} := (\hat{t}^{(1)}, \dots, \hat{t}^{(l)}) \in \mathbb{Z}_p^l$,
- $\mathbf{b} := (\phi^0, \dots, \phi^{l-1}) \in \mathbb{Z}_p^l$,

, for $n = l$.
- step 17 : $\mathcal{P}$ and $\mathcal{V}$ engage in a "Schnorr-Extended Argument" for the following relation :

$$R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a}, \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\}$$

, on input :

- $\mathbf{g} := (g_1, \dots, g_l) \in \mathbb{G}^{(l)}$,
- $P := h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} \in \mathbb{G}$,
- $\mathbf{a} := (\hat{t}^{(1)}, \dots, \hat{t}^{(l)}) \in \mathbb{Z}_p^l$,

for $n = l$.
- step 18 : $\mathcal{V}$ accepts the proof if the two "Bulletproofs" inner-product arguments and the "Extended-Schnorr Argument" are valid.

0-1 Protocol (first part)

Prover		**Verifier**
$\mathbf{a_L^{(j)}} = (v_j^{(1)}, \ldots, v_j^{(m)})$		
$\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$		
$\mathbf{s_L^{(j)}}, \mathbf{s_R^{(j)}} \xleftarrow{\$} \mathbb{Z}_p^m$		
$A = h^\alpha(\mathbf{g_1^{a_L^{(1)}}} \ldots \mathbf{g_l^{a_L^{(l)}}})(\mathbf{h_1^{a_R^{(1)}}} \ldots \mathbf{h_l^{a_R^{(l)}}})$		
$S = h^\alpha(\mathbf{g_1^{s_L^{(1)}}} \ldots \mathbf{g_l^{s_L^{(l)}}})(\mathbf{h_1^{s_R^{(1)}}} \ldots \mathbf{h_l^{s_R^{(l)}}})$		
	$\xrightarrow{A, S}$	
		$y, z \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{y, z}$	
$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p^*$		
$T_1 = h^{\tau_1} g_1^{t_1^{(1)}} \ldots g_l^{t_1^{(l)}}$		
$T_2 = h^{\tau_2} g_1^{t_2^{(1)}} \ldots g_l^{t_2^{(l)}}$		
	$\xrightarrow{T_1, T_2}$	
		$x \xleftarrow{\$} \mathbb{Z}_p^*$
	$\xleftarrow{x}$	
$\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} x$		
$\mathbf{r^{(j)}} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m$		
$\hat{t}^{(j)} = < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$		
$\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{j=1}^{m} z^{1+j} \gamma_j$		
$\mu = \alpha + \rho x$		
	$\xrightarrow{\tau_x, \mu}$	
		$\phi \xleftarrow{\$} \mathbb{Z}_p^*$
	$\xleftarrow{\phi}$	
$\bar{t} := \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$		
	$\xrightarrow{\bar{t}}$	

0-1 Protocol (part 2)

Prover		**Verifier**
$\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$		$\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$
$\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$		$\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$
$P = AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} \cdot$ $(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$		$P = AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} \cdot$ $(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$
$\mathbf{g} := (\mathbf{g'_1} \| \dots \| \mathbf{g'_l})$		$\mathbf{g} := (\mathbf{g'_1} \| \dots \| \mathbf{g'_l})$
$\mathbf{h} := (\mathbf{h'_1} \| \dots \| \mathbf{h'_l})$		$\mathbf{h} := (\mathbf{h'_1} \| \dots \| \mathbf{h'_l})$
$\overline{P} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} \cdot$ $T_1^x T_2^{x^2} (h_1, \dots, h_l)^{(\phi^0, \dots, \phi^{l-1})}$		$\overline{P} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} \cdot$ $T_1^x T_2^{x^2} (h_1, \dots, h_l)^{(\phi^0, \dots, \phi^{l-1})}$
$\mathbf{a} := (\mathbf{l^{(1)}} \phi^0 \| \dots \| \mathbf{l^{(l)}} \phi^{l-1})$		
$\mathbf{b} := (\mathbf{r^{(1)}} \| \dots \| \mathbf{r^{(l)}})$		
	$BP(\mathbf{g}, \mathbf{h}, P, \bar{t}; \mathbf{a}, \mathbf{b})$ $\longleftrightarrow$	
$\overline{\mathbf{g}} := (g_1, \dots, g_l)$		$\overline{\mathbf{g}} := (g_1, \dots, g_l)$
$\overline{\mathbf{h}} := (h_1, \dots, h_l)$		$\overline{\mathbf{h}} := (h_1, \dots, h_l)$
$\overline{\mathbf{a}} := (\hat{t}^{(1)}, \dots, \hat{t}^{(l)})$		
$\overline{\mathbf{b}} := (\phi^0, \dots, \phi^{l-1})$		
	$BP(\overline{\mathbf{g}}, \overline{\mathbf{h}}, \overline{P}, \bar{t}; \overline{\mathbf{a}}, \overline{\mathbf{b}})$ $\longleftrightarrow$	
$\tilde{P} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} \cdot$ $T_1^x T_2^{x^2}$		$\tilde{P} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} \cdot$ $T_1^x T_2^{x^2}$
	$ES(\overline{\mathbf{g}}, \tilde{P}; \overline{\mathbf{a}})$ $\longleftrightarrow$	

8.3 Complexity

The Logarithmic 0-1 Protocol has the following complexities:

- **# Communications :** $2log_2(ml) + 4log_2(l) + 12$ elements :
  - $2log_2(ml) + 4log_2(l) + 4$ **Group elements :**
    - A, S, T_1, T_2 : 4 elements in $\mathbb{G}$
    - "Bulletproofs" inner-product argument with $n = ml$: $2log_2(ml)$ elements in $\mathbb{G}$
    - "Bulletproofs" inner-product argument with $n = l$: $2log_2(l)$ elements in $\mathbb{G}$
    - "Extended-Schnorr" argument with $n = l$: $2log_2(l)$ elements in $\mathbb{G}$
  - 8 $\mathbb{Z}_p$ **elements :**
    - $\tau_x, \mu, \bar{t}$: 3 elements in $\mathbb{Z}_p$
    - "Bulletproofs" inner-product argument with $n = lm$: 2 elements in $\mathbb{Z}_p$
    - "Bulletproofs" inner-product argument with $n = l$: 2 elements in $\mathbb{Z}_p$
    - "Exetended Schnorr" argument with $n = l$: 1 element in $\mathbb{Z}_p$
- **# Bases :** $2ml + 2l + 2$ bases:
  - $\mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{h_1}, \dots, \mathbf{h_l}$: $2ml$ elements in $\mathbb{G}$

  - $g_1, \ldots, g_l, h_1, \ldots, h_l$: $2l$ elements in $\mathbb{G}$
  - h, u : 2 elements in $\mathbb{G}$

- **# Exponentiations in $\mathbb{G}$:**
  - **Prover :** $16lm + 14l + m + 4log_2(lm) + 6log_2(l) - 11$ exponentiations in $\mathbb{G}$:
  - **Verifier :** $8lm + 7l + m + 2log_2(lm) + 4log_2(l) + 2$ exponentiations in $\mathbb{G}$

  The details are shown in Table 8.1.

- **# Exponentiations in $\mathbb{Z}_p$:**
  - **Prover:** $3log_2(lm) + 6log_2(l) + 9$ exponentiations in $\mathbb{Z}_p$
  - **Verifier:** $3log_2(lm) + 6log_2(l) + 9$ exponentiations in $\mathbb{Z}_p$

  Details are shown in Table 8.1.

- **# Multiplications in $\mathbb{G}$:**
  - **Prover :** $10lm + 11l + m + 4log_2(lm) + 2log_2(l) - 8$ multiplications:
    * $2ml$ for each of A, S : $4ml$ multiplications
    * l for each of T_1, T_2 : $2l$ multiplications
    * P: $2ml + 1$ multiplications
    * $\overline{P}$:$m + 2l + 2$ multiplications
    * "Bulletproofs" inner-product argument with $n = ml$: $4ml + 2log_2(ml) - 4$ multiplications
    * "Bulletproofs" inner-product argument with $n = l$: $4l + 2log_2(l) - 4$ multiplications
    * "Extended-Schnorr" argument with $n = l$:$3l - 3$ multiplications
  - **Verifier :** $4lm + 5l + m + 2log_2(l) + 2$ multiplications
    * P: $2ml + 1$
    * $\overline{P}$: $m + 2l + 2$
    * "Bulletproofs" inner-product argument with $n = ml$: $2ml$
    * "Bulletproofs" inner-product argument with $n = l$: $2l$
    * "Extended-Schnorr" argument with $n = l$: $l + 2log_2(l) - 1$

- **# Multiplications in $\mathbb{Z}_p$:**
  - **Prover :** $12ml + 9l - 16$ multiplications:
    * $\{t_1^{(j)}\}_{j=1}^{l}$: $4ml$ multiplications
    * $\{t_2^{(j)}\}_{j=1}^{l}$: $2ml$ multiplications
    * $\{z^{1+j}, y^{j-1}\}_{j=1}^{m}$: $2(m-1)$ multiplications
    * $\{\phi^{j-1}\}_{j=1}^{l}$: $l - 1$ multiplications
    * "Bulletproofs inner product" argument with $n = ml$: $6(ml - 1)$ multiplications
    * "Bulletproofs inner product" argument with $n = l$: $6(l - 1)$ multiplications
    * "Extended-Schnorr" argument: $2l - 2$ multiplications
  - **Verifier :** $2ml + 3m + l - 1$ multiplications
    * $\{y^{j-1}\}_{j=1}^{m}$: $m - 2$ multiplications
    * $\{\phi^{j-1}\}_{j=1}^{l}$: $l - 1$ multiplications
    * $\mathbf{h}'$: $2ml$ multiplications
    * P : m multiplications
    * $\{z^{1+j}\}_{j=1}^{m}$: m multiplications
    * "Bulletproofs" inner-product argument with $n = ml$: 1 multiplication
    * "Bulletproofs" inner-product argument with $n = l$: 1 multiplication

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
A	2lm+1			
S	2lm+1			
$\{z^{1+j}\}_{j=1}^{m}$		1		1
T_1	l+1			
T_2	l+1			
x^2		1		1
y^{-1}		1		1
ϕ		1		1
δ		1		1
$\overline{P}$	l+m+3		l+m+3	
P	2lm+2		2lm+2	
τ_x		l		
h'	lm		lm	
g'	lm		lm	
P_{ext}	l		l	
BP(lm)	$8lm + 4log_2(lm) - 8$	$3log_2(lm) + 2$	$4lm + 2log_2(lm) - 1$	$3log_2(lm) + 2$
BP(l)	$8l + 4log_2(l) - 8$	$3log_2(l) + 2$	$4l + 2log_2(l) - 1$	$3log_2(l) + 2$
ES(l)	$4l + 2log_2(l) - 4$	$3log_2(l)$	$2l + 2log_2(l) - 1$	$3log_2(l)$
TOTAL	$16lm + 16l + m + 4log_2(lm) + 6log_2(l) - 11$	$3log_2(lm)+6log_2(l)+9$	$8lm + 7l + m + 2log_2(lm)+4log_2(l)+2$	$3log_2(lm)+6log_2(l)+9$

Table 8.1: Complexity of Logarithmic 0-1 Protocol

As desired the number of exponentiation and multiplications is linear in lm and the size of the elements exchanged scales logarithmically in lm.

8.4 Theorem 5

The zero-knowledge protocol presented in section 8B for relation R_1 has perfect completeness, perfect special honest-verifier zero-knowledge and computational witness extended emulation.

In the followings subsections, we will show point by point that the 0-1 protocol satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* for relation R_1.

8.4.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $\{\gamma_j\}_{j=1}^{m}, \{v_i^{(j)}\}_{i=1,j=1}^{l,m}$ for the relation $R_1 \equiv \{(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m;$
$(v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}_p^m) : V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\},$
$\forall i \in [1,l], j \in [1,m]\}$.
We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \in R_1\right] = 1$.

To see this, let's look at the verification equations that the verifier $\mathcal{V}$ checks in the protocol :

$$\mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$$
$$= (h^{\gamma_1} g_1^{v_1^{(1)}} \dots g_l^{v_l^{(1)}}, \dots, h^{\gamma_m} g_1^{v_1^{(m)}} \dots g_l^{v_l^{(m)}})^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} (h^{\tau_1} (g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}}))^x (h^{\tau_2} (g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}}))^{x^2}$$
$$= (h^{\gamma_1 z^2} g_1^{v_1^{(1)} z^2} \dots g_l^{v_l^{(1)} z^2} \dots h^{\gamma_m z^{m+2}} g_1^{v_1^{(m)} z^{m+2}} \dots g_l^{v_l^{(m) z^{m+2}}}) g_1^{\delta(y,z)} \dots g_l^{\delta(y,z)} h^{\tau_1 x} g_1^{t_1^{(1)} x} \dots g_l^{t_1^{(l)} x} h^{\tau_2 x^2} g_1^{t_2^{(1)} x^2} \dots g_l^{t_2^{(l)} x^2}$$
$$= g_1^{\sum_{k=1}^{m} v_1^{(k)} z^{k+1} + \delta(y,z) + t_1^{(1)} x + t_2^{(1)} x^2} \dots g_l^{\sum_{k=1}^{m} v_l^{(k)} z^{k+1} + \delta(y,z) + t_1^{(l)} x + t_2^{(l)} x^2} h^{\tau_2 x^2 + \tau_1 x + \sum_{k=1}^{m} \gamma_k z^{k+1}}$$
$$= g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}} h^{\tau_x},$$

since $\mathcal{P}$ has followed the protocol and thus has computed :

$\hat{t}^{(j)} = \sum_{k=1}^{m} v_j^{(k)} z^{k+1} + \delta(y,z) + t_1^{(j)} x + t_2^{(j)} x^2$, $j \in \{1,\dots,l\}$ and $\tau_x = \tau_2 x^2 + \tau_1 x + \sum_{k=1}^{m} \gamma_k z^{k+1}$.

Hence, since the "Extend-Schnorr Argument" is complete and $\mathcal{P}$ runs it honestly knowing the witness $(\hat{t}^{(1)},\dots,\hat{t}^{(1)})$, $\mathcal{V}$ will thus accept the "Extended-Schnorr Argument" for relation R_4 on input:

- $\mathbf{g} := (g_1,\dots,g_l) \in \mathbb{G}^l$,
- $P := \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$,
- $\mathbf{a} := (\hat{t}^{(1)},\dots,\hat{t}^{(l)})$.

Multiplying both sides of the previous equation by $h_1^{\phi^0} \cdots h_l^{\phi^{l-1}}$ and putting the term $h^{-\tau_x}$ on the other side we get :

$$h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} h_1^{\phi^0} \cdots h_l^{\phi^{l-1}} = \overline{P} = (g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}}) h_1^{\phi^0} \cdots h_l^{\phi^{l-1}}.$$

Finally, $\mathcal{P}$ computed $\bar{t}$ as : $\bar{t} = \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1} =< (\hat{t}^{(1)},\dots,\hat{t}^{(l)}), (\phi^0,\dots,\phi^{l-1}) >$.
Hence, since the "Bulletproofs" argument is complete and $\mathcal{P}$ runs it honestly knowing the witness, $\mathcal{V}$ will thus accept the "Bulletproofs" inner-product argument for the relation R_5 (8.2) on inputs:

- $\mathbf{g} := (g_1,\dots,g_l) \in \mathbb{G}^l$,
- $\mathbf{h} := (h_1,\dots,h_l) \in \mathbb{G}^l$,
- $P := \overline{P} \in \mathbb{G}$,
- $c := \bar{t}$,
- $\mathbf{a} := (\hat{t}^{(1)},\dots,\hat{t}^{(l)})$,
- $\mathbf{b} := (\phi^0,\dots,\phi^{l-1})$,

, for $n = l$.

We also have :

$$\begin{aligned}
P &= AS^x (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= \left(h^\alpha (\mathbf{g_1}^{\mathbf{a_L^{(1)}}} \dots \mathbf{g_l}^{\mathbf{a_L^{(l)}}}) (\mathbf{h_1}^{\mathbf{a_R^{(1)}}} \dots \mathbf{h_l}^{\mathbf{a_R^{(l)}}}) \right) \left(h^\rho (\mathbf{g_1}^{\mathbf{s_L^{(1)}}} \dots \mathbf{g_l}^{\mathbf{s_L^{(l)}}}) (\mathbf{h_1}^{\mathbf{s_R^{(1)}}} \dots \mathbf{h_l}^{\mathbf{s_R^{(l)}}}) \right)^x \\
&\quad \mathbf{g'_1}^{-z\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{-z\phi^{l-1} \cdot \mathbf{1}^m} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= \left(h^\alpha (\mathbf{g'_1}^{\phi^0 \mathbf{a_L^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \mathbf{a_L^{(l)}}}) (\mathbf{h'_1}^{\mathbf{y}^m \circ \mathbf{a_R^{(1)}}} \dots \mathbf{h'_l}^{\mathbf{y}^m \circ \mathbf{a_R^{(l)}}}) \right) \left(h^\rho (\mathbf{g'_1}^{\phi^0 \mathbf{s_L^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \mathbf{s_L^{(l)}}}) (\mathbf{h'_1}^{\mathbf{y}^m \circ \mathbf{s_R^{(1)}}} \dots \mathbf{h'_l}^{\mathbf{y}^m \circ \mathbf{s_R^{(l)}}}) \right)^x \\
&\quad \mathbf{g'_1}^{-z\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{-z\phi^{l-1} \cdot \mathbf{1}^m} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} \\
&= h^{(\alpha + \rho x)} \mathbf{g'_1}^{\phi^0 (\mathbf{a_L^{(1)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(1)}} x)} \dots \mathbf{g'_l}^{\phi^{l-1} (\mathbf{a_L^{(l)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(l)}} x)} \mathbf{h'_1}^{[\mathbf{y}^m \circ (\mathbf{a_R^{(1)}} + \mathbf{s_R^{(1)}} x + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m]} \dots \\
&\quad \mathbf{h'_l}^{[\mathbf{y}^m \circ (\mathbf{a_R^{(l)}} + \mathbf{s_R^{(l)}} x + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m]} \\
&= h^\mu (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{l^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{l^{(l)}}}) ((\mathbf{h'_1})^{\mathbf{r^{(1)}}} \dots (\mathbf{h'_l})^{\mathbf{r^{(1)}}})
\end{aligned}$$

Where the last equality follows by :

$$\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z\mathbf{1}^m + \mathbf{s_L^{(j)}} x, j \in \{1,\dots,l\}$$

and

$$\mathbf{r^{(j)}} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m, j \in \{1,\dots,l\}.$$

Moreover, the prover $\mathcal{P}$ computed $\hat{t}^{(j)}$ as $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$ and thus :
$< (\mathbf{l^{(j)}} \phi^{j-1})_{j=1}^l, (\mathbf{r^{(j)}})_{j=1}^l >= \sum_{j=1}^l < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \phi^{j-1} = \sum_{j=1}^l \hat{t}^{(j)} \phi^{j-1}$
Hence, since the "Bulletproofs" argument is complete and $\mathcal{P}$ runs it honestly knowing the witness, $\mathcal{V}$ will accept the "Bulletproofs" inner-product argument for relation R_5 (8.2) on input:

- $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$,

- $\mathbf{h} := (\mathbf{h'_1}||\dots||\mathbf{h'_l}) \in \mathbb{G}^{(m\cdot l)}$,
- $P := Ph^{-\mu} \in \mathbb{G}$,
- $c := \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1} \in \mathbb{Z}_p$,
- $\mathbf{a} := (\mathbf{l^{(1)}}\phi^0||\dots||\mathbf{l^{(l)}}\phi^{l-1}) \in \mathbb{Z}_p^{(m\cdot l)}$,
- $\mathbf{b} := (\mathbf{r^{(1)}}||\dots||\mathbf{r^{(l)}}) \in \mathbb{Z}_p^{(m\cdot l)}$,

for $n = (m \cdot l)$,
Together with the first "Bulletproofs" inner-product argument being accepted by the verifier, $\mathcal{V}$ will accept the proof thereby showing the *Perfect Completeness* of the protocol.

8.4.2 Perfect Special Honest-Verifier Zero-Knowledge

To show *Perfect Special Honest-Verifier Zero-Knowledge* (SHVZK) we build explicitly an efficient simulator that given the C.R.S. and a statement ($g_1,\dots,g_l,h \in \mathbb{G}$ and $\mathbf{V} = (V_1,\dots,V_m) \in \mathbb{G}$ respectively) produces indistinguishable transcript (in the sense that the transcript will have identical distribution) from a transcript resulting from the true interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. That is :

$$P\left[\mathcal{A}(tr)=1 \;\middle|\; \begin{array}{l}\sigma \leftarrow Setup(1^\lambda),\\ (u,w,\rho)\leftarrow \mathcal{A}(\sigma),\\ tr \leftarrow \langle \mathcal{P}(\sigma,u,w),\mathcal{V}(\sigma,u,\rho)\rangle\end{array}\right] = P\left[\mathcal{A}(tr)=1 \;\middle|\; \begin{array}{l}\sigma \leftarrow Setup(1^\lambda),\\ (u,w,\rho)\leftarrow \mathcal{A}(\sigma),\\ tr \leftarrow \mathcal{S}(\sigma,u,\rho)\end{array}\right].$$

The simulator pseudo-code is as follows :

0-1 Simulator

- Input : $(g_1,\dots,g_l,h,u,\mathbf{g_1},\dots,\mathbf{g_l},\mathbf{h_1},\dots,\mathbf{h_l},\mathbf{V})$.
- Output : transcript identically distributed to a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

- step 1 : Picks challenges x,y,z,ϕ uniformly at random in $\mathbb{Z}_p^*$.
- step 2 : Compute $\mathbf{h'_j}[i] = (\mathbf{h_j}[i])^{y^{-i+1}}$, $i \in \{1,\dots,m\}$, $\mathbf{h'_j} = (\mathbf{h'_j}[1],\dots,\mathbf{h'_j}[m])$, $j \in [1,l]$.
- step 3 : Compute $\mathbf{g'_j}[i] = (\mathbf{g_j}[i])^{\phi^{-j+1}}$, $i \in \{1,\dots,m\}$, $\mathbf{g'_j} = (\mathbf{g'_j}[1],\dots,\mathbf{g'_j}[m])$, $j \in [1,l]$.
- step 4 : Picks μ,τ_x uniformly at random in $\mathbb{Z}_p^*$.
- step 5 : Picks $\mathbf{l^{(j)}},\mathbf{r^{(j)}}$ uniformly at random in $\mathbb{Z}_p^m$, $j \in \{1,\dots,l\}$.
- step 6 : Computes $\hat{t}^{(j)} =< \mathbf{l^{(j)}},\mathbf{r^{(j)}} >$, $j \in \{1,\dots,l\}$.
- step 7 : Compute $\bar{t} = \sum_{j=1}^{l} \hat{t}^{(j)}\phi^{j-1}$.
- step 8 : Picks A uniformly at random in $\mathbb{G}$.
- step 9 : Computes

$$S = \Big(h^{-\mu}(\mathbf{g'_1}^{-\phi^0\mathbf{l^{(1)}}}\dots\mathbf{g'_l}^{-\phi^{l-1}\mathbf{l^{(l)}}})(\mathbf{h'_1}^{-\mathbf{r^{(1)}}}\dots\mathbf{h'_l}^{-\mathbf{r^{(l)}}})A$$
$$(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}\,(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\Big)^{-x^{-1}}$$

- step 10 : Picks T_2 uniformly at random in $\mathbb{G}$.
- step 11 : Computes $T_1 = \Big(h^{-\tau_x}\mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_2^{x^2}(g_1^{-\hat{t}^{(1)}}\dots g_l^{-\hat{t}^{(l)}})\Big)^{-x^{-1}}$.
- step 12 : Compute $P = AS^x(\mathbf{g'_1}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g'_l}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h'_1}\dots\mathbf{h'_l})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$.
- step 13 : Compute $\overline{P} = h^{-\tau_x}\mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_1^xT_2^{x^2}$.
- step 14 : Compute $\tilde{P} = h^{-\tau_x}\mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1\dots g_l)^{\delta(y,z)}T_1^xT_2^{x^2}(h_1,\dots,h_l)^{(\phi^0,\dots,\phi^{l-1})}$.
- step 15 : Given the witness $\mathbf{a} := (\phi^{j-1}\mathbf{l^{(j)}})_{j=1}^l$ and $\mathbf{b} := (\mathbf{r^{(j)}})_{j=1}^l$ run the protocol 2 on input $((\mathbf{g'_1}||\dots||\mathbf{g'_l}),(\mathbf{h'_1}||\dots||\mathbf{h'_l}),Ph^{-\mu},\bar{t};\mathbf{a};\mathbf{b})$ to get transcript S_1^{inner} of the first inner-product argument.
- step 16 : Given the witness $\overline{\mathbf{a}} := (\hat{t}^{(j)})_{j=1}^l$ and $\overline{\mathbf{b}} := (\phi^j)_{j=1}^l$ run the protocol 2 (6.4) on input $((g_1,\dots,g_l),(h_1,\dots,h_l),\overline{P},\bar{t};\overline{a},\overline{\mathbf{b}})$ to get the transcript S_2^{inner} of the second inner-product argument.
- step 17 : Given the witness $\overline{\mathbf{a}} = (\hat{t}^{(j)})_{j=1}^l$ run the protocol 1 on input $((g_1,\dots,g_l),\tilde{P};\overline{\mathbf{a}})$ to get transcript S^{ES} of the "Extended-Schnorr Argument".
- step 18 : Output : $(A,S;y,z;T_1,T_2;x;\tau_x,\mu;\phi;\bar{t};S_1^{inner};S_2^{inner};S^{ES})$.

An honest prover interacting with an honest-verifier will produce independent random values $A, \mathbf{l^{(j)}}, \mathbf{r^{(j)}}, \mu, \tau_x$ given $\alpha, \rho, \tau_1, \tau_2, x, y, z, \mathbf{s_L^{(j)}}, \mathbf{s_R^{(j)}}$ are chosen independently and randomly. They will produce $\bar{t} = \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$ as : $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$, $j \in \{1, \dots, l\}$.
The simulated S is fully defined by :
$AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m} = h^{\mu}(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{l^{(1)}}} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{l^{(l)}}})((\mathbf{h'_1})^{\mathbf{r^{(1)}}} \dots (\mathbf{h'_l})^{\mathbf{r^{(l)}}})$, which is ensured by computing S accordingly. Given that $\mathcal{P}$ follows the protocol honestly, T_1, T_2 are perfectly hiding multi-Pedersen commitments and thus are random group elements. Given $\hat{t}$ and answer τ_x, the internal relation between T_1 and T_2 is fully defined by the equation $(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}}) h^{\tau_x} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$, which is ensured by computing T_1 accordingly.

S_1^{inner} is ran knowing the true witness for this sub-protocol, $(\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{l)}})$ such that : $\bar{t} =< (\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{l)}}) >$ and $AS^x(\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z}(\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}$
$= (\mathbf{g_1}^{-\phi^0} || \dots || \mathbf{g_l}^{\phi^{l-1}})^{(\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1})} (\mathbf{h'_1} || \dots || \mathbf{h'_l})^{(\mathbf{r^{(1)}} || \dots || \mathbf{r^{l)}})}$ so the transcript produced for the first "Bulletproofs" inner product argument inside the protocol will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

S_2^{inner} is ran knowing the true witness $(\hat{t}^{(1)}, \dots, \hat{t}^{(l)}), (\phi^0, \dots, \phi^{l-1})$ such that :
$\bar{t} =< ((\hat{t}^{(1)}, \dots, \hat{t}^{(l)}), (\phi^0, \dots, \phi^{l-1}) >$ and $h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} (h_1, \dots, h_l)^{(\phi^0, \dots, \phi^{l-1})}$
$= (g_1, \dots, g_l)^{(\hat{t}^{(1)}, \dots, \hat{t}^{(l)})} (h_1, \dots, h_l)^{(\phi^0, \dots, \phi^{l-1})}$ so the transcript produced for the second "Bulletproofs" inner product argument will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

S^{ES} is ran knowing the true witness $(\hat{t}^{(1)}, \dots, \hat{t}^{(l)})$ such that :
$h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} = (g_1, \dots, g_l)^{(\hat{t}^{(1)}, \dots, \hat{t}^{(l)})}$ so the transcript produced for the "Extended-Schnorr" argument will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

We can thus conclude that the transcript obtained by the simulator pseudo-code is distributed identically to the transcript of a true protocol interaction between honest $\mathcal{P}$ and $\mathcal{V}$ with independently uniformly selected challenges. This shows our protocol proof has the *Perfect Special Honest-Verifier Zero-Knowledge*.

8.4.3 Computational Witness Extended-Emulation

In this section, in order to prove the *Computational Witness Extended-Emulation* property, we construct an efficient extractor χ that uses χ^{BP}, the extractor of the "Bulletproofs" inner-product argument as a subroutine and a polynomial number of $N^{0-1} := 3 \cdot m \cdot (m+2) \cdot l \cdot N^{BP}(l \cdot m) \cdot N^{BP}(l) \cdot N^{ES}(l) = O(m^4 l^{6.58})$ transcripts with 3 different values of the challenge x : x_1, x_2, x_3, m different values of the challenge y : $\{y_j\}_{j=1}^m$, $m+2$ different values of the challenge z : $\{z_j\}_{j=1}^{m+2}$, l different values of the challenge ϕ : $\{\phi_i\}_{i=1}^l$, $(ml)^2$ for the first "Bulletproofs" inner product argument, l^2 for the second one and $\approx l^{1.58}$ to extract a valid witness for relation $R_1 \equiv \{(g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \dots, \mathbf{g_l}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m; (v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}_p^m) : V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0,1\}, \forall i \in [1,l], j \in [1,m]\}$.

This allows to prove, using the Forking Lemma (3.4), the *Computational Witness Extended Emulation* (3.3.4) property which is a larger notion of security than soundness.

We will use the *Discrete Log Relation* assumption, which was presented in section 3.1.

The following proof may seem a bit long and intricate but it is not very complicated. All that is required is to build the extractor χ.
Informally the extractor will work as follows: χ first uses the "Bulletproofs" inner product argument extractor, χ^{BP} to extract the witnesses $(\hat{t}^{(1)}, \dots, \hat{t}^{(l)})$ and $(\phi^0, \dots, \phi^{l-1})$ of the first "Bulletproofs" inner product argument. It also use it to extract the witnesses $(\phi^0 \mathbf{l^{(1)}} || \dots || \phi^{l-1} \mathbf{l^{(l)}})$ and $(\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}})$ of the second "Bulletproofs" inner product argument.
It then uses the "Extended-Schnorr" argument extractor, χ^{ES} to extract the witness $(\hat{t}^{(1)}, \dots, \hat{t}^{(l)})$ and we show that this witness must be the same that the one extracted with χ^{BP}.
Then we combine different transcripts to open the commitments S, A, T_1, T_2 and finally $\{V_j\}_{j=1}^m$, which leads to a candidate witness for relation R_1. We then use the verification equations and the fact that $< (\phi^0 \mathbf{l^{(1)}} || \dots || \phi^{l-1} \mathbf{l^{(l)}}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) >= \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}$ to show that the extracted witness satisfy the relation R_1.

Our extractor χ first runs the extractor of the "Bulletproofs" inner-product argument, χ^{BP}, on each of the transcripts for the second "Bulletproofs" inner-product argument for relation R_5 (8.2) on inputs:

- $\mathbf{g} := (g_1, \dots, g_l) \in \mathbb{G}^l$,
- $\mathbf{h} := (h_1, \dots, h_l) \in \mathbb{G}^l$,
- $P := \overline{P} \in \mathbb{G}$,
- $c := \bar{t}$,
- $\mathbf{a} := (\hat{t}^{(1)}, \dots, \hat{t}^{(l)})$,
- $\mathbf{b} := (\phi^0, \dots, \phi^{l-1})$,

, for $n = l$.

Our extractor will obtain in this way the witness $(\hat{\mathbf{t}}, \psi) = ((\hat{t}^{(1)}, \dots, \hat{t}^{(l)}), (\psi^{(1)}, \dots, \psi^{(l)}))$ such that
$g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}} h_1^{\psi^{(1)}} \dots h_l^{\psi^{(l)}} = \overline{P} := h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} h_1^{\phi^0} \dots h_l^{\phi^{l-1}}$ and
$< (\hat{t}^{(1)}, \dots, \hat{t}^{(l)}), (\psi^{(1)}, \dots, \psi^{(l)}) >= \bar{t}$.

Our extractor will then run the extractor of the "Extended-Schnorr" argument, χ^{ES}, to extract the witnesses $\tilde{\mathbf{t}} = (\tilde{t}^{(1)}, \dots, \tilde{t}^{(l)})$, such that: $g_1^{\tilde{t}^{(1)}} \dots g_l^{\tilde{t}^{(l)}} = h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$.
Plugging this into the previous equation leads to:
$g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}} h_1^{\psi^{(1)}} \dots h_l^{\psi^{(l)}} = \overline{P} := h^{-\tau_x} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} h_1^{\phi^0} \dots h_l^{\phi^{l-1}} = g_1^{\tilde{t}^{(1)}} \dots g_l^{\tilde{t}^{(l)}} h_1^{\phi^0} \dots h_l^{\phi^{l-1}}$.

Using the *Discrete Log Relation* assumption we infer $\psi = (\psi^{(1)}, \dots, \psi^{(l)}) = (\phi^0, \dots, \phi^{l-1})$ and $\tilde{\mathbf{t}} = \hat{\mathbf{t}}$.
This implies $g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}} h^{\tau_x} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$.
Finally we must also have that $\bar{t}$ provided by $\mathcal{P}$ in step **- step 14 :** of the protocol is :
$\bar{t} =< (\tilde{t}^{(1)}, \dots, \tilde{t}^{(l)}), (\psi^{(1)}, \dots, \psi^{(l)}) >= \sum_{j=1}^{l} \tilde{t}^{(j)} \cdot \psi^{(j)} = \sum_{j=1}^{l} \hat{t}^{(j)} \cdot \phi^{j-1}$.

The proof now will be essentially the same that the one for the first protocol presented in the previous section. The reader can skip the rest of the proof if he already read the *Computational Witness Extended-Emulation* proof from the previous section.

Our extractor χ will run the extractor of the "Bulletproofs" inner-product argument on each of the transcripts produced by χ^{BP} for the second "Bulletproofs" inner-product argument for relation R_5 (8.2) on inputs:

- $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $\mathbf{h} := (\mathbf{h'_1} || \dots || \mathbf{h'_l}) \in \mathbb{G}^{(m \cdot l)}$,
- $P := Ph^{-\mu} \in \mathbb{G}$,
- $c := \bar{t} (= \sum_{j=1}^{l} \hat{t}^{(j)} \phi^{j-1}) \in \mathbb{Z}_p$,
- $\mathbf{a} := (\mathbf{l^{(1)}} \phi^0 || \dots || \mathbf{l^{(l)}} \phi^{l-1}) \in \mathbb{Z}_p^{(m \cdot l)}$,
- $\mathbf{b} := (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) \in \mathbb{Z}_p^{(m \cdot l)}$,

for $n = (m \cdot l)$.
In this way, it will obtain the witnesses $(\tilde{\mathbf{l}}, \tilde{\mathbf{r}}) = \left((\tilde{\mathbf{l}}^{(\mathbf{1})} || \dots || \tilde{\mathbf{l}}^{(\mathbf{l})}), (\tilde{\mathbf{r}}^{(\mathbf{1})} || \dots || \tilde{\mathbf{r}}^{(\mathbf{l})})\right) \in \mathbb{Z}_p^{(m \cdot l)} \times \mathbb{Z}_p^{(m \cdot l)}$ such that : $\mathbf{g}^{\tilde{l}} \mathbf{h}^{\tilde{r}} = Ph^{-\mu} \wedge < \tilde{\mathbf{l}}, \tilde{\mathbf{r}} >= \bar{t}$, where $\mathbf{g} := (\mathbf{g'_1} || \dots || \mathbf{g'_l}) \in \mathbb{G}^{(m \cdot l)}$ and $\mathbf{h} := (\mathbf{h'_1} || \dots || \mathbf{h'_l}) \in \mathbb{G}^{(m \cdot l)}$.

Now, consider two valid transcripts with different values of the challenge x : $x_1, x_2 \in \mathbb{Z}_p^*$ and their inner-product argument witnesses $\tilde{\mathbf{l}}_\mathbf{1}, \tilde{\mathbf{r}}_\mathbf{1}, \tilde{\mathbf{l}}_\mathbf{2}, \tilde{\mathbf{r}}_\mathbf{2}$. We have : $\mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}} = \overline{P}_i$, which is equivalent to :
$\mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}} h^{\mu_i} = AS^{x_i} (\mathbf{g'_1}^{\phi^0 \cdot \mathbf{1}^m} \dots \mathbf{g'_l}^{\phi^{l-1} \cdot \mathbf{1}^m})^{-z} (\mathbf{h'_1} \dots \mathbf{h'_l})^{z \cdot \mathbf{y}^m + z^2 \cdot \mathbf{z}^m}, i \in \{1, 2\}$.
We combine these two equations to open the commitments A and S and extract their committed values as follows. Now, let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 0, \sum_{i=1}^{2} \nu_i x_i = 1$. Then, we can compute $\prod_{i=1}^{2} (P_i)^{\nu_i}$ in two ways:

- (i) $\prod_{i=1}^{2} (P_i)^{\nu_i} = \prod_{i=1}^{2} \left(h^{\mu_i} \mathbf{g}^{\tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\tilde{\mathbf{r}}_\mathbf{i}}\right)^{\nu_i} = h^{\sum_{i=1}^{2} \nu_i \mu_i} \mathbf{g}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}} \mathbf{h}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}}$
  $= h^{\sum_{i=1}^{2} \nu_i \mu_i} (\mathbf{g'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{1})}} \dots \mathbf{g'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{l}}_\mathbf{i}^{(\mathbf{l})}}) (\mathbf{h'_1}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(\mathbf{1})}} \dots \mathbf{h'_l}^{\sum_{i=1}^{2} \nu_i \tilde{\mathbf{r}}_\mathbf{i}^{(\mathbf{l})}})$.

- (ii) $\prod_{i=1}^{2}(P_i)^{\nu_i} = \prod_{i=1}^{2}\left(AS^{x_i}(\mathbf{g_1'}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}\dots\mathbf{h_l'})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\right)^{\nu_i}$
$= A^{\sum_{i=1}^{2}\nu_i}S^{\sum_{i=1}^{2}\nu_i x_i}(\mathbf{g_1'}^{\phi^0\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m}\dots\mathbf{h_l'}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$
$= A^0S^1(\mathbf{g_1'}^{\mathbf{0}^m}\dots\mathbf{g_l'}^{\mathbf{0}^m})^{-z}(\mathbf{h_1'}^{\mathbf{0}^m}\dots\mathbf{h_l'}^{\mathbf{0}^m})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$
$= S.$

We thus have : $S = \prod_{i=1}^{2}(P_i)^{\nu_i} = h^{\sum_{i=1}^{2}\nu_i\mu_i}(\mathbf{g_1'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(1)}}\dots\mathbf{g_l'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(l)}})(\mathbf{h_1'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(1)}}\dots\mathbf{h_l'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(l)}})$.
Hence, we can write $S = h^{\rho}(\mathbf{g_1}^{\mathbf{s_L^{(1)}}}\dots\mathbf{g_l}^{\mathbf{s_L^{(l)}}})(\mathbf{h_1}^{\mathbf{s_R^{(1)}}}\dots\mathbf{h_l}^{\mathbf{s_R^{(l)}}})$, where :

- $\rho = \sum_{i=1}^{2}\nu_i\mu_i$,
- $\mathbf{s_L^{(j)}} = \phi^{-j+1}(\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(j)})$, $j \in \{1,\dots,l\}$ and
- $\mathbf{s_R^{(j)}} = \mathbf{y}^{-m}(\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i})$, $j \in \{1,\dots,l\}$.

Now similarly, let $\nu_1,\nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2}\nu_i = 1, \sum_{i=1}^{2}\nu_i x_i = 0$. Then, once again we can do the same computations to get :

$$\begin{aligned}\prod_{i=1}^{2}(P_i)^{\nu_i} &= h^{\sum_{i=1}^{2}\nu_i\mu_i}(\mathbf{g_1'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(1)}}\dots\mathbf{g_l'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(l)}})(\mathbf{h_1'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(1)}}\dots\mathbf{h_l'}^{\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(l)}})\\ &= A^{\sum_{i=1}^{2}\nu_i}S^{\sum_{i=1}^{2}\nu_i x_i}(\mathbf{g_1'}^{\phi^0\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m}\dots\mathbf{h_l'}^{\sum_{i=1}^{2}\nu_i\cdot\mathbf{1}^m})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\\ &= A(\mathbf{g_1'}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}\dots\mathbf{h_l'})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}\end{aligned}$$

Hence, we can write $A = h^{\alpha}(\mathbf{g_1}^{\mathbf{a_L^{(1)}}}\dots\mathbf{g_l}^{\mathbf{a_L^{(l)}}})(\mathbf{h_1}^{\mathbf{a_R^{(1)}}}\dots\mathbf{h_l}^{\mathbf{a_R^{(l)}}})$, where :

- $\alpha = \sum_{i=1}^{2}\nu_i\mu_i$,
- $\mathbf{a_L^{(j)}} = \sum_{i=1}^{2}\nu_i\tilde{\mathbf{l}}_\mathbf{i}^{(j)}\phi^{-j+1} + z\cdot\mathbf{1}^m$, $j \in \{1,\dots,l\}$ and
- $\mathbf{a_R^{(j)}} = \mathbf{y}^{-m}\left(\sum_{i=1}^{2}\nu_i\tilde{\mathbf{r}}_\mathbf{i}^{(j)} - z\cdot\mathbf{y}^m + z^2\cdot\mathbf{z}^m\right)$, $j \in \{1,\dots,l\}$.

Notice that since A and S are provided before the choice of challenges x, y, z, ϕ, we can assume these two expressions hold for any challenge x, y, z, ϕ provided later to the prover in the transcripts considered by our extractor.

Now, we can use the 2 expressions we just found for A and S and substitute them in the equation:
$h^{\mu}\mathbf{g_1'}^{\tilde{\mathbf{l}}^{(1)}}\dots\mathbf{g_l'}^{\tilde{\mathbf{l}}^{(l)}}\mathbf{h_1'}^{\tilde{\mathbf{r}}^{(1)}}\dots\mathbf{h_l'}^{\tilde{\mathbf{r}}^{(l)}} = AS^{x}(\mathbf{g_1'}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}\dots\mathbf{h_l'})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$.

In this way, we obtain the equation:
$h^{\mu}\mathbf{g_1'}^{\tilde{\mathbf{l}}^{(1)}}\dots\mathbf{g_l'}^{\tilde{\mathbf{l}}^{(l)}}\mathbf{h_1'}^{\tilde{\mathbf{r}}^{(1)}}\dots\mathbf{h_l'}^{\tilde{\mathbf{r}}^{(l)}} = \left(h^{\alpha}(\mathbf{g_1'}^{\phi^0\mathbf{a_L^{(1)}}}\dots\mathbf{g_l'}^{\phi^{l-1}\mathbf{a_L^{(l)}}})((\mathbf{h_1'}^{\mathbf{y}^m\circ\mathbf{a_R^{(1)}}}\dots\mathbf{h_l'}^{\mathbf{y}^m\circ\mathbf{a_R^{(l)}}})\right)$
$\left(h^{\rho x}(\mathbf{g_1'}^{\phi^0\mathbf{s_L^{(1)}}x}\dots\mathbf{g_l'}^{\phi^{l-1}\mathbf{s_L^{(l)}}x})((\mathbf{h_1'}^{\mathbf{y}^m\circ\mathbf{s_R^{(1)}}x}\dots\mathbf{h_l'}^{\mathbf{y}^m\circ\mathbf{s_R^{(l)}}x})\right)(\mathbf{g_1'}^{\phi^0\cdot\mathbf{1}^m}\dots\mathbf{g_l'}^{\phi^{l-1}\cdot\mathbf{1}^m})^{-z}(\mathbf{h_1'}\dots\mathbf{h_l'})^{z\cdot\mathbf{y}^m+z^2\cdot\mathbf{z}^m}$

Now using the *Discrete Log Relation* assumption we can infer the following except with negligible probability:

- (i) $\mu = \alpha + \rho x$,
- (ii) $\tilde{\mathbf{l}}^{(\mathbf{j})} = \phi^{j-1}(\mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m + \mathbf{s_L^{(j)}}x)$, $j \in \{1,\dots,l\}$,
- (iii) $\tilde{\mathbf{r}}^{(\mathbf{j})} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2\cdot\mathbf{z}^m$, $j \in \{1,\dots,l\}$.

We can thus write $\tilde{\mathbf{l}}^{(\mathbf{j})} = \phi^{j-1}\mathbf{l}^{(\mathbf{j})}$ and $\tilde{\mathbf{r}}^{(\mathbf{j})} = \mathbf{r}^{(\mathbf{j})}$, where $\mathbf{l}^{(\mathbf{j})} = \mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m + \mathbf{s_L^{(j)}}x$, (1)
and $\mathbf{r}^{(\mathbf{j})} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m + \mathbf{s_R^{(j)}}x) + z^2\cdot\mathbf{z}^m$, $j \in \{1,\dots,l\}$, (2).

Now, recall that $\tilde{\mathbf{l}}, \tilde{\mathbf{r}}$ satisfy : $< \tilde{\mathbf{l}}, \tilde{\mathbf{r}} >= c := \sum_{j=1}^{l}\hat{t}^{(j)}\phi^{j-1}$ or just written differently :
$< (\mathbf{l^{(1)}}\phi^0 || \dots || \mathbf{l^{(l)}}\phi^{l-1}), (\mathbf{r^{(1)}} || \dots || \mathbf{r^{(l)}}) >= \sum_{j=1}^{l} < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \phi^{j-1} = \sum_{j=1}^{l}\hat{t}^{(j)}\phi^{j-1}$.

This last equality holds for each of our transcripts and thus in particular for all l different challenges ϕ. Notice that since, from our early discussion, the expressions of S and A are independent of the value of the challenge ϕ, we can conclude that the $\mathbf{a_R^{(j)}}$, $\mathbf{a_L^{(j)}}$ and consequently the $\mathbf{l^{(j)}}$ and $\mathbf{r^{(j)}}$ are also independent of the value of the challenge ϕ. Hence, if we define $p(\phi) := \sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$, it is a polynomial of degree $l-1$ since the term with highest degree will be ϕ^{l-1} and all coefficients $(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$ are independent of ϕ.
Denote $\Phi = \{\phi_k\}_{k=1}^{l}$, the set of these l different challenges ϕ; then we have :
$\sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >) = 0 \quad \forall \phi \in \{\phi_k\}_{k=1}^{l}$. Since we assume $\phi_i \neq \phi_j$, $i \neq j$, this implies that the polynomial $p(\phi) = \sum_{j=1}^{l} \phi^{j-1}(\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >)$ has (at least) l distinct roots $\{\phi_k\}_{k=1}^{l}$. The only polynomial of degree $l-1$ with more than l distinct roots is the polynomial 0, with 0 coefficients. Hence, $p(\phi)$ is the 0 polynomial and thereby all his coefficients are 0, that is:
$\hat{t}^{(j)} - < \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >= 0 \Leftrightarrow \hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} > \forall j \in \{1, \dots, l\}$. (3)

Now, given the challenges z, y, ϕ, we consider transcripts with 3 different challenges $x : x_1, x_2, x_3$ and the verification equation : $(g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}}) h^{\tau_{x_i}} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2}$, that must be verified $\forall i \in \{1, 2, 3\}$.
Let $\{\nu_j\}_{j=1}^{3} \subset \mathbb{Z}_p$ such that $\sum_{j=1}^{3} \nu_j = 0, \sum_{j=1}^{3} \nu_j x_j = 1, \sum_{j=1}^{3} \nu_j x_j^2 = 0$, then we can compute :

$$\begin{aligned}
\prod_{i=1}^{3} \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}}) h^{\tau_{x_i}} \right)^{\nu_i} &= (g_1^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(l)}}) h^{\sum_{i=1}^{3} \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^{3} \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2} \right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^{3} \nu_i \cdot z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\sum_{i=1}^{3} \nu_i \delta(y,z)} T_1^{\sum_{i=1}^{3} \nu_i x_i} T_2^{\sum_{i=1}^{3} \nu_i x_i^2} \\
&= T_1
\end{aligned}$$

Hence, we can write $T_1 = h^{\tau_1} g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}}$, where $\tau_1 = \sum_{i=1}^{3} \nu_i \tau_{x_i}$, $t_1^{(j)} = \sum_{i=1}^{3} \nu_i \hat{t}_i^{(j)}$, $j \in \{1, \dots, l\}$.

Similarly, let $\{\nu_j\}_{j=1}^{3} \subset \mathbb{Z}_p$ such that $\sum_{j=1}^{3} \nu_j = 0, \sum_{j=1}^{3} \nu_j x_j = 0, \sum_{j=1}^{3} \nu_j x_j^2 = 1$, then we can compute:

$$\begin{aligned}
\prod_{i=1}^{3} \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}}) h^{\tau_{x_i}} \right)^{\nu_i} &= (g_1^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(l)}}) h^{\sum_{i=1}^{3} \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^{3} \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2} \right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^{3} \nu_i z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\sum_{i=1}^{3} \nu_i \delta(y,z)} T_1^{\sum_{i=1}^{3} \nu_i x_i} T_2^{\sum_{i=1}^{3} \nu_i x_i^2} \\
&= T_2
\end{aligned}$$

Hence, we can write $T_2 = h^{\tau_2} g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}}$, where $\tau_2 = \sum_{i=1}^{3} \nu_i \tau_{x_i}$, $t_2^{(j)} = \sum_{i=1}^{3} \nu_i \hat{t}_i^{(j)}$, $j \in \{1, \dots, l\}$.

Finally, let $\{\nu_j\}_{j=1}^{3} \subset \mathbb{Z}_p$ such that $\sum_{j=1}^{3} \nu_j = 1, \sum_{j=1}^{3} \nu_j x_j = 0, \sum_{j=1}^{3} \nu_j x_j^2 = 0$, then we can compute :

$$\begin{aligned}
\prod_{i=1}^{3} \left((g_1^{\hat{t}_i^{(1)}} \dots g_l^{\hat{t}_i^{(l)}}) h^{\tau_{x_i}} \right)^{\nu_i} &= (g_1^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(1)}} \dots g_l^{\sum_{i=1}^{3} \nu_i \hat{t}_i^{(l)}}) h^{\sum_{i=1}^{3} \nu_i \tau_{x_i}} \\
&= \prod_{i=1}^{3} \left(\mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)} T_1^{x_i} T_2^{x_i^2} \right)^{\nu_i} \\
&= \mathbf{V}^{\sum_{i=1}^{3} \nu_i z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\sum_{i=1}^{3} \nu_i \delta(y,z)} T_1^{\sum_{i=1}^{3} \nu_i x_i} T_2^{\sum_{i=1}^{3} \nu_i x_i^2} \\
&= \mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1 \dots g_l)^{\delta(y,z)}
\end{aligned}$$

Hence, we can write $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} = h^{\gamma} g_1^{v_1} \dots g_l^{v_l}$, where $\gamma = \sum_{i=1}^{3} \nu_i \tau_{x_i}$ and $v_j = \sum_{i=1}^{3} \nu_i \hat{t}_i^{(j)} - \delta(y,z)$, $j \in \{1, \dots, l\}$.

Now, given y, x, ϕ, consider m transcripts with different challenges $\{z_j\}_{j=1}^{m}$. From our previous discussion, we can write $\mathbf{V}^{z_j^2 \cdot \mathbf{z_j}^m} = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$.

Given $j \in \{1,\dots,m\}$, let $\{\nu_i\}_{i=1}^m \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^m \nu_i z_i^{1+j} = 1$ and $\sum_{i=1}^m \nu_i z_i^{1+k} = 0\ \forall k \in (\{1,\dots,m\}\backslash\{j\})$.
In other words, $\{\nu_i\}_{i=1}^m$ are such that given $k \in \{1,\dots,m\}$: $\sum_{i=1}^m z_i^{k+1}\nu_i = \Gamma(k,j) := \begin{cases} 1, \text{ if } j = k, \\ 0, \text{ otherwise.} \end{cases}$
In what follows below, we denote V_j as the j-th component of the m-dimensional vector $\mathbf{V}$, i.e. $\mathbf{V} = (V_j)_{j=1}^m = (V_1,\dots,V_m)$.
Then, we can compute the product $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i}$ as :

- $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^m \left(\prod_{k=1}^m V_k^{z_i^{k+1}}\right)^{\nu_i} = \prod_{k=1}^m \left[\prod_{i=1}^m \left(V_k^{z_i^{k+1}}\right)^{\nu_i}\right] = \prod_{k=1}^m \left(V_k^{\sum_{i=1}^m z_i^{k+1}\nu_i}\right)$
  $= \prod_{k=1}^m V_k^{\Gamma(k,j)} = V_j^{\Gamma(j,j)} = V_j$, since $V_k^{\Gamma(k,j)} = 1\ \forall k \neq j$.

- $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^m \left(h^{\gamma_i} g_1^{v_1^{(i)}} \dots g_l^{v_l^{(i)}}\right)^{\nu_i} = h^{\sum_{i=1}^m \gamma_i\nu_i} g_1^{\sum_{i=1}^m v_1^{(i)}\nu_i} \dots g_l^{\sum_{i=1}^m v_l^{(i)}\nu_i}$

Hence given any $j \in \{1,\dots,m\}$, we can write $V_j = h^{\overline{\gamma}_j} g_1^{\overline{v}_1^{(j)}} \dots g_l^{\overline{v}_l^{(j)}}$, where $\overline{\gamma}_j = \sum_{i=1}^m \gamma_i\nu_i$ and $\overline{v}_k^{(j)} = \sum_{i=1}^m v_k^{(i)}\nu_i$, $k \in \{1,\dots,l\}$.

We have succeed to extract a candidate witness $\{(\overline{\gamma}_j, v_1^{(j)},\dots,v_l^{(j)})\}$ such that $h^{\overline{\gamma_j}} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} = V_j$.
It remains to show that $\{(\overline{\gamma}_j, v_1^{(j)},\dots,v_l^{(j)})\}$ satisfy the relation R_1, that is :
$v_i^{(j)} \in \{0,1\}\ \forall (i,j) \in \{1,\dots,l\} \times \{1,\dots,m\}$.
We show this by recombining all the expressions we found earlier for the commitments and exploiting the different verification equations.

Now, given any transcript with fixed challenges x, y, z, ϕ we can substitute the expressions we just found for $(V_j)_{j=1}^m$, T_1 and T_2 in the equation, $(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}})h^{\tau_x} = \mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2}$ to get :

$$
\begin{aligned}
(g_1^{\hat{t}^{(1)}} \dots g_l^{\hat{t}^{(l)}})h^{\tau_x} &= \mathbf{V}^{z^2\cdot\mathbf{z}^m}(g_1 \dots g_l)^{\delta(y,z)} T_1^x T_2^{x^2} \\
&= \prod_{j=1}^m V_j^{z^{j+1}} (g_1 \dots g_l)^{\delta(y,z)} (h^{\tau_1} g_1^{t_1^{(1)}} \dots g_l^{t_1^{(l)}})^x (h^{\tau_2} g_1^{t_2^{(1)}} \dots g_l^{t_2^{(l)}})^{x^2} \\
&= \prod_{j=1}^m (h^{\overline{\gamma}_j} g_1^{\overline{v}_1^{(j)}} \dots g_l^{\overline{v}_l^{(j)}})^{z^{j+1}} (g_1 \dots g_l)^{\delta(y,z)} h^{\tau_1 x} g_1^{t_1^{(1)} x} \dots g_l^{t_1^{(l)} x} h^{\tau_2 x^2} g_1^{t_2^{(1)} x^2} \dots g_l^{t_2^{(l)} x^2} \\
&= (h^{\sum_{j=1}^m \overline{\gamma}_j z^{j+1}} g_1^{\sum_{j=1}^m \overline{v}_1^{(j)} z^{j+1}} \dots g_l^{\sum_{j=1}^m \overline{v}_l^{(j)} z^{j+1}}) g_1^{\delta(y,z)} \dots g_l^{\delta(y,z)} h^{\tau_1 x + \tau_2 x^2} g_1^{t_1^{(1)} x + t_2^{(1)} x^2} \dots g_l^{t_1^{(l)} x + t_2^{(l)} x^2} \\
&= h^{\tau_1 x + \tau_2 x^2 + \sum_{j=1}^m \overline{\gamma}_j z^{j+1}} g_1^{\sum_{j=1}^m \overline{v}_1^{(j)} z^{j+1} + \delta(y,z) + t_1^{(1)} x + t_2^{(1)} x^2} \dots g_l^{\sum_{j=1}^m \overline{v}_l^{(j)} z^{j+1} + \delta(y,z) + t_1^{(l)} x + t_2^{(l)} x^2}
\end{aligned}
$$

Now using the *Discrete Log Relation* assumption we can infer the following except with negligible probability :

- (i) $\tau_x = \tau_1 x + \tau_2 x^2 + \sum_{j=1}^m \overline{\gamma}_j z^{j+1}$,
- (ii) $\hat{t}^{(j)} = \sum_{k=1}^m \overline{v}_j^{(k)} z^{k+1} + \delta(y,z) + t_1^{(j)} x + t_2^{(j)} x^2$, $j \in \{1,\dots,l\}$.

We can rewrite the last expression as : $\hat{t}^{(j)} = t_0^{(j)} + t_1^{(j)} x + t_2^{(j)} x^2$ (4), $j \in \{1,\dots,l\}$, where
$t_0^{(j)} = \sum_{k=1}^m \overline{v}_j^{(k)} z^{k+1} + \delta(y,z)$. For the rest of the proof, it remains to show that
$t_0^{(j)} =< \mathbf{a_L^{(j)}} - z\cdot\mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\cdot\mathbf{1}^m) + z^2\cdot\mathbf{z}^m > \forall j \in \{1,\dots,l\}$.
Once we have this we can show that that $\{(\gamma_j, v_1^{(j)},\dots,v_l^{(j)})\}$ is a valid witness for relation R_1.
Indeed, recall that in section 7.1 we have mentioned that the equation $< \mathbf{a_L^{(j)}} - z\mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z\mathbf{1}^m) + z^2\cdot\mathbf{z}^m >= \sum_{k=1}^m z^{1+k} v_j^{(k)} + \delta(y,z)$ implies except with negligible probability the three following equations $\forall j \in [1,l]$: $\mathbf{a_L^{(j)}}[i] = v_j^{(i)}, i \in \{1,\dots,m\}, \mathbf{a_L^{(j)}} \circ \mathbf{a_R^{(j)}} = \mathbf{0}^m, \mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$, which proves $v_i^{(j)} \in \{0,1\}$.

For this, given $j \in [1,l]$, define the two linear vectors polynomials $p_L^{(j)}(X), p_R^{(j)}(X) \in \mathbb{Z}_p^m[X]$ and the two quadratic scalar polynomials $p^{(j)}(X), t^{(j)}(X) \in \mathbb{Z}_p[X]$ that will depend of the challenge x as follows:

- (i) $p_L^{(j)}(X) = \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(j)}} X$,
- (ii) $p_R^{(j)}(X) = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m + \mathbf{s_R^{(j)}} X) + z^2 \cdot \mathbf{z}^m$,
- (iii) $p^{(j)}(X) =< p_L^{(j)}(X), p_R^{(j)}(X) >= p_0^{(j)} + p_1^{(j)} X + p_2^{(j)} X^2$,
- (iv) $t^{(j)}(X) = t_0^{(j)} + t_1^{(j)} X + t_2^{(j)} X^2$.

Now, recall that given a transcript with challenges x, y, z, ϕ, we have shown earlier (3) : $\hat{t}^{(j)} =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$ and by (4) we have : $\hat{t}^{(j)} = t^{(j)}(x)$ so that $t^{(j)}(x) =< \mathbf{l^{(j)}}, \mathbf{r^{(j)}} >$.
Moreover, from (1) and (2) we have the following: $\mathbf{l^{(j)}} = \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m + \mathbf{s_L^{(j)}} x = p_L^{(j)}(x)$ and
$\mathbf{r^{(j)}} = \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m + \mathbf{s_R^{(j)}} x) + z^2 \cdot \mathbf{z}^m = p_R^{(j)}(x)$.

This implies in particular that for all challenges z, y, ϕ and 3 different challenges x_1, x_2, x_3, we have
$t^{(j)}(x_i) =< p_L^{(j)}(x_i), p_R^{(j)}(x_i) >= p^{(j)}(x_i)$, $i \in \{1, 2, 3\}$. This can be expressed equivalently as the fact that the scalar quadratic polynomial $\sum_{k=0}^{2}(t_k^{(j)} - p_k^{(j)})X^k \in \mathbb{Z}_p[X]$ admits at least the 3 following roots: (x_1, x_2, x_3). Since the only quadratic polynomial in $\mathbb{Z}_p$ with more than 2 roots is the zero polynomial, we can conclude that
$t_k^{(j)} - p_k^{(j)} = 0 \Leftrightarrow t_k^{(j)} = p_k^{(j)}$ $\forall k \in \{0, 1, 2\}$.
In particular, $t_0^{(j)} = p_0^{(j)} \Leftrightarrow \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + \delta(y, z) =< \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >$.

Now, using the proof in Appendix A, this can be expressed equivalently as :

$$\sum_{k=1}^{m} (\mathbf{a_L^{(j)}}[k] - v_j^{(k)}) z^{k+1} + z < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m > + < \mathbf{a_L^{(j)}}, \mathbf{a_R^{(j)}} \circ \mathbf{y}^m >= 0$$

Since this holds for $m + 2$ different challenges z and m different challenges y (using again the reasoning that a polynomial of degree m with more than m roots is the 0 polynomial) we can infer the following :

- (i) $\mathbf{a_L^{(j)}} \circ \mathbf{a_R^{(j)}} = \mathbf{0}^m$,
- (ii) $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^m$,
- (iii) $v_j^{(k)} = \mathbf{a_L^{(j)}}[k]$ $\forall k \in \{1, \ldots, l\}$.

(i) and (ii) together imply that $\mathbf{a_L^{(j)}} \in \{0, 1\}^m$ while (iii) implies $v_i^{(j)} \in \{0, 1\}$ $\forall i \in \{1, \ldots, l\}$.
Since all this final reasoning holds for any $j \in \{1, \ldots, m\}$ and $V_j = h^{\overline{\gamma_j}} g_1^{\overline{v}_1^{(j)}} \ldots g_l^{\overline{v}_l^{(j)}}$ $\forall j \in \{1, \ldots, m\}$, we have extracted a valid witness $(\overline{v_1}, \ldots, \overline{v_l}, \overline{\gamma})$ for the relation:

$$R_1 = \{(g_1, \ldots, g_l, h \in \mathbb{G}, \mathbf{g_1}, \ldots, \mathbf{g_l}, \mathbf{V} = (V_1, \ldots, V_m) \in \mathbb{G}^m; v_1, \ldots, v_l, \gamma \in \mathbb{Z}_p^m) :$$
$$V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}} \wedge v_i^{(j)} \in \{0, 1\}, \forall i \in [1, l], j \in [1, m]\}.$$

Using the *Forking Lemma* presented in section 3.4 this implies the *Computational Witness Extended Emulation* property (3.3.4) which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The 0-1 protocol is in $6 + (2log_2(ml) + 4log_2(l) + 1) = 2log_2(m) + 6log_2(l) + 7$ moves and has $3 + log_2(ml) + 2log_2(l)$ challenge $(y, z), x, \phi$, $log_2(ml)$ challenges for the first "Bulletproofs" inner product argument on input $n = ml$, $log_2(l)$ for the second "Bulletproofs" inner product argument on input $n = l$ and $log_2(l)$ for the "Extended-Schnorr" argument on input $n = l$.

Hence the parameter μ in the statement of the *Forking Lemma* is equal to $3 + log_2(m) + 3log_2(l)$. The corresponding tree of accepting transcripts has depth equal to $\mu = 3 + log_2(m) + 3log_2(l)$. The node in the first level has $n_1 = m(m + 2)$ children in the second level of the tree corresponding to the m^2 pairs of challenges (y, z) for the first challenge $(\{y_j\}_{j=1}^m \times \{z_j\}_{j=1}^m)$. Each node in the second level has $n_2 = 3$ children each corresponding to a different value of the challenge x $(\{x_1, x_2, x_3\})$. Each node in the third level has $n_3 = l$ children corresponding to the l different challenges ϕ $(\{\phi_i\}_{i=1}^l)$. Each node in the fourth level of our tree is connected to the root of a tree of accepting transcripts for the first "Bulletproofs" inner product argument on input $n = ml$, which is a tree of depth $log_2(ml)$ with 4 children per node at each level (see the "Bulletproofs" paper), hence $n_i = 4$, $4 \leq i \leq 4 + log_2(ml)$.

Then, each node of the last round of the first "Bulletproofs" inner product argument is connected to a tree of accepting transcripts for the second "Bulletproofs" inner product argument on input $n = l$, which is a tree of depth $log_2(l)$ with 4 children per node (see the "Bulletproofs" paper), hence $n_i = 4$, $5+log_2(ml) \leq i \leq 5+log_2(ml)+log_2(l)$.

Finally, each node of the last round of the second "Bulletproofs" inner product argument is connected to a tree of accepting transcripts for the "Extended-Schnorr" argument on input $n = l$, which is a tree of depth $log_2(l)$ with 3 children per node (see the "Bulletproofs" paper), hence $n_i = 3$, $5 + log_2(ml) + log_2(l) \leq i \leq 5 + log_2(ml) + 2log_2(l)$.

Therefore, the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = \prod_{i=1}^{3+log_2(ml)+2log_2(l)} n_i = n_1 \cdot n_2 \cdot n_3 \cdot \prod_{i=4}^{3+log_2(ml)} n_i \cdot \prod_{i=4+log_2(ml)}^{4+log_2(ml)+log_2(l)} n_i \cdot \prod_{i=5+log_2(ml)+log_2(l)}^{5+log_2(ml)+2log_2(l)} n_i = m^2 \cdot 3 \cdot l \cdot 4^{log_2(ml)} \cdot 4^{log_2(l)} \cdot 3^{log_2(l)} = 3lm(m + 2)(ml)^2 l^2 l^{1.58} = 3(m^3 + 2m^2)l^{6.58} = N^{0-1}(ml)$, which is clearly bounded above by a polynomial in the parameter λ (the constant polynomial $3(m^3 + 2m^2)l^7$ for example). In addition, χ is in polynomial time since computing the ν_i amounts each time to solving a system of linear equations in $\mathbb{Z}_p$ for example for the opening of the commitment A: $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ in $\mathbb{Z}_p$, which can be done efficiently. All other operations are also basic polynomial time computations. Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute the ν_i to open a commitment, which occurs only when the determinant of the matrix is 0 which happens only with negligible probability. For example for the opening of the commitment A, the matrix corresponding to the linear system of equation in $\mathbb{Z}_p$ is $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$. The determinant is $x_1 - x_2$. It is zero when $x_1 = x_2$, which occurs with negligible probability $(1/p)$ since x_1 and x_2 are picked uniformly at random in $\mathbb{Z}_p$.

Chapter 9

Min-Max K-selection Proof Protocol

9.1 Selection Limit

In an election, recall that the selection limit is the maximum number of selections a voter can choose.
More formally, in an election, assume a ballot has the form $\mathbf{v} = (v_j)_{j=1}^{l}$, with $v_j \in \{0,1\}$, $v_j = \begin{cases} 1 & \text{if box } j \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$
Then, usually, the selection limit is a natural number $K \geq 1$ that is to be determined before the election is organised such that $\sum_{j=1}^{l} v_j \leq K$ for all ballots $(v_j)_{j=1}^{l}$ in the election. However, sometimes we can also consider a lower bound so we will prove more generally $\sum_{j=1}^{l} v_j \in [Min, Max]$

For example, in a contest with 5 candidates if the selection limit is set to 1 this means that the voter can only choose (i.e. give his vote) to one of the 5 candidates.
The selection limit is often (depending on the election's system) set to 1 however to be general we allow our system to work for any value.

9.2 K-selection Proof

In this section, we want to prove the following statement in order to prove a batch of ballots is valid: given multi-Pedersen commitments of votes $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$, $j \in \{1, \dots, m\}$, the committed ballots $\{v_1^{(j)}, \dots, v_l^{(j)}\}_{j=1}^{m}$ satisfy $\sum_{k=1}^{l} v_k^{(j)} \in [Min, Max]$.
We can then prove the statement equivalently by showing that $\sum_{k=1}^{l} v_k^{(j)}$ can be encoded on n bits.

We assume without loss of generality that $Min = 0$ and Max is a power of 2, so that $\sum_{k=1}^{l} v_k^{(j)} \in [Min, Max]$ is equivalent to $\sum_{k=1}^{l} v_k^{(j)} \in [0, 2^n - 1]$.
Written formally the relation we want to prove is the following:

$$\begin{aligned} R_2 = \{ & (g_1, \dots, g_l, h \in \mathbb{G}, \mathbf{V} = (V_1, \dots, V_m) \in \mathbb{G}^m; \\ & (v_1^{(1)}, \dots, v_1^{(m)}), \dots, (v_l^{(1)}, \dots, v_l^{(m)}), \gamma = (\gamma_1, \dots, \gamma_m) \in \mathbb{Z}_p^m) : \\ & \left[V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge \sum_{i=1}^{l} v_i \in [0, 2^n - 1] \right] \} \end{aligned}$$

The C.R.S. is $\sigma := (g_1, \dots, g_l, h)$, the statement is $u := \mathbf{V} = (V_1, \dots, V_m)$ and the witness is $w := \left((v_i^{(j)})_{i=1,j=1}^{l,m} || (\gamma_j)_{j=1}^{m} \right)$.

9.3 Proof Key ideas

In this section, we briefly describe the key ideas we had in mind when we designed the following protocol to prove the relation R_2 presented above.
Once again, the ideas are inspired by the "Bulletproofs" paper.

9.3.1 Rewriting relation R_2

In order to prove relation R_2, that is proving, for each commitment $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$, that we have $\sum_{k=1}^{l} v_k^{(j)} \in [0, 2^n - 1]$, we prove that $\sum_{k=1}^{l} v_k^{(j)}$ can be encoded on n bits, that is there exists $\mathbf{a_L^{(j)}} = (\mathbf{a_l^{(j)}}[1], \dots, \mathbf{a_l^{(j)}}[n]) \in \{0,1\}^n$ such that : $< \mathbf{a_L^{(j)}}, \mathbf{2}^n >= \sum_{k=1}^{l} v_k^{(j)}$.
We will denote $\mathbf{a_L}$ as $\mathbf{a_L} := (\mathbf{a_L^{(1)}} || \dots || \mathbf{a_L^{(l)}}) \in \mathbb{Z}_p^{n \cdot m}$.

We will construct a protocol where the prover, $\mathcal{P}$, to prove relation R_2, shows in a zero-knowledge fashion the knowledge of $\mathbf{a_L} = (\mathbf{a_L^{(1)}} || \dots \mathbf{a_L^{(m)}})$ such that :

$$< \mathbf{v^j}, \mathbf{1}^l >=< \mathbf{a_L^{(j)}}, \mathbf{2}^n >, j \in \{1, \dots, m\}, \mathbf{a_L} \circ \mathbf{a_R} = \mathbf{0}^{m \cdot n}, \mathbf{a_R} = \mathbf{a_L} - \mathbf{1}^{m \cdot n}$$

where $\mathbf{v^j} = (v_1^{(j)}, \dots, v_l^{(j)})$. The second and third equations ensure that $\mathbf{a_L}$ is a sequence of $n \cdot m$ bits and hence that $a_L^{(j)}$ is a sequence of n bits $\forall j \in [1, m]$, i.e. $\mathbf{a_L} \in \{0,1\}^{n \cdot m}$ and $\mathbf{a_L^{(j)}} \in \{0,1\}^n$.

We want the proof system only to reveal that $\sum_{k=1}^{l} v_k^{(j)} \in [0, 2^n - 1]$ and not reveal any information about the votes. In particular it should not reveal $\{v_i^{(j)}\}_{i=1,j=1}^{l,m}$ or the values $\{\sum_{k=1}^{l} v_k^{(j)}\}_{j=1}^{m}$. The prover $\mathcal{P}$ thus cannot send in clear the values $\mathbf{a_L}$ to the verifier $\mathcal{V}$. He will instead commit on the values $\mathbf{a_L}$ and $\mathbf{a_R}$ through the perfectly hiding multi-Pedersen commitment $A = h^{\alpha} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}}$ and we will prove in a zero-knowledge fashion using commitment A that the above equations are satisfied.

9.3.2 Adapting relation R_2 for "Bulletproofs"

We will reduce the above system of equations to a single dot product equation in order to use later on the "Bulletproofs" inner product argument to have compact proofs (small communication volume between $\mathcal{P}$ and $\mathcal{V}$).

Using the *Schwartz-Zippel Lemma* presented in section 3.2.1, the system of equations presented above are equivalent except with negligible probability to the following system of equations for y picked uniformly at random in $\mathbb{Z}_p$:

$$< \mathbf{v^j}, \mathbf{1}^l >=< \mathbf{a_L^{(j)}}, \mathbf{2}^n >, j \in \{1, \dots, m\}, \sum_{k=1}^{m \cdot n} \mathbf{a_L}[k] \cdot \mathbf{a_R}[k] = 0, \sum_{k=1}^{m \cdot n} (\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k]) = 0.$$

We can rewrite these $m + 2$ equations equivalently in dot-product form as follows :

$$< \mathbf{v^j}, \mathbf{1}^l >=< \mathbf{a_L^{(j)}}, \mathbf{2}^n >, j \in \{1, \dots, m\}, < \mathbf{a_L} - \mathbf{1}^{m \cdot n} - \mathbf{a_R}, \mathbf{y}^{m \cdot n} >= 0, < \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{m \cdot m} >= 0.$$

Using the *Schwartz-Zippel Lemma* a second time, these $m + 2$ equations are equivalent to the following single equation for z picked uniformly at random in $\mathbb{Z}_p$:

$$\sum_{j=1}^{m} (< \mathbf{v^j}, \mathbf{1}^l > - < \mathbf{a_L^{(j)}}, \mathbf{2}^n >) z^{j+1} + < \mathbf{a_L} - \mathbf{1}^{m \cdot n} - \mathbf{a_R}, \mathbf{y}^{m \cdot n} > z + < \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n \cdot m} >= 0,$$

which can be rewritten equivalently in one single dot product equation :

$$\boxed{< \mathbf{a_L} - z \cdot \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n \cdot m}) + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} >= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y, z)}$$

where $\delta(y, z) := (z - z^2) < \mathbf{1}^{m \cdot n}, \mathbf{y}^{m \cdot n} > - \sum_{j=1}^{m} < \mathbf{1}^n, \mathbf{2}^n > z^{j+2}$ as proved in appendix B.

We have at the end obtained a dot product verification equation that looks very similar to the one obtained for the 0-1 protocol in section 8, that was :

$$< \mathbf{a_L^{(j)}} - z \mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >= \sum_{k=1}^{m} z^{1+k} v_j^{(k)} + \delta(y, z)$$

For the analogy, observe that $z^2 \cdot \mathbf{z}^m = \sum_{j=1}^{m} (\mathbf{0}^{j-1} || 1 || \mathbf{0}^{m-j})$. You will then recognize a lot of similar terms between the two equations. Notice that in the K-selection protocol verification equation just presented above we have the term $< \mathbf{v^j}, \mathbf{1}^l >$ instead of the term $v_j^{(k)}$ in the 0-1 protocol verification equation. This difference between those two terms stems from the very nature of what we wish to demonstrate: on one side $v_j^{(k)} \in \{0,1\}\ \forall k \in \{0,1\}$ and $\forall j \in \{1, \dots, m\}$, on the other side $< \mathbf{v^j}, \mathbf{1}^l >= \sum_{k=1}^{l} v_k^{(j)} \in [0, 2^n - 1]\ \forall j \in \{1, \dots, m\}$.

9.3.3 Dot product equation verification

We have obtained a single dot product equation of the following form: $< A, B >= C$.
Assume the verifier $\mathcal{V}$ can compute the term C. Then, if the prover $\mathcal{P}$ sends to the verifier the two vectors representing the left-hand side and the right-hand side of the dot product (A and B respectively), the verifier could just verify the equality between the dot product and the right-hand side of the equation: $< A, B >= C$. However, sending in clear the terms A and B breaks the zero-knowledge argument because they contain information about the witness.

To remedy this problem, we will proceed as in the Schnorr's proof and in the 0-1 protocol, that is we will blind these two terms using the two following blinding factors denoted $\mathbf{s_L} \in \mathbb{Z}_p^{n \cdot m}$ and $\mathbf{s_R} \in \mathbb{Z}_p^{n \cdot m}$ and a challenge $x \in \mathbb{Z}_p$, also like in the Schnorr's Protocol, which allows to send them in clear while preserving the zero-knowledge property.

Commitment on A and B

The resulting blinded left-hand side term of the dot product equation will be denoted $\mathbf{l}$ $(= \mathbf{a_L} - z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s_L} \cdot x)$ while the right-hand side term will be denoted $\mathbf{r}$ $(= \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m} + \mathbf{s_R} x) + \sum_{j=1}^{m} z^{1+j} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}))$.

We will then combine all the commitments to create a commitment over the left-hand side and the right-hand side of the dot product. The verifier then checks that this commitment denoted P is valid and compares it to the commitment on C.

$\mathcal{P}$ sends then $\mathbf{l}$ and $\mathbf{r}$ to $\mathcal{V}$.
$\mathcal{P}$ and $\mathcal{V}$ can compute a commitment over $\mathbf{a_L} - z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s_L} \cdot x$ and $\mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m} + \mathbf{s_R} x) + \sum_{j=1}^{m} z^{1+j} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n})$ as $P := AS^x \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu}$ so to check that $\mathcal{P}$ has honestly computed $\mathbf{l}$ and $\mathbf{r}$ through the verification equation $\mathbf{g^l h^r} = P$.

However, in order to reduce the number of communications $\mathcal{P}$ will not send $\mathbf{l}$ and $\mathbf{r}$ but instead engage in a "Bulletproofs" inner product argument to verify the equation.

Commitment on C

Recall that in the 0-1 protocol, we were able, using the commitments $\{V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{[j)}}\}$, $j \in [1, m]$, to compute directly a commitment over the right-hand side terms of the dot product equation that is a commitment over the terms $C_j = \sum_{k=1}^{m} z^{1+k} v_j^{(k)} + \delta(y, z)$ in the dot-product equation $< A_j, B_j >= C_j$, $j \in \{1 \dots, l\}$.
Indeed, $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} = \prod_{i=1}^{m} \left[h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \right]^{z^{j+1}} = h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g_1^{\sum_{j=1}^{m} v_1^{(j)} z^{j+1}} \dots g_l^{\sum_{j=1}^{m} v_l^{(j)} z^{j+1}}$. Hence, multiplying $\mathbf{V}^{z^2 \cdot \mathbf{z}^m}$ by $(g_1, \dots, g_l)^{\delta(y,z)}$ gives a commitment over $C_1, \dots, C_l$:
$\mathbf{V}^{z^2 \cdot \mathbf{z}^m} (g_1, \dots, g_l)^{\delta(y,z)} = h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g_1^{\sum_{j=1}^{m} v_1^{(j)} z^{j+1} + \delta(y,z)} \dots g_l^{\sum_{j=1}^{m} v_l^{(j)} z^{j+1} + \delta(y,z)} = h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g_1^{C_1} \dots g_l^{C_l}$.

On the contrary here we cannot compute directly the right-hand side term, $C = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y, z)$, of the dot product equation using the commitments $\{V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{[j)}}\}$, $j \in [1, m]$. The reason is that given $j \in \{1 \dots, m\}$, all values $(v_k^{(j)})_{k=1}^{l}$ are inside the same commitment but on different basis : $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$.
We do not have one different commitment for each value $v_k^{(j)}$ which would allow to sum over them by multiplying the commitments or combining them in any other way (without the prover's help) to obtain the sum of the votes $< \mathbf{v^j}, \mathbf{1}^l >$.
Generally speaking given some commitment $h^b g_1^{a_1} \dots g_n^{a_n}$, there is no possible way of getting a commitment over the value $\sum_{i=1}^{n} a_i$. So, the verifier cannot compute the commitment over the sum $< \mathbf{v^j}, \mathbf{1}^l >$. At least not without the prover's help .

We will use the commitments $\{V_j\}_{j=1}^{m}$ indirectly by first asking the prover $\mathcal{P}$ to send a commitment $T_0 = h^{\tau_0} g^{t_0}$ over the value $t_0 := \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}$ to the verifier $\mathcal{V}$. Then, using $\{V_j\}_{j=1}^{m}$ in the protocol, the prover proves in a zero-knowledge fashion to the verifier that the commitment T_0 is indeed a commitment over the value $t_0 = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}$. This constitutes the second part of the proof.
Let us see how the prover $\mathcal{P}$ can actually convince the verifier $\mathcal{V}$ that $T_0 = h^{\tau_0} g^{t_0}$ is a commitment over the value $t_0 = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}$ with $\mathbf{v^j} = (v_1^{(j)}, \dots, v_l^{(j)})$ such that $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}}$, $j \in \{1, \dots, m\}$.

For this, let us define $\overline{v}_k := \sum_{j=1}^{m} v_k^{(j)} z^{j+1}$, $k \in \{1 \dots, l\}$.
Then, we have $t_0 = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} = \sum_{j=1}^{m} \sum_{k=1}^{l} v_k^j z^{j+1} = \sum_{k=1}^{l} \sum_{j=1}^{m} v_k^{(j)} z^{j+1} = \sum_{k=1}^{l} \overline{v}_k =$
$< (\overline{v}_1, \dots, \overline{v}_k), \mathbf{1}^l >$.
In addition, $\mathcal{P}$ and $\mathcal{V}$ can compute a multi-Pedersen commitment over the values $(\overline{v}_1, \dots, \overline{v}_k)$ as follows : $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} = \prod_{i=1}^{m} \left[h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \right]^{z_{j+1}} = h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g_1^{\sum_{j=1}^{m} v_1^{(j)} z^{j+1}} \dots g_l^{\sum_{j=1}^{m} v_l^{(j)} z^{j+1}} = h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g_1^{\overline{v}_1} \dots g_l^{\overline{v}_l}$.

Let's assume first a version of our scheme where $\mathcal{P}$ sends a blinded version of $\overline{v}_k$ as $\tilde{v}_k := \overline{v}_k + s_k \cdot x'$, $k \in \{1, \dots, l\}$, to $\mathcal{V}$, where s_k is the blinding factor and x' is the challenge. They are picked uniformly at random by the prover and the verifier respectively.
$\mathcal{V}$ will then verify the two following equations :

- (i) $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} \stackrel{?}{=} h^{\mu} g_1^{\tilde{v}_1} \dots g_l^{\tilde{v}_l}$,
- (ii) $g^{\sum_{k=1}^{l} \tilde{v}_k} h^{\tau_{x'}} \stackrel{?}{=} T_0 T_1'^{x'}$.

The first verification will guarantee that $\tilde{v}_k = \overline{v}_k + s_k \cdot x'$, while the second equation together with the first equation implies that $T_0 = h^{\tau_0} g^{t_0}$ is a commitment over $t_0 = \sum_{k=1}^{l} \overline{v}_k = \sum_{j=1}^{m} \sum_{k=1}^{l} v_k^{(j)} z^{j+1}$, which is what we wanted.
In the first equation, $S' = h^{\rho'} g_1^{s_1} \dots g_l^{s_l}$ is a commitment on the blinding values $(s_k)_{k=1}^{l}$, while $\mu = \sum_{j=1}^{m} \gamma_j z^{j+1} + \rho' \cdot x'$ to maintain the equality in base h.
In the second equation, $T_1' = h^{\tau_1'} g^{t_1'}$ is a commitment over $t_1' = \sum_{k=1}^{l} s_k$, so that on both sides of the equation we have $\sum_{k=1}^{l} \tilde{v}_k = t_0 + t_1' \cdot x'$, while $\tau_{x'} = \tau_0 + \tau_1' \cdot x'$ to maintain the equality in base h.
Above, τ_1' and ρ' are picked uniformly at random to have perfectly hiding commitments T_1' and S'.

However, in practice in order to reduce the number of communications between $\mathcal{P}$ and $\mathcal{V}$ we will not send in clear the values $\tilde{v}_k$, $k \in [1, l]$ but we will instead use the "Bulletproofs" inner product argument to send only $\sum_{k=1}^{l} \tilde{v}_k =< (\tilde{v}_1, \dots, \tilde{v}_l), \mathbf{1}^l >$ (denoted t' in the protocol) and then $\mathcal{P}$ and $\mathcal{V}$ engage in a "Bulletproofs" inner-product argument to prove that the committed values in basis g_k in the commitment $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} (h_1, \dots, h_l)^{\mathbf{1}^l} h^{-\mu'}$, which is $\sum_{j=1}^{m} v_k^{j+1} + s_k \cdot x' = \tilde{v}_k$, $k \in \{1, \dots, l\}$ (if the prover has followed honestly the protocol) satisfy an inner product equal to t'.
However, remember the "Bulletproofs" requires to have an input that is a power of 2 but it is not a problem if l is not a power of 2: we can simply pad the input to make it a power of 2 (we can do the same for the "Extended-Schnorr" argument used in the protocol).

Once, $\mathcal{P}$ has sent T_0 to $\mathcal{V}$ and convinced him that is a commitment over $\sum_{j=1}^{m} v_k^{(j)} z^{j+1}$, $\mathcal{V}$ can compute a commitment over C as $T_0 g^{\delta(y,z)}$ $(= h^{\sum_{j=1}^{m} \gamma_j z^{j+1}} g^{\delta(y,z)})$.

Back to $< A, B >= C$

Recall that we wanted the verifier to be able to check a verification equation of the form $< A, B >= C$.

$\mathcal{V}$ can now check the dot product equation we wanted to verify from the beginning, that was :

$$< \mathbf{a_L} - z \cdot \mathbf{1}^{n \cdot m}, \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n \cdot m}) + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} >= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y, z)$$

as $g^{<\mathbf{l},\mathbf{r}>} h^{\tau_x} = T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$, where $T_1 = h^{\tau_1} g^{t_1}$ and $h^{\tau_2} g^{t_2}$ are just commitments over t_1 and t_2 which are things to add due to the blinding terms.

Indeed, assuming $\mathbf{l}$ and $\mathbf{r}$ and T_0, T_1, T_2 are computed are honestly, $g^{<\mathbf{l},\mathbf{r}>} h^{\tau_x} = T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$ is equivalent to :

$$\begin{aligned} & g^{<\mathbf{a_L} - z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s_L} x, \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n \cdot m} + \mathbf{s_R} x) + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} >} h^{\tau_x} \\ &= h^{\tau_0} g^{\sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}} (h^{\tau_1} g^{t_1})^x (h^{\tau_2} g^{t_2})^{x^2} g^{\delta(y,z)} \\ &= h^{\tau_0 + \tau_1 x + \tau_2 x^2} g^{\sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y,z) + t_1 x + t_2 x^2} \end{aligned}$$

and hence equalizing the exponents of $\mathbf{g}$ (which we can except otherwise $\mathcal{P}$ has broken the *Discrete Log Relation*) we get :

$$<\mathbf{a_L} - z\cdot\mathbf{1}^{n\cdot m} + \mathbf{s_L}x, \mathbf{y}^{n\cdot m}\circ(\mathbf{a_R} + z\cdot\mathbf{1}^{n\cdot m} + \mathbf{s_R}x) + \sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} >$$

$$= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y,z) + t_1 x + t_2 x^2$$

$$\Leftrightarrow < \mathbf{a_L} - z\cdot\mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m}\circ(\mathbf{a_R} + z\cdot\mathbf{1}^{n\cdot m}) + \sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} >$$

$$+ \underbrace{< \mathbf{s_L}, \mathbf{y}^{n\cdot m}\circ(\mathbf{a_R} + z\mathbf{1}^{n\cdot m}) + \sum_{j=1}^{m} z^{1+j}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n}) > + < \mathbf{a_L} - z\mathbf{1}^{n\cdot m}, \mathbf{s_R}\circ\mathbf{y}^{n\cdot m} >}_{=t_1}\cdot x + \underbrace{< \mathbf{s_L}, \mathbf{y}^{n\cdot m}\circ\mathbf{s_R}}_{=t_2}\cdot x^2$$

$$= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y,z) + t_1 x + t_2 x^2$$

$$\Leftrightarrow < \mathbf{a_L} - z\cdot\mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m}\circ(\mathbf{a_R} + z\cdot\mathbf{1}^{n\cdot m}) + \sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} >= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y,z)$$

, t_1 and t_2 are here to remove the terms corresponding to the blinding terms in our equation and go back to our original single dot-product equation.

9.4 Final Protocol

We first give a formal written description of the protocol and then a diagram to see more visually the exchanges in the protocol.

K-selection protocol

- Input : $(g_1,\ldots,g_l,h_1,\ldots,h_l,h,g,u \in \mathbb{G}, \mathbf{g},\mathbf{h} \in \mathbb{G}^{n\cdot m}, \mathbf{V} = (V_j)_{j=1}^m \in \mathbb{G}^m; \{v_1^{(j)},\ldots,v_l^{(j)},\gamma_j\}_{j=1}^m \subseteq \mathbb{Z}_p^{l+1})$,

- $\mathcal{P}$'s input : $(g_1,\ldots,g_l,h_1,\ldots,h_l,g,h,u,\mathbf{g},\mathbf{h},\mathbf{V},\{v_1^{(j)},\ldots,v_l^{(j)},\gamma_j\}_{j=1}^m)$,
- $\mathcal{V}$'s input : $(g_1,\ldots,g_l,h_1,\ldots,h_l,g,h,u,\mathbf{g},\mathbf{h},\mathbf{V})$.

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 1 : $\mathcal{P}$ on input $\{v_1^{(j)},\ldots,v_l^{(j)},\gamma_j\}_{j=1}^m$ computes :

- $\mathbf{a_L^{(j)}} \in \mathbb{Z}_p^n : < \mathbf{a_L^{(j)}}, \mathbf{2}^n >= \sum_{k=1}^{l} v_k^{(j)}$, $j \in \{1,\ldots,m\}$, $\mathbf{a_L} = (\mathbf{a_L^{(1)}}||\ldots||\mathbf{a_L^{(m)}}) \in \mathbb{Z}_p^{n\cdot m}$.
- $\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^n$, $j \in \{1,\ldots,m\}$, $\mathbf{a_R} = (\mathbf{a_R^{(1)}}||\ldots||\mathbf{a_R^{(l)}}) \in \mathbb{Z}_p^{n\cdot m}$.
- Pick uniformly at random α,ρ in $\mathbb{Z}_p$ and $\mathbf{s_L},\mathbf{s_R}$ in $\mathbb{Z}_p^{n\cdot m}$.
- $A = h^{\alpha}\mathbf{g}^{\mathbf{a_L}}\mathbf{h}^{\mathbf{a_R}} \in \mathbb{G}$,
- $S = h^{\rho}\mathbf{g}^{\mathbf{s_L}}\mathbf{h}^{\mathbf{s_R}} \in \mathbb{G}$.

- step 2 : $\mathcal{P}$ sends to $\mathcal{V}$: A, S.
- step 3 : $\mathcal{V}$ picks uniformly at random $y,z \in \mathbb{Z}_p^*$.
- step 4 : $\mathcal{V}$ sends y,z to $\mathcal{P}$.

At this state of the proof we define $\mathbf{l}(X), \mathbf{r}(X) \in \mathbb{Z}_p^{n \cdot m}[X]$ and $t(X) \in \mathbb{Z}_p[X]$ as follows :

$$\mathbf{l}(X) := \mathbf{a_L} - z\mathbf{1}^{n \cdot m} + \mathbf{s_L}X \in \mathbb{Z}_p^{n \cdot m}[X],$$
$$\mathbf{r}(X) := \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m} + \mathbf{s_R}X) + \sum_{j=1}^{m} z^{1+j}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n}) \in \mathbb{Z}_p^{n \cdot m}[X],$$
$$t(X) :=< \mathbf{l}(X), \mathbf{r}(X) >= t_0 + t_1 X + t_2 X^2 \in \mathbb{Z}_p[X].$$

- step 5 : $\mathcal{P}$ computes :

- Pick uniformly at random τ_0, τ_1, τ_2 in $\mathbb{Z}_p$.
- Compute $t_0 := \sum_{j=1}^{m} \sum_{k=1}^{l} v_k^{(j)} z^{j+1}$ $(= t_0)$
- $t_1 =< \mathbf{s_L}, \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m}) + \sum_{j=1}^{m} z^{1+j}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n}) > + < \mathbf{a_L} - z\mathbf{1}^{n \cdot m}, \mathbf{s_R} \circ \mathbf{y}^{n \cdot m} >$,
- $t_2 =< \mathbf{s_L}, \mathbf{y}^{n \cdot m} \circ \mathbf{s_R} >$,
- $T_0 = h^{\tau_0} g^{t_0}$, $T_1 = h^{\tau_1} g^{t_1}$, $T_2 = h^{\tau_2} g^{t_2}$.

- step 6 : $\mathcal{P}$ sends to $\mathcal{V}$: T_0, T_1, T_2.
- step 7 : $\mathcal{V}$ picks uniformly at random x in $\mathbb{Z}_p^*$.
- step 8 : $\mathcal{V}$ sends x to $\mathcal{P}$.
- step 9 : $\mathcal{P}$ computes :

- $\mathbf{l} := \mathbf{l}(X) = \mathbf{a_L} - z\mathbf{1}^{n \cdot m} + \mathbf{s_L}x \in \mathbb{Z}_p^{n \cdot m}[X]$,
- $\mathbf{r} := \mathbf{r}(X) = \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m} + \mathbf{s_R}x) + \sum_{j=1}^{m} z^{1+j}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n}) \in \mathbb{Z}_p^{n \cdot m}$,
- $\hat{t} := t(x) =< \mathbf{l}, \mathbf{r} > \in \mathbb{Z}_p$,
- $\tau_x = \tau_2 x^2 + \tau_1 x + \tau_0 \in \mathbb{Z}_p$,
- $\mu = \alpha + \rho x \in \mathbb{Z}_p$.

- step 10 : $\mathcal{P}$ sends to $\mathcal{V}$: $\tau_x, \mu, \hat{t}$.
- step 11 : $\mathcal{P}$ and $\mathcal{V}$ compute:

- $\mathbf{h}'[i] = (\mathbf{h}[i])^{y^{-i+1}}$, $i \in \{1, \ldots, n \cdot m\}$, $\mathbf{h}' = (\mathbf{h}'[1], \ldots, \mathbf{h}'[n \cdot m])$,
- $P := AS^x \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h}'^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n}) z^{j+1}} h^{-\mu}$.

- step 12 : $\mathcal{V}$ checks the following verification equation : $g^{\hat{t}} h^{\tau_x} = T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$.
- step 13 : $\mathcal{P}$ and $\mathcal{V}$ engage in a "Bulletproofs" inner-product argument for the following relation :

$$R_5 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c =< \mathbf{a}, \mathbf{b} >\}$$

, on input :

- $\mathbf{g} := \mathbf{g} \in \mathbb{G}^{(m \cdot n)}$,
- $\mathbf{h} := \mathbf{h}' \in \mathbb{G}^{(m \cdot n)}$,
- $P := P \in \mathbb{G}$,
- $c := \hat{t} \in \mathbb{Z}_p$,
- $\mathbf{a} := \mathbf{l} \in \mathbb{Z}_p^{(n \cdot m)}$,

- $\mathbf{b} := \mathbf{r} \in \mathbb{Z}_p^{(n \cdot m)}$,

for $n = (n \cdot m)$.
- step 14 : $\mathcal{P}$ computes :

- Pick uniformly at random ρ' in $\mathbb{Z}_p^*$ and $\mathbf{s} = (s_1, \ldots, s_l)$ in $\mathbb{Z}_p^l$.
- $\overline{v}_i = \sum_{j=1}^{m} v_i^{(j)} z^{j+1} \in \mathbb{Z}_p$, $i \in \{1, \ldots, l\}$, $\overline{\mathbf{v}} = (\overline{v}_1, \ldots, \overline{v}_l)$.
- $S' = h^{\rho'} g_1^{s_1} \ldots g_l^{s_l}$.

- step 15 : $\mathcal{P}$ sends to $\mathcal{V}$: S'.
At this state of the proof we define $l'(X), r'(X) \in \mathbb{Z}_p^l[X]$ and $t'(X) \in \mathbb{Z}_p[X]$ as follows :

$$\begin{aligned} l'(X) &:= \overline{\mathbf{v}} + \mathbf{s}X \in \mathbb{Z}_p^l[X], \\ r'(X) &:= \mathbf{1}^l \in \mathbb{Z}_p^l[X], \\ t'(X) &:=< \mathbf{l}(X), \mathbf{r}(X) >= t_0 + t_1' X \in \mathbb{Z}_p[X]. \end{aligned}$$

- step 16 : $\mathcal{P}$ computes :

- Pick uniformly at random $\tau_1' \in \mathbb{Z}_p$.
- $t_1' =< \mathbf{s}, \mathbf{1}^l >$
- $T_1' = h^{\tau_1'} g^{t_1'}$.

- step 17 : $\mathcal{P}$ sends to $\mathcal{V}$: T_1'.
- step 18 : $\mathcal{V}$ picks uniformly at random x' in $\mathbb{Z}_p^*$.
- step 19 : $\mathcal{V}$ sends x' to $\mathcal{P}$.
- step 20 : $\mathcal{P}$ computes :

- $\mathbf{l}' := l'(x') = \overline{\mathbf{v}} + \mathbf{s}x' \in \mathbb{Z}_p^l$,
- $\mathbf{r}' := r'(x) = \mathbf{1}^l \in \mathbb{Z}_p^l$,
- $t' := t'(x') =< \mathbf{l}', \mathbf{r}' >= t_0 + t_1' x' \in \mathbb{Z}_p$,
- $\tau_{x'}' := \tau_0 + \tau_1' x' \in \mathbb{Z}_p$,
- $\mu' := \rho' x' + \sum_{j=1}^{m} \gamma_j z^{j+1} \in \mathbb{Z}_p$.

- step 21 : $\mathcal{P}$ sends to $\mathcal{V}$: $\mu', \tau_{x'}', t'$.
- step 22 : $\mathcal{V}$ checks the following verification equation : $g^{t'} h^{\tau_{x'}} = T_0 T_1'^{x'}$.
- step 23 : $\mathcal{P}$ and $\mathcal{V}$ engage in a "Bulletproofs" inner-product argument for the following relation :

$$R_5 \equiv \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c =< \mathbf{a}, \mathbf{b} >\}$$

, on input :

- $\mathbf{g} := (g_1, \ldots, g_l) \in \mathbb{G}^l$,
- $\mathbf{h} := (h_1, \ldots, h_l) \in \mathbb{G}^l$,
- $P := \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} (h_1, \ldots, h_l)^{\mathbf{1}^l} h^{-\mu'} \in \mathbb{G}$,
- $c := t' \in \mathbb{Z}_p$,
- $\mathbf{a} := \mathbf{l}' \in \mathbb{Z}_p^l$,
- $\mathbf{b} := \mathbf{r}' \in \mathbb{Z}_p^l$,

for $n = l$.
- step 24 : $\mathcal{P}$ and $\mathcal{V}$ engage in a "Extended-Schnorr Argument" for the following relation :

$$R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\}$$

, on input :

- $\mathbf{g} := (g_1, \ldots, g_l) \in \mathbb{G}^l$,
- $P := h^{-\mu'} \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} \in \mathbb{G}$,
- $\mathbf{a} := \mathbf{l}' \in \mathbb{Z}_p^l$,

for $n = l$.
- step 25 : $\mathcal{V}$ accepts the proof if the two verification equations, the "Extended-Schnorr Argument" and the "Bulletproofs" inner product argument hold.

K-Selection Protocol (part 1)

Prover		Verifier
$\mathbf{a_L^{(j)}} \in \mathbb{Z}_p^n$:		
$< \mathbf{a_L^{(j)}}, \mathbf{2}^n >= \sum_{k=1}^{l} v_k^{(j)}$		
$\mathbf{a_R^{(j)}} = \mathbf{a_L^{(j)}} - \mathbf{1}^n$		
$\alpha, \rho \xleftarrow{\$} \mathbb{Z}_p$		
$\mathbf{s_L}, \mathbf{s_R} \xleftarrow{\$} \mathbb{Z}_p^{n \cdot m}$		
$A := h^{\alpha} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}}$		
$S := h^{\rho} \mathbf{g}^{\mathbf{s_L}} \mathbf{h}^{\mathbf{s_R}}$		
	$\xrightarrow{A, S}$	
		$y, z \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{y, z}$	
$\tau_0, \tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p$		
$t_0 := \sum_{j=1}^{m} \sum_{k=1}^{l} v_k^{(j)} z^{j+1}$		
$T_0 = h^{\tau_0} g^{t_0}$		
$T_1 = h^{\tau_1} g^{t_1}$		
$T_2 = h^{\tau_2} g^{t_2}$		
	$\xrightarrow{T_0, T_1, T_2}$	
		$x \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{x}$	
$\mathbf{l} := \mathbf{a_L} - z\mathbf{1}^{n \cdot m} + \mathbf{s_L} x$		
$\mathbf{r} := \mathbf{y}^{n \cdot m} \circ (\mathbf{a_R} + z\mathbf{1}^{n \cdot m} + \mathbf{s_R} x) + \sum_{j=1}^{m} z^{1+j} (\mathbf{0}^{(j-1) \cdot n} \| \mathbf{2}^n \| \mathbf{0}^{(m-j) \cdot n})$		
$\hat{t} =< \mathbf{l}, \mathbf{r} >$		
$\tau_x = \tau_2 x^2 + \tau_1 x + \tau_0$		
$\mu = \alpha + \rho \cdot x$		
	$\xrightarrow{\tau_x, \mu, \hat{t}}$	
		$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$

K-Selection Protocol (part 2)

Prover		**Verifier**
$\mathbf{h}'[i] = (\mathbf{h}[i])^{y^{-i+1}}$		$\mathbf{h}'[i] = (\mathbf{h}[i])^{y^{-i+1}}$
$P := AS^x \mathbf{g}^{-z\cdot\mathbf{1}^{n\cdot m}} \cdot \mathbf{h}'^{z\cdot\mathbf{y}^{n\cdot m}+z^2\cdot\mathbf{z}^m} h^{-\mu}$		$P := AS^x \mathbf{g}^{-z\cdot\mathbf{1}^{n\cdot m}} \cdot \mathbf{h}'^{z\cdot\mathbf{y}^{n\cdot m}+z^2\cdot\mathbf{z}^m} h^{-\mu}$
		$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$
	$BP(\mathbf{g}, \mathbf{h}, P, \hat{t}; \mathbf{l}; \mathbf{r})$	
$\rho' \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$		
$\mathbf{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^l$		
$\overline{\mathbf{v}} = (\sum_{j=1}^{m} v_i^{(j)} z_i^{j+1})_{i=1}^{l}$		
$S' := h^{\rho'} (g_1 \dots g_l)^{\overline{\mathbf{v}}}$		
	S'	
$\tau_1' \stackrel{\$}{\leftarrow} \mathbb{Z}_p$		
$T_1' := h^{\tau_1'} g^{t_1'}$		
	T_1'	
		$x' \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$
	x'	
$\mathbf{l}' := \overline{\mathbf{v}} + \mathbf{s} \cdot x$		
$\mathbf{r}' := \mathbf{1}^l$		
$t' = < \mathbf{l}', \mathbf{r}' >$		
$\tau_{x'}' := \tau_0 + \tau_1' x'$		
$\mu' := \rho' \cdot x' + \sum_{j=1}^{m} \gamma_j z^{j+1}$		
	$\mu', \tau_{x'}', t'$	
		$g^{t'} h^{\tau_{x'}'} \stackrel{?}{=} T_0 T_1'^{x'}$
$\overline{P'} := h^{-\mu'} \mathbf{V}^{z^2\cdot\mathbf{z}^m} S'^{x'} \cdot (h_1, \dots, h_l)^{\mathbf{1}^l} h^{-\mu'}$		$\overline{P'} := h^{-\mu'} \mathbf{V}^{z^2\cdot\mathbf{z}^m} S'^{x'} \cdot (h_1, \dots, h_l)^{\mathbf{1}^l} h^{-\mu'}$
$\mathbf{g}' := (g_1, \dots, g_l)$		$\mathbf{g}' := (g_1, \dots, g_l)$
$\mathbf{h}' := (h_1, \dots, h_l)$		$\mathbf{h}' := (h_1, \dots, h_l)$
	$BP(\mathbf{g}', \mathbf{h}', \overline{P'}, t'; \mathbf{l}'; \mathbf{r}')$	
$\tilde{P}' := h^{-\mu'} \mathbf{V}^{z^2\cdot\mathbf{z}^m} S'^{x'}$		$\tilde{P}' := h^{-\mu'} \mathbf{V}^{z^2\cdot\mathbf{z}^m} S'^{x'}$
	$ES(\mathbf{g}', \tilde{P}'; \mathbf{a}')$	

9.5 Complexity

The K-selection proof protocol has the following complexities :

- **# Communications :** $2log_2(mn) + 4log_2(l) + 18$ elements :
  - $2log_2(mn) + 4log_2(l) + 7$ **Group elements :**
    - $A, S, T_0, T_1, T_2, S', T'_1$: 7 elements in $\mathbb{G}$
    - "Bulletproofs" inner-product argument with $n = mn$: $2log_2(mn)$ elements in $\mathbb{G}$
    - "Bulletproofs" inner-product argument with $n = l$:$2log_2(l)$ elements in $\mathbb{G}$
    - "Extended-Schnorr" argument with $n = l$: $2log_2(l)$ elements in $\mathbb{G}$
  - 11 $\mathbb{Z}_p$ **elements :**
    - $\tau_x, \mu, \hat{t}, \mu', \tau'_x, t'$: 6 elements in $\mathbb{Z}_p$
    - "Bulletproofs" inner-product argument with $n = nm$: 2 elements in $\mathbb{Z}_p$
    - "Bulletproofs" inner-product argument with $n = l$: 2 elements in $\mathbb{Z}_p$
    - "Extended-Schnorr" argument with $n = l$: 1 element in $\mathbb{Z}_p$
- **# Bases :** $2ml + 2l + 3$ bases:
  - $\mathbf{g_1}, \ldots, \mathbf{g_l}, \mathbf{h_1}, \ldots, \mathbf{h_l}$: $2ml$ bases in $\mathbb{G}$
  - $g_1, \ldots, g_l, h_1, \ldots, h_l$: $2l$ bases in $\mathbb{G}$
  - g, h, u : 3 bases in $\mathbb{G}$
- **# Exponentiations :**
  - **Prover :** $15mn + 13l + m + 4log_2(mn) + 6log_2(l) - 7$ exponentiations in $\mathbb{G}$
  - **Verifier :** $7mn + 6l + m + 2log_2(mn) + 4log_2(l) + 10$ exponentiations in $\mathbb{G}$

  The details are shown in Table 9.1.
- **# Exponentiation in $\mathbb{Z}_p$**:
  - **Prover:** $3log_2(mn) + 6log_2(l) + 9$ exponentiations in $\mathbb{Z}_p$
  - **Verifier:**$3log_2(mn) + 6log_2(l) + 7$ exponentiations in $\mathbb{Z}_p$

  The detail can be found in Table 9.1.
- **# Multiplications in $\mathbb{G}$:**
  - **Prover :** $9mn + 2log_2(mn) + 2m + 8l + 2log_2(l) - 4$ multiplications:
    - $2mn$ multiplications for each of A, S : $4mn$ multiplications
    - 1 multiplication for each of T_0, T_1, T_2 : 4 multiplications
    - P: $mn + 2$ multiplications
    - S' : l multiplications
    - T'_1 : 1 multiplications
    - the P input of the "Bulletproofs" inner-product argument with $n = l$: $2m + 1$ multiplications
    - "Bulletproofs" inner-product argument with $n = mn$: $4mn + 2log_2(mn) - 4$ multiplications
    - "Bulletproofs" inner-product argument with $n = l$: $4l + 2log_2(l) - 4$ multiplications
    - "Extended-Schnorr" argument with $n = l$: $3l - 3$ multiplications
  - **Verifier :** $3mn + 2m + 3l + 2log_2(l) + 8$ multiplications
    - P : $mn + 2$ multiplications
    - verification equation at step (**12**) : 4 multiplications
    - P input of the "Bulletproofs" inner-product argument with $n = l$: $2m + 1$ multiplications
    - verification equation at step (**22**) : 2 multiplications
    - "Bulletproofs" inner-product argument with $n = mn$: $2mn$ multiplications
    - "Bulletproofs" inner-product argument with $n =$: $2l$
    - "Extended-Schnorr" argument with $n = l$: $l + 2log_2(l) - 1$ multiplications

- **# Multiplications in $\mathbb{Z}_p$:**
  - **Prover :** $18mn + 2m + 10l - 11$ multiplications:
    - $\{z^{1+j}\}_{j=1}^{m}$: m multiplications
    - $\{y^j\}_{j=1}^{mn}$: $mn - 2$ multiplications
    - t_1 : $4mn$ multiplications
    - t_2 : $2mn$ multiplications
    - **l**: mn multiplications
    - **r**: $2mn$ multiplications
    - t_0 : mn multiplications
    - $\hat{t}$: mn multiplications
    - τ_x : 2 multiplications
    - μ: 1 multiplication
    - t_1' : l multiplications
    - l' : l multiplications
    - t': 1 multiplication
    - τ_x' : 1 multiplication
    - mu': m multiplications
    - "Bulletproofs" inner-product argument with $n = mn$: $6mn - 6$ multiplications
    - "Bulletproofs" inner-product argument with $n = l$: $6l - 6$ multiplications
    - "Extended-Schnorr" argument with $n = l$: $2l - 2$ multiplications
  - **Verifier :**

    $2mn + m$ multiplications:
    - P : mn multiplications
    - verification equation at step (**12**) : m multiplications
    - $\{y^j\}_{j=1}^{mn}$: $mn - 2$ multiplications
    - "Bulletproofs" inner-product argument with $n = mn$: 1 multiplication
    - "Bulletproofs" inner-product argument with $n = l$:1 multiplication

As desired the number of exponentiation and multiplications is linear in mn and the size of the elements exchanged scales logarithmically in mn.

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
A	2mn+1			
S	2mn+1			
$\{z_j\}_{j=1}^{m+2}$		1		1
y^{-1}		1		1
h'	mn		mn	
P	2mn+2		2mn+2	
T_0	2			
T_1	2			
T_2	2			
x^2		1		
S'	l+1			
T_1'	2			
P_2	m+1		m+2	
δ		1		1
Verif Eq 1			6	
Verif Eq 2			3	
BP(mn)	$8mn + 4log_2(mn) - 8$	$3log_2(mn) + 2$	$4mn + 2log_2(mn) - 1$	$3log_2(mn) + 2$
BP(l)	$8l + 4log_2(l) - 8$	$3log_2(l) + 2$	$4l + 2log_2(l) - 1$	$3log_2(l) + 2$
ES(l)	$4l + 2log_2(l) - 4$	$3log_2(l)$	$2l + 2log_2(l) - 2$	$3log_2(l)$
TOTAL	$15mn + 13l + m + 4log_2(mn)+6log_2(l)-7$	$3log_2(mn)+6log_2(l)+9$	$7mn + 6l + m + 2log_2(mn) + 4log_2(l) + 10$	$3log_2(mn)+6log_2(l)+7$

Table 9.1: Complexity of Min-Max K Selection Protocol

9.6 Theorem 7

The zero-knowledge protocol presented in section 10C for relation R_2 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.

In the followings subsections, we will show point by point that the K-selection protocol satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* for relation R_2.

9.6.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $\{\gamma_j, \mathbf{v^j}\}_{j=1}^m \subseteq \mathbb{Z}_p^{l+1}$ for the relation $R_2 \equiv \{(g_1, \ldots, g_l, h \in \mathbb{G}, \mathbf{V} = (V_1, \ldots, V_m) \in \mathbb{G}^m;$
$(v_1^{(1)}, \ldots, v_1^{(m)}), \ldots, (v_l^{(1)}, \ldots, v_l^{(m)}), \gamma = (\gamma_1, \ldots, \gamma_m) \in \mathbb{Z}_p^m), n \in \mathbb{N} : \left[V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}} \wedge \sum_{i=1}^{l} v_i \in [0, 2^n - 1]\right]\}$.
We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \in R_2\right] = 1$.

To see this, let's look at the verification equations that the verifier $\mathcal{V}$ checks in the protocol.

In order to show *Perfect Completeness*, assume a Prover $\mathcal{P}$ follows honestly the protocol knowing a valid witness $\{\gamma_j, \mathbf{v^j}\}_{j=1}^m : V_j = h^{\gamma_j} \mathbf{g}^{\mathbf{v^j}}$ and $< \mathbf{v^j}, \mathbf{1}^l > \in [0, 2^n - 1]$.

Then, $T_0 T_1^x T_2^{x^2} g^{\delta(y,z)} = (h^{\tau_0} g^{t_0})(h^{\tau_1} g^{t_1})^x (h^{\tau_2} g^{t_2})^{x^2} g^{\delta(y,z)} = h^{\tau_0 + \tau_1 x + \tau_2 x^2} g^{t_0 + t_1 x + t_2 x^2 + \delta(y,z)} = h^{\tau_x} g^{\hat{t}}$, since $\mathcal{P}$ follows honestly the protocol and has computed $\hat{t}$ and τ_x respectively as $\tau_x = \tau_0 + \tau_1 x + \tau_2 x^2$ and $\hat{t} = t_0 + t_1 x + t_2 x^2 + \delta(y, z)$. Hence, the verification equation at **-step 12 :** of the protocol will be verified.

The "Bulletproofs" inner product argument at **step 13 :** will be also accepted by the verifier.
Indeed, $AS^x \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu} =$

$(h^{\alpha}\mathbf{g}^{\mathbf{a_L}}\mathbf{h}^{\mathbf{a_R}})(h^{\rho}\mathbf{g}^{\mathbf{s_L}}\mathbf{h}^{\mathbf{s_R}})^{x}\mathbf{g}^{-z\cdot\mathbf{1}^{n\cdot m}}\mathbf{h'}^{z\cdot\mathbf{y}^{n\cdot m}+\sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^{n}||\mathbf{0}^{(m-j)\cdot n})z^{j+1}}h^{-\mu} =$
$h^{-\mu+[\alpha+\rho x]}\mathbf{g}^{\mathbf{a_L}-z\cdot\mathbf{1}^{n\cdot m}-\mathbf{s_L}\cdot x}\mathbf{h'}^{\mathbf{y}^{n\cdot m}\circ(\mathbf{a_R}+z\cdot\mathbf{1}^{n\cdot m}+\mathbf{s_L}\cdot x)+\sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^{n}||\mathbf{0}^{(m-j)\cdot n})} = \mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}}$, since $\mathcal{P}$ has committed $A = h^{\alpha}\mathbf{g}^{\mathbf{a_L}}\mathbf{h}^{\mathbf{a_R}}$, $S = h^{\rho}\mathbf{g}^{\mathbf{s_L}}\mathbf{h}^{\mathbf{s_R}}$ and computed $\mathbf{l},\mathbf{r}$ as $\mathbf{l} = \mathbf{a_L} - z\mathbf{1}^{n\cdot m} + \mathbf{s_L}x$ and $\mathbf{r} = \mathbf{y}^{n\cdot m}\circ(\mathbf{a_R} + z\mathbf{1}^{n\cdot m} + \mathbf{s_R}x) + \sum_{j=1}^{m} z^{1+j}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^{n}||\mathbf{0}^{(m-j)\cdot n})$. In addition, $\mathcal{P}$ has computed $\hat{t}$ as $\hat{t} =< \mathbf{l},\mathbf{r} >$.

Verification equation at **step 21 :** will be verified too.
Indeed, $T_0T_1'^{x'} = (h^{\tau_0}g^{t_0})(h^{\tau_1'}g^{t_1})^{x'} = h^{\tau_0+\tau_1'x'}g^{t_0+t_1'x'} = h^{\tau_{x'}'}g^{\hat{t}}$, since $\tau_{x'}' = \tau_0 + \tau_1'x'$ and $\hat{t} = t_0 + t_1'x'$.

Finally, the "Extended" Schnorr Argument" at **step 23 :** will be accepted since
$h^{-\mu'}\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'} = h^{-\mu}\prod_{j=1}^{m}\left[h^{\gamma_j}\mathbf{g'}^{\mathbf{v^j}}\right]^{z^{j+1}}(h^{\rho'}\mathbf{g'}^{\mathbf{s}})^{x'} = h^{-\mu'}h^{\sum_{j=1}^{m}\gamma_j z^{j+1}}\mathbf{g'}^{\sum_{j=1}^{m}\mathbf{v^j}z^{j+1}}h^{\rho'x'}\mathbf{g'}^{\mathbf{s}\cdot x'}$
$= h^{-\mu'+\left[\rho'x'+\sum_{j=1}^{m}\gamma_j z^{j+1}\right]}\mathbf{g'}^{\sum_{j=1}^{m}\mathbf{v^j}z^{j+1}+\mathbf{s}\cdot x'} = \mathbf{g'}^{\mathbf{l'}}$, where $\mathbf{g'} = (g_1,\ldots,g_l)$, since $\mathcal{P}$ has computed μ' and $\mathbf{l'}$ as , $\mu' = \rho'x' + \sum_{j=1}^{m}\gamma_j z^{j+1}$ and $\mathbf{l'} = \overline{\mathbf{v}} + \mathbf{s}x' = \sum_{j=1}^{m}\mathbf{v^j}z^{j+1}$ respectively.

Multiplying both sides of the previous equation $h^{-\mu'}\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'} = \mathbf{g'}^{\mathbf{l'}}$ by $\mathbf{h}^{\mathbf{1}^l}$ leads to $h^{-\mu'}\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'}\mathbf{h}^{\mathbf{1}^l} = \mathbf{g'}^{\mathbf{l'}}\mathbf{h}^{\mathbf{1}^l}$, which is equivalent to $h^{-\mu'}\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'}\mathbf{h}^{\mathbf{1}^l} = \mathbf{g'}^{\mathbf{l'}}\mathbf{h}^{\mathbf{r'}}$ since $\mathbf{r'} = \mathbf{1}^l$. In addition since $\mathcal{P}$ has computed t' as $t' =< \mathbf{l'},\mathbf{r'} >$ the "Bulletproofs" inner product argument at **step 22 :** will be accepted by the verifier.

9.6.2 Perfect Special Honest-Verifier Zero-Knowledge

To show *Perfect Special Honest-Verifier Zero-Knowledge* (SHVZK) we build explicitly an efficient simulator that given the C.R.S. and a statement ($\sigma := (g_1,\ldots,g_l,h) \in \mathbb{G}^{l+1}$ and $u := ((V_1,\ldots,V_m) \in \mathbb{G}^m, n \in \mathbb{N})$ respectively) produces indistinguishable transcript (in the sense that the transcript will have identical distribution) from a transcript resulting from the true interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. That is :

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

The simulator pseudo-code is as follows:

K-selection Proof Protocol Simulator

- **Input :** $(g_1,\ldots,g_l,h_1,\ldots,h_l,g,h \in \mathbb{G},\mathbf{g_1},\ldots,\mathbf{g_l},\mathbf{h_1},\ldots,\mathbf{h_l},\mathbf{V} = (V_1,\ldots,V_m))$.
- **Output :** transcript identically distributed to a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

- **step 1 :** Picks challenges x,y,z,x' uniformly at random in $\mathbb{Z}_p^*$.
- **step 2 :** Compute $\mathbf{h'}[i] = (\mathbf{h}[i])^{y^{-i+1}}$, $i \in \{1,\ldots,n\cdot m\}$, $\mathbf{h'} = (\mathbf{h'}[1],\ldots,\mathbf{h'}[n\cdot m])$.
- **step 3 :** Picks $\mu,\tau_x,\mu',\tau_{x'}'$ uniformly at random in $\mathbb{Z}_p$.
- **step 4 :** Picks $\mathbf{l},\mathbf{r}$ uniformly at random in $\mathbb{Z}_p^{n\cdot m}$ and $\mathbf{l'}$ uniformly at random in $\mathbb{Z}_p^l$, set $\mathbf{r'} := \mathbf{1}^l$.
- **step 5 :** Computes $\hat{t} =< \mathbf{l},\mathbf{r} >$ and $t' =< \mathbf{l'},\mathbf{r'} >$.
- **step 6 :** Picks A uniformly at random in $\mathbb{G}$.
- **step 7 :** Compute

$$S = \left(h^{\mu}\mathbf{g}^{\mathbf{l}}\mathbf{h'}^{\mathbf{r}}A^{-1}\mathbf{g}^{z\cdot\mathbf{1}^{n\cdot m}}\mathbf{h'}^{-z\cdot\mathbf{y}^{n\cdot m}-\sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^{n}||\mathbf{0}^{(m-j)\cdot n})z^{j+1}}\right)^{x^{-1}}$$

- **step 8 :** Compute

$$S' = \left(h^{\mu'}\mathbf{g'}^{\mathbf{l'}}\mathbf{V}^{-z^2\cdot\mathbf{z}^m}\right)^{x'^{-1}}$$

- **step 9 :** Given the witness $\mathbf{l},\mathbf{r}$ and $\mathbf{l'},\mathbf{r'}$ respectively run the protocol 2 to get transcript S_1^{inner} and S_2^{inner} of the first and second "Bulletproofs" inner-product argument.
- **step 10 :** Picks T_1,T_2 uniformly at random in $\mathbb{G}$.
- **step 11 :** Computes $T_0 = g^{-\delta(y,z)}T_1^{-x}T_2^{-x^2}g^{\hat{t}}h^{\tau_x}$.
- **step 12 :** Compute $T_1' = (g^{t'}h^{\tau_{x'}'}T_0^{-1})^{x'^{-1}}$.
- **step 13 :** Given the witness $\mathbf{l'}$ run the "Extended-Schnorr Argument" protocol to get transcript S^{ES} of

the "Extended-Schnorr Argument".
- step 14 : Output : $(A, S; y, z; T_0, T_1, T_2; x; \tau_x, \mu, \hat{t}; S_1^{inner}; S'; T_1'; x'; \mu', \tau_{x'}', t'; S_2^{inner}; S^{ES})$.

An honest prover interacting with an honest verifier will produce independent random elements $A, \mathbf{l}, \mathbf{r}, \mu, \tau_x, \mathbf{l}', \mu', \tau_{x'}'$ given $\alpha, \rho, \tau_0, \tau_1, \tau_2, x, y, z, \mathbf{s_L}, \mathbf{s_R}, \rho', \tau_1', x', \mathbf{s}$ are chosen independently and uniformly.
$\mathbf{r}' = \mathbf{1}^l$ as in any true interaction between honest $\mathcal{P}$ and $\mathcal{V}$.
The simulator produce $\hat{t} =< \mathbf{l}, \mathbf{r} >$ and $t' =< \mathbf{l}', \mathbf{r}' >$ as in any valid transcript.
The simulated S is fully defined by :
$AS^x\mathbf{g}^{-z\cdot\mathbf{1}^{n\cdot m}}\mathbf{h}'^{z\cdot\mathbf{y}^{n\cdot m}+\sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1}}h^{-\mu} = \mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}}$, which is ensured by computing S accordingly.
Given that $\mathcal{P}$ follows the protocol honestly, T_0, T_1, T_2 are perfectly hiding Pedersen commitments and thus are random group elements. Given $\hat{t}$ and answer τ_x, their internal relation is fully defined by the equation $g^{\hat{t}}h^{\tau_x} = T_0T_1^xT_2^{x^2}g^{\delta(y,z)}$, which is ensured by computing T_0 accordingly.

The simulated S' and T_1' are fully defined by : $h^{-\mu'}\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'} = \mathbf{g}'^{\mathbf{l}'}$ and $T_0T_1'^{x'} = g^{t'}h^{\tau_{x'}'}$ respectively, which is ensured by computing S' and T_1' accordingly.

S_1^{inner} is produced by the simulator knowing actually the true witness for this sub-protocol: $\mathbf{l}, \mathbf{r}$ such that $\hat{t} =< \mathbf{l}, \mathbf{r} >$ and $AS^x\mathbf{g}^{-z\cdot\mathbf{1}^{n\cdot m}}\mathbf{h}'^{z\cdot\mathbf{y}^{n\cdot m}+\sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1}}h^{-\mu} = \mathbf{g}^{\mathbf{l}}\mathbf{h}^{\mathbf{r}}$ so the transcript produced for the first "Bulletproofs" inner product argument inside the protocol will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

S_2^{inner} is produced by the simulator knowing actually the true witness for this sub-protocol: $\mathbf{l}', \mathbf{r}'$ such that $t' =< \mathbf{l}', \mathbf{r}' >$ and $A\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'}\mathbf{h}^{\mathbf{1}^l}h^{-\mu'} = \mathbf{g}'^{\mathbf{l}'}\mathbf{h}'^{\mathbf{r}'}$ so the transcript produced for the second "Bulletproofs" inner product argument inside the protocol will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

S^{ES} is produced by the simulator knowing actually the true witness for this sub-protocol: $\mathbf{l}'$ such that $A\mathbf{V}^{z^2\cdot\mathbf{z}^m}S'^{x'}h^{-\mu'} = \mathbf{g}'^{\mathbf{l}'}$ so the transcript produced for the "Extended-Schnorr" argument inside the protocol will be indistinguishable from the one resulting from a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

We can thus conclude that the transcript obtained by the simulator pseudo-code is distributed identically to the transcript of a true protocol interaction between honest $\mathcal{P}$ and $\mathcal{V}$ with independently uniformly selected challenges. This shows our protocol proof has the *Perfect Special Honest-Verifier Zero-Knowledge*.

9.6.3 Computational Witness Extended-Emulation

In this section we construct an extractor χ that uses χ^{BP} and χ^{ES}, the extractors of the "Bulletproofs" inner-product argument and of the "Extended-Schnorr" argument respectively as subroutines and a polynomial number of $N^{KS} := 2 \cdot 3 \cdot (m+2) \cdot mn \cdot N^{BP}(l) \cdot N^{ES}(l) \cdot N^{BP}(mn) = O(nm^2N^{BP}(l)N^{ES}(l)N^{BP}(mn)) = O(n^3m^4l^{3.58})$ transcripts with 2 different challenges $x' : x_1', x_2'$, $m+2$ different challenges $z : \{z_i\}_{i=1}^{m+2}$, 3 different challenges $x : x_1, x_2, x_3$, mn different challenges $y : \{y_i\}_{i=1}^{n\cdot m}$, $N^{BP}(mn)$ different sub-transcripts to run the "Bulletproofs" inner product argument extractor for the first "Bulletproofs" inner-product argument on $n = mn$, $N^{BP}(l)$ for the second one with $n = l$ and finally N^{ES} sub-transcripts to run the "Extended-Schnorr" argument extractor for the "Extended-Schnorr" argument with $n = l$ in order to extract a valid witness $\gamma = (\gamma_1, \ldots, \gamma_m)$, $\mathbf{v} = (\mathbf{v^1}||\ldots||\mathbf{v^m})$ such that $V_j = h^{\gamma_j}\mathbf{g}^{\mathbf{v^j}}$ and $< \mathbf{v^j}, \mathbf{1}^l >\in [0, 2^n - 1]$ $\forall j \in [1, m]$.

This allows to prove, using the Forking Lemma 3.4, the *Computational Witness Extended Emulation* 3.3.4 property.

The following proof may seem a bit long and intricate but it is not very complicated. All that is required is to build the extractor χ.
Informally the extractor will work as follows: χ first uses the "Bulletproofs" inner product argument extractor, χ^{BP}, to extract the witnesses $\mathbf{l}'$ and $\mathbf{r}'$ corresponding to the second "Bulletproofs" inner product argument with $n = l$.
Using the "Extended-Schnorr" argument extractor, χ^{ES} we prove that $\mathbf{r}' = \mathbf{1}^l$ except with negligible probability.
We use also use the "Bulletproofs" inner product argument extractor, χ^{BP}, to extract the witnesses $\mathbf{l}$ and $\mathbf{r}$ corresponding to the second "Bulletproofs" inner product argument with $n = mn$.
Then we combine different transcripts to open the commitments S', $\{V_j\}_{j=1}^m$ (which leads to a candidate witness

for relation R_2), T_1', T_0, S, A and T_1.
We then use the verification equations, the fact that the dot product equations associated with the "Bulletproofs" arguments and the extracted committed values must be valid to show that the candidate witness extracted from the commitments $\{V_j\}_{j=1}^m$ satisfy indeed the relation R_2, i.e. is a valid witness.

In what follows, $\mathbf{g}'$ denote $(g_1, \ldots, g_l) \in \mathbb{G}^l$.

Using the "Bulletproofs" inner product argument extractor, χ^{BP}, we can extract $\mathbf{l}', \mathbf{r}' \in \mathbb{Z}_p^l$ such that $\mathbf{g'}^{\mathbf{l}'}\mathbf{h'}^{\mathbf{r}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} \mathbf{h}^{\mathbf{1}^l} h^{-\mu'}$ and $< \mathbf{l}', \mathbf{r}' >= t'$.
Using the "Extended-Schnorr Argument", χ^{ES}, we can extract a witness $\tilde{\mathbf{l}}' \in \mathbb{Z}_p : \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} h^{-\mu'} = \mathbf{g'}^{\tilde{\mathbf{l}}'}$.
Injecting it back in the previous equation leads to $\mathbf{g'}^{\mathbf{l}'}\mathbf{h'}^{\mathbf{r}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} \mathbf{h}^{\mathbf{1}^l} h^{-\mu'} = \mathbf{h}^{\mathbf{1}^l} \mathbf{g'}^{\tilde{\mathbf{l}}'}$.
Using the *Discrete Log Relation* assumption we can infer : $\mathbf{l}' = \tilde{\mathbf{l}}'$ and $\mathbf{r}' = \mathbf{1}^l$.
It implies that $t' =< \mathbf{l}', \mathbf{1}^l >$ and $\mathbf{g'}^{\mathbf{l}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} h^{-\mu'}$.

Consider two valid transcripts with two different challenges $x' : x_1', x_2'$, we have : $\mathbf{g'}^{\mathbf{l}_\mathbf{i}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x_i'} h^{-\mu_i'}$, $i \in \{1, 2\}$. Let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^2 \nu_i = 0$ and $\sum_{i=1}^2 \nu_i x_i' = 1$. We combine these two equations to open the commitments S' and $\mathbf{V}^{z^2 \cdot \mathbf{z}^m}$ and extract their committed values as follows.
We can make a linear combination in the exponents of the two previous equations to obtain:
$\prod_{i=1}^2 \left[\mathbf{g'}^{\mathbf{l}_\mathbf{i}'}\right]^{\nu_i} = \prod_{i=1}^2 \left[\mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x_i'} h^{-\mu_i'}\right]^{\nu_i} \Leftrightarrow \mathbf{g'}^{\sum_{i=1}^2 \nu_i \mathbf{l}_\mathbf{i}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m \cdot \sum_{i=1}^2 \nu_i} S'^{\sum_{i=1}^2 \nu_i x_i'} h^{-\sum_{i=1}^2 \nu_i \mu_i'} = S' h^{-\sum_{i=1}^2 \nu_i \mu_i'}$.
Hence, $S' = h^{\rho'} \mathbf{g'}^{\mathbf{s}'}$, where $\mathbf{s}' = \sum_{i=1}^2 \nu_i \mathbf{l}_\mathbf{i}'$ and $\rho' = \sum_{i=1}^2 \nu_i \mu_i'$.

Now, consider $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^2 \nu_i = 1$ and $\sum_{i=1}^2 \nu_i x_i = 0$ then We can make a linear combination in the exponents of the two previous equations to obtain:
$\prod_{i=1}^2 \left[\mathbf{g'}^{\mathbf{l}_\mathbf{i}'}\right]^{\nu_i} = \prod_{i=1}^2 \left[\mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x_i'} h^{-\mu_i'}\right]^{\nu_i} \Leftrightarrow \mathbf{g'}^{\sum_{i=1}^2 \nu_i \mathbf{l}_\mathbf{i}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m \cdot \sum_{i=1}^2 \nu_i} S'^{\sum_{i=1}^2 \nu_i x_i'} h^{-\sum_{i=1}^2 \nu_i \mu_i'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} h^{-\sum_{i=1}^2 \nu_i \mu_i'}$.
Hence, $\mathbf{V}^{z^2 \cdot \mathbf{z}^m} = h^{\overline{\gamma}} \mathbf{g'}^{\overline{\mathbf{v}}}$, where $\overline{\mathbf{v}} = \sum_{i=1}^2 \nu_i \mathbf{l}_\mathbf{i}'$ and $\overline{\gamma} = \sum_{i=1}^2 \nu_i \mu_i'$.

Now, given y, x, consider m transcripts with different challenges $z : \{z_j\}_{j=1}^m$. From our previous discussion, we can write $\mathbf{V}^{z_j^2 \cdot \mathbf{z_j}^m} = h^{\overline{\gamma}_j} \mathbf{g'}^{\overline{\mathbf{v}}_j} = h^{\overline{\gamma}_j} g_1^{\overline{v}_1^{(j)}} \ldots g_l^{\overline{v}_l^{(j)}}$, with $(\overline{v}_1^{(j)}, \ldots, \overline{v}_l^{(j)}) = \overline{\mathbf{v}}_j$.
Given $j \in \{1, \ldots, m\}$, let $\{\nu_i\}_{i=1}^m \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^m \nu_i z_i^{1+j} = 1$ and $\sum_{i=1}^m \nu_i z_i^{1+k} = 0 \ \forall k \in (\{1, \ldots, m\} \backslash \{j\})$.
In other words, $\{\nu_i\}_{i=1}^m$ are such that given $k \in \{1, \ldots, m\} : \sum_{i=1}^m z_i^{k+1} \nu_i = \Gamma(k, j) := \begin{cases} 1, \text{ if } j = k, \\ 0, \text{ otherwise.} \end{cases}$
In what follows below, we denote V_j as the j-th component of the m-dimensional vector $\mathbf{V}$, i.e. $\mathbf{V} = (V_j)_{j=1}^m = (V_1, \ldots, V_m)$.
Then, we can compute the product $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i}$ as :

- $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^m \left(\prod_{k=1}^m V_k^{z_i^{k+1}}\right)^{\nu_i} = \prod_{k=1}^m \left[\prod_{i=1}^m \left(V_k^{z_i^{k+1}}\right)^{\nu_i}\right] = \prod_{k=1}^m \left(V_k^{\sum_{i=1}^m z_i^{k+1} \nu_i}\right)$
  $= \prod_{k=1}^m V_k^{\Gamma(k,j)} = V_j^{\Gamma(j,j)} = V_j$, since $V_k^{\Gamma(k,j)} = 1 \ \forall k \neq j$.

- $\prod_{i=1}^m \left(\mathbf{V}^{z_i^2 \mathbf{z_i}^m}\right)^{\nu_i} = \prod_{i=1}^m \left(h^{\gamma_i} g_1^{v_1^{(i)}} \ldots g_l^{v_l^{(i)}}\right)^{\nu_i} = h^{\sum_{i=1}^m \gamma_i \nu_i} g_1^{\sum_{i=1}^m v_1^{(i)} \nu_i} \ldots g_l^{\sum_{i=1}^m v_l^{(i)} \nu_i}$

Hence given any $j \in \{1, \ldots, m\}$, we can write $V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \ldots g_l^{v_l^{(j)}}$, where $\gamma_j = \sum_{i=1}^m \overline{\gamma}_i \nu_i$ and
$v_k^{(j)} = \sum_{i=1}^m \overline{v}_k^{(i)} \nu_i$, $k \in \{1, \ldots, l\}$.

So far, we have managed to extract a candidate witness $\{\gamma_j, \mathbf{v^j} := (v_1^{(j)}, \ldots, v_l^{(j)})\}_{j=1}^m$ such that $V_j = h^{\gamma_j} \mathbf{g'}^{\mathbf{v^j}}$.
It remains to show that it is a valid witness by showing that $< \mathbf{v^j}, \mathbf{1}^l > \in [0, 2^n - 1]$, $j \in [1, l]$.

We can inject back the expressions we just found for S' and $\{V_j\}_{j=1}^m$ in the equation $\mathbf{g'}^{\mathbf{l}'} = \mathbf{V}^{z^2 \cdot \mathbf{z}^m} S'^{x'} h^{-\mu'}$.
We obtain in this way :
$\mathbf{g'}^{\mathbf{l}'} = \prod_{j=1}^m \left[h^{\gamma_j} \mathbf{g'}^{\mathbf{v^j}}\right]^{z^{j+1}} (h^{\rho'} \mathbf{g'}^{\mathbf{s}'})^{x'} h^{-\mu'} = h^{\sum_{j=1}^m \gamma_j z^{j+1}} \mathbf{g'}^{\sum_{j=1}^m \mathbf{v^j} z^{j+1}} h^{\rho' x'} \mathbf{g'}^{\mathbf{s}' x'} h^{-\mu'}$
$= h^{-\mu' + \left[\sum_{j=1}^m \gamma_j z^{j+1} + \rho' x'\right]} \mathbf{g'}^{\sum_{j=1}^m \mathbf{v^j} z^{j+1} + \mathbf{s}' x'}$. Using the *Discrete Log Relation* assumption we can thus infer the following except with negligible probability :

- $\mu' = \sum_{j=1}^m \gamma_j z^{j+1} + \rho' x'$,
- $\mathbf{l}' = \sum_{j=1}^m \mathbf{v^j} z^{j+1} + \mathbf{s}' x'$.

Hence, $t' =< \mathbf{l}', \mathbf{1}^l >= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + < \mathbf{s}', \mathbf{1}^l > x'$.

Now, given z consider two valid transcripts with different challenges $x' : x'_1, x'_2$ and the verification equation : $g^{t'} h^{\tau'_{x'}} = T_0 T_1'^{x'}$ that must hold for these two transcripts.

$\implies g^{t'_i} h^{\tau'_{x'_i}} = T_0 T_1'^{x'_i}$, $i \in \{1, 2\}$

We will as usual combine these equations in order to open the commitments T_0 and T'_1 and extract their committed values.

For this, consider $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 0$ and $\sum_{i=1}^{2} \nu_i x'_i = 1$, then : $\prod_{i=1}^{2} \left[g^{t'_i} h^{\tau'_{x'_i}} \right]^{\nu_i} = \prod_{i=1}^{2} \left[T_0 T_1'^{x'_i} \right]^{\nu_i}$

$\Leftrightarrow g^{\sum_{i=1}^{2} t'_i \nu_i} h^{\sum_{i=1}^{2} \tau'_{x'_i} \nu_i} = T_0^{\sum_{i=1}^{2} \nu_i} T_1'^{\sum_{i=1}^{2} x'_i \nu_i} = T_0^0 T_1'^1 = T'_1$.

Hence, $T'_1 = h^{\tau'_1} g^{t'_1}$, with $\tau'_1 = \sum_{i=1}^{2} \tau'_{x'_i} \nu_i$ and $t'_1 = \sum_{i=1}^{2} t'_i \nu_i$.

Now similarly consider $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 1$ and $\sum_{i=1}^{2} \nu_i x'_i = 0$, then : $\prod_{i=1}^{2} \left[g^{t'_i} h^{\tau'_{x'_i}} \right]^{\nu_i} = \prod_{i=1}^{2} \left[T_0 T_1'^{x'_i} \right]^{\nu_i}$

$\Leftrightarrow g^{\sum_{i=1}^{2} t'_i \nu_i} h^{\sum_{i=1}^{2} \tau'_{x'_i} \nu_i} = T_0^{\sum_{i=1}^{2} \nu_i} T_1'^{\sum_{i=1}^{2} x'_i \nu_i} = T_0^1 T_1'^0 = T_0$.

Hence, $T_0 = h^{\tau_0} g^{t_0}$, with $\tau_0 = \sum_{i=1}^{2} \tau'_{x'_i} \nu_i$ and $t_0 = \sum_{i=1}^{2} t'_i \nu_i$.

Now, we can reinject our openings of the Pedersen commitments T_0, T'_1 in the verification equation $g^{t'} h^{\tau'_{x'}} = T_0 T_1'^{x'}$.

This leads to : $g^{t'} h^{\tau'_{x'}} = T_0 T_1'^{x'} = h^{\tau_0} g^{t_0} (h^{\tau'_1} g^{t'_1})^{x'} = h^{\tau_0 + \tau'_1 x'} g^{t_0 + t'_1 x'}$. Using the *Discrete Log Relation* assumption we can infer the followings except with negligible probability :

- $\tau'_{x'} = \tau_0 + \tau'_1 x'$,
- $t' = t_0 + t'_1 x'$.

We thus have : $t' = t_0 + t'_1 x' = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + < \mathbf{s}, \mathbf{1}^l > x'$. This must hold $\forall x' \in \{x'_1, x'_2\}$. That is : $t'_i = t_0 + t'_1 x'_i = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + < \mathbf{s}, \mathbf{1}^l > x'_i$, $i \in \{1, 2\}$.

Hence, let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 1$, $\sum_{i=1}^{2} \nu_i x'_i = 0$. Then, we can make a linear combination of the two previous equations to obtain:

$\sum_{i=1}^{2} [t_0 + t'_1 x'_i] \cdot \nu_i = \sum_{i=1}^{2} \left[\sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + < \mathbf{s}, \mathbf{1}^l > x'_i \right] \cdot \nu_i$

$\Leftrightarrow t_0 \cdot (\sum_{i=1}^{2} \nu_i) + t'_1 \cdot (\sum_{i=1}^{2} \nu_i x'_i) = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} \cdot (\sum_{i=1}^{2} \nu_i) + < \mathbf{s}, \mathbf{1}^l > \cdot (\sum_{i=1}^{2} \nu_i x'_i)$

$\Leftrightarrow t_0 \cdot 1 + t'_1 \cdot 0 = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} \cdot 1 + < \mathbf{s}, \mathbf{1}^l > \cdot 0$

$\Leftrightarrow t_0 = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}$.

Now, using the "Bulletproofs" inner product argument extractor we can extract witness $\mathbf{l}, \mathbf{r} \in \mathbb{Z}_p^{(n \cdot m)}$:

$\mathbf{g^l h'^r} = P = A S^x \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu}$ and $< \mathbf{l}, \mathbf{r} >= \hat{t}$.

Consider two valid transcripts with same challenge y, z but with two different challenges $x : x_1, x_2$, then we have : $P_i = \mathbf{g^{l_i} h'^{r_i}} = A S^{x_i} \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu_i}$.

Let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 0$, $\sum_{i=1}^{2} \nu_i x_i = 1$. Then,

$\prod_{i=1}^{2} [P_i]^{\nu_i} = \prod_{i=1}^{2} \left[\mathbf{g^{l_i} h'^{r_i}} \right]^{\nu_i} = \prod_{i=1}^{2} \left[A S^{x_i} \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu_i} \right]^{\nu_i}$

$\Leftrightarrow \mathbf{g}^{\sum_{i=1}^{2} \mathbf{l_i} \nu_i} \mathbf{h'}^{\sum_{i=1}^{2} \mathbf{r}_i \nu_i} = A^{\sum_{i=1}^{2} \nu_i} S^{\sum_{i=1}^{2} x_i \nu_i} \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m} \cdot \sum_{i=1}^{2} \nu_i} \mathbf{h'}^{\left[z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} \right] \cdot \sum_{i=1}^{2} \nu_i} h^{-\sum_{i=1}^{2} \nu_i \mu_i} =$

$A^0 S^1 \mathbf{g}^{\mathbf{0}^{n \cdot m}} \mathbf{h'}^{\mathbf{0}^{n \cdot m}} h^{-\sum_{i=1}^{2} \nu_i \mu_i} = S h^{-\sum_{i=1}^{2} \nu_i \mu_i}$.

Hence, we can write $S = h^{\rho} \mathbf{g}^{\mathbf{s_L}} \mathbf{h}^{\mathbf{s_R}}$, where $\rho = \sum_{i=1}^{2} \nu_i \mu_i$, $\mathbf{s_L} = \sum_{i=1}^{2} \mathbf{l_i} \nu_i$ and $\mathbf{s_R} = \mathbf{y}^{n \cdot m} \circ \sum_{i=1}^{2} \mathbf{r}_i \nu_i$.

Similarly, let $\nu_1, \nu_2 \in \mathbb{Z}_p : \sum_{i=1}^{2} \nu_i = 1$, $\sum_{i=1}^{2} \nu_i x_i = 0$. Then,

$\prod_{i=1}^{2} [P_i]^{\nu_i} = \prod_{i=1}^{2} \left[\mathbf{g^{l_i} h'^{r_i}} \right]^{\nu_i} = \prod_{i=1}^{2} \left[A S^{x_i} \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu_i} \right]^{\nu_i}$

$\Leftrightarrow \mathbf{g}^{\sum_{i=1}^{2} \mathbf{l_i} \nu_i} \mathbf{h'}^{\sum_{i=1}^{2} \mathbf{r}_i \nu_i} = A^{\sum_{i=1}^{2} \nu_i} S^{\sum_{i=1}^{2} x_i \nu_i} \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m} \cdot \sum_{i=1}^{2} \nu_i} \mathbf{h'}^{\left[z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} \right] \cdot \sum_{i=1}^{2} \nu_i} h^{-\sum_{i=1}^{2} \nu_i \mu_i} =$

$A \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{\left[z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1} \right]} h^{-\sum_{i=1}^{2} \nu_i \mu_i}$.

Hence, we can write $A = h^{\alpha} \mathbf{g}^{\mathbf{a_L}} \mathbf{h}^{\mathbf{a_R}}$, where $\alpha = \sum_{i=1}^{2} \nu_i \mu_i$, $\mathbf{a_L} = \sum_{i=1}^{2} \mathbf{l_i} \nu_i + z \cdot \mathbf{1}^{n \cdot m}$ and $\mathbf{a_R} = \mathbf{y}^{-n \cdot m} \circ (\sum_{i=1}^{2} \mathbf{r}_i \nu_i - z \cdot \mathbf{y}^{n \cdot m} - \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1})$.

Now, we can inject back in the equation $\mathbf{g^l h'^r} = A S^x \mathbf{g}^{-z \cdot \mathbf{1}^{n \cdot m}} \mathbf{h'}^{z \cdot \mathbf{y}^{n \cdot m} + \sum_{j=1}^{m} (\mathbf{0}^{(j-1) \cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j) \cdot n}) z^{j+1}} h^{-\mu}$ the two expressions we just found for A and S to obtain :

$\mathbf{g^l h'^r} = (h^\alpha \mathbf{g^{a_L} h^{a_R}})(h^\rho \mathbf{g^{s_L} h^{s_R}})^x \mathbf{g}^{-z\cdot \mathbf{1}^{n\cdot m}} \mathbf{h'}^{z\cdot \mathbf{y}^{n\cdot m} + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1}} h^{-\mu} =$
$h^\alpha \mathbf{g^{a_L}} \mathbf{h'}^{\mathbf{y}^{n\cdot m} \circ \mathbf{a_R}} h^{\rho x} \mathbf{g^{s_L}}^{\cdot x} \mathbf{h'}^{(\mathbf{y}^{n\cdot m} \circ \mathbf{s_R})\cdot x} \mathbf{g}^{-z\cdot \mathbf{1}^{n\cdot m}} \mathbf{h'}^{z\cdot \mathbf{y}^{n\cdot m} + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1}} h^{-\mu} =$
$h^{[\alpha+\rho x]-\mu} \mathbf{g^{a_L + s_L}}\cdot x - z\cdot \mathbf{1}^{n\cdot m} \mathbf{h'}^{\mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + \mathbf{s_R} x + z\cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}}$.
Hence, using the *Discrete Log Relation* assumption, we can infer the followings except with negligible probability :

- $\mu = \alpha + \rho x$,
- $\mathbf{l} = \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m} + \mathbf{s_L} \cdot x$,
- $\mathbf{r} = \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m} + \mathbf{s_R} x) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1}$.

Now, consider the verification equation $g^{\hat{t}} h^{\tau_x} = T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$ and three valid transcripts with same challenges y, z but with different challenges x : x_1, x_2, x_3, then the verification equation must hold, that is : $g^{\hat{t}_i} h^{\tau_{x_i}} = T_0 T_1^{x_i} T_2^{x_i^2} g^{\delta(y,z)}$, $i \in \{1, 2, 3\}$.
Then, we make linear combinations of these equations in the exponent once again to open the commitments T_1, T_2 as follows.
Let $\nu_1, \nu_2, \nu_3 \in \mathbb{Z}_p$: $\sum_{i=1}^3 \nu_i = 0$, $\sum_{i=1}^3 \nu_i x_i = 1$ and $\sum_{i=1}^3 \nu_i x_i^2 = 0$.
Then,
$\prod_{i=1}^3 \left[g^{\hat{t}_i} h^{\tau_{x_i}} \right]^{\nu_i} = \prod_{i=1}^3 \left[T_0 T_1^{x_i} T_2^{x_i^2} g^{\delta(y,z)} \right]^{\nu_i} \Leftrightarrow g^{\sum_{i=1}^3 \hat{t}_i \nu_i} h^{\sum_{i=1}^3 \tau_{x_i} \nu_i} = T_0^{\sum_{i=1}^3 \nu_i} T_1^{\sum_{i=1}^3 x_i \nu_i} T_2^{\sum_{i=1}^3 x_i^2 \nu_i} g^{\delta(y,z) \cdot \sum_{i=1}^3 \nu_i}$
$\Leftrightarrow g^{\sum_{i=1}^3 \hat{t}_i \nu_i} h^{\sum_{i=1}^3 \tau_{x_i} \nu_i} = T_1$.
Hence, we can write $T_1 = h^{\tau_1} g^{t_1}$, where $\tau_1 = \sum_{i=1}^3 \tau_{x_i} \nu_i$ and $t_1 = \sum_{i=1}^3 \hat{t}_i \nu_i$.

Similarly, let $\nu_1, \nu_2, \nu_3 \in \mathbb{Z}_p$: $\sum_{i=1}^3 \nu_i = 0$, $\sum_{i=1}^3 \nu_i x_i = 0$ and $\sum_{i=1}^3 \nu_i x_i^2 = 1$.
Then,
$\prod_{i=1}^3 \left[g^{\hat{t}_i} h^{\tau_{x_i}} \right]^{\nu_i} = \prod_{i=1}^3 \left[T_0 T_1^{x_i} T_2^{x_i^2} g^{\delta(y,z)} \right]^{\nu_i} \Leftrightarrow g^{\sum_{i=1}^3 \hat{t}_i \nu_i} h^{\sum_{i=1}^3 \tau_{x_i} \nu_i} = T_0^{\sum_{i=1}^3 \nu_i} T_1^{\sum_{i=1}^3 x_i \nu_i} T_2^{\sum_{i=1}^3 x_i^2 \nu_i} g^{\delta(y,z) \cdot \sum_{i=1}^3 \nu_i}$
$\Leftrightarrow g^{\sum_{i=1}^3 \hat{t}_i \nu_i} h^{\sum_{i=1}^3 \tau_{x_i} \nu_i} = T_2$.
Hence, we can write $T_2 = h^{\tau_2} g^{t_2}$, where $\tau_2 = \sum_{i=1}^3 \tau_{x_i} \nu_i$ and $t_2 = \sum_{i=1}^3 \hat{t}_i \nu_i$.

Recall that we have already done an opening of the commitment T_0 : $T_0 = h^{\tau_0} g^{t_0}$.
Now, we can reinject our openings of T_0, T_1 and T_2 in the equation $g^{\hat{t}} h^{\tau_x} = T_0 T_1^x T_2^{x^2} g^{\delta(y,z)}$.
This leads to $g^{\hat{t}} h^{\tau_x} = (h^{\tau_0} g^{t_0})(h^{\tau_1} g^{t_1})^x (h^{\tau_2} g^{t_2})^{x^2} g^{\delta(y,z)} = h^{\tau_0 + \tau_1 x + \tau_2 x^2} g^{(t_0 + \delta(y,z)) + t_1 x + t_2 x^2}$.
Now, using the *Discrete Log Relation* assumption we can infer the followings except with negligible probability :

- $\tau_x = \tau_0 + \tau_1 x + \tau_2 x^2$,
- $\hat{t} = (t_0 + \delta(y,z)) + t_1 x + t_2 x^2$.

Now, recall that $\hat{t} =< \mathbf{l}, \mathbf{r} >=< \mathbf{a_L} - z\cdot \mathbf{1}^{n\cdot m} + \mathbf{s_L}\cdot x, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z\cdot \mathbf{1}^{n\cdot m} + \mathbf{s_R} x) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1} >= \tilde{t}_0 + \tilde{t}_1 x + \tilde{t}_2 x^2$, where $\tilde{t}_0 :=< \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1} >$, $\tilde{t}_1 =< \mathbf{s_L}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1} > + < \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ \mathbf{s_R}) >$ and $\tilde{t}_2 =< \mathbf{s_L}, \mathbf{y}^{n\cdot m} \circ \mathbf{s_R} >$.

Hence, $\hat{t} = (t_0 + \delta(y,z)) + t_1 x + t_2 x^2 = \tilde{t}_0 + \tilde{t}_1 x + \tilde{t}_2 x^2$. This equation must hold for all $x \in \{x_1, x_2, x_3\}$.
Or just written differently : $\hat{t}_i = t_0 + \delta(y,z) + t_1 x_i + t_2 x_i^2 = \tilde{t}_0 + \tilde{t}_1 x_i + \tilde{t}_2 x_i^2$, $i \in \{1, 2, 3\}$.
Now, consider $\nu_1, \nu_2, \nu_3 \in \mathbb{Z}_p$: $\sum_{i=1}^3 \nu_i = 1$, $\sum_{i=1}^3 \nu_i x_i = 0$ and $\sum_{i=1}^3 \nu_i x_i^2 = 0$.
Then, $\sum_{i=1}^3 \left[\hat{t}_i \right] \cdot \nu_i = \sum_{i=1}^3 \left[(t_0 + \delta(y,z)) + t_1 x_i + t_2 x_i^2 \right] \cdot \nu_i = \sum_{i=1}^3 \left[\tilde{t}_0 + \tilde{t}_1 x_i + \tilde{t}_2 x_i^2 \right] \cdot \nu_i$
$\Leftrightarrow (t_0 + \delta(y,z)) \cdot \sum_{i=1}^3 \nu_i + t_1 \cdot \sum_{i=1}^3 \nu_i x_i + t_2 \cdot \sum_{i=1}^3 \nu_i x_i^2 = \tilde{t}_0 \cdot \sum_{i=1}^3 \nu_i + \tilde{t}_1 \cdot \sum_{i=1}^3 \nu_i x_i + \tilde{t}_2 \cdot \sum_{i=1}^3 \nu_i x_i^2$
$\Leftrightarrow (t_0 + \delta(y,z)) \cdot 1 + t_1 \cdot 0 + t_2 \cdot 0 = \tilde{t}_0 \cdot 1 + \tilde{t}_1 \cdot 0 + \tilde{t}_2 \cdot 0 \Leftrightarrow t_0 + \delta(y,z) = \tilde{t}_0$.

Now remember that we have also seen : $t_0 = \sum_{j=1}^m < \mathbf{v^j}, \mathbf{1}^l > z^{j+1}$. Reinjecting this equality and $\tilde{t}_0 =< \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1} >$ in $t_0 + \delta(y,z) = \tilde{t}_0$ leads to : $\sum_{j=1}^m < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y,z) =< \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^m (\mathbf{0}^{(j-1)\cdot n} || \mathbf{2}^n || \mathbf{0}^{(m-j)\cdot n}) z^{j+1} >$.

Using the proof of appendix B it is equivalent to:
$\sum_{j=1}^m (< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >) z^{j+1} + < \mathbf{a_L} - \mathbf{1}^{n\cdot m} - \mathbf{a_R}, \mathbf{y}^{n\cdot m} > z + < \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n\cdot m} >= 0$.
This equation must hold for all $z \in \{z_i\}_{i=1}^{m+2}$. We have a system of $m+2$ equations :
$\sum_{j=1}^m (< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >) z_i^{j+1} + < \mathbf{a_L} - \mathbf{1}^{n\cdot m} - \mathbf{a_R}, \mathbf{y}^{n\cdot m} > z_i + < \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n\cdot m} >= 0$, $i \in \{1, \ldots, m+2\}$.

We can write it equivalently as $\sum_{j=1}^{m+2} d_j z_i^{j-1} = 0,\ i \in \{1,\dots,m+2\}$, for $d_1 =< \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n\cdot m} >$, $d_2 = \mathbf{1}^{n\cdot m} - \mathbf{a_R}, \mathbf{y}^{n\cdot m} >$ and $d_{j+2} = (< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >)$.

Now, given $k \in \{1,\dots,m+2\}$, let $\{\nu_i\}_{i=1}^{m+2} \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^{m+2} \nu_i z_i^{k-1} = 1$ and $\sum_{i=1}^{m+2} \nu_i z_i^{j-1} = 0$ $\forall j \in (\{1,\dots,m+2\}\backslash\{k\})$. In other words, $\{\nu_i\}_{i=1}^{m}$ are such that given $k \in \{1,\dots,m+2\}$: $\sum_{i=1}^{m+2} z_i^{j-1}\nu_i = \Gamma(j,k) := \begin{cases} 1, \text{ if } k = j, \\ 0, \text{ otherwise.} \end{cases}$
Then, we can compute : $\sum_{i=1}^{m+2} \left(\sum_{j=1}^{m+2} d_j z_i^{j-1}\right) \cdot \nu_i = \sum_{j=1}^{m+2} \left(\sum_{i=1}^{m+2} z_i^{j-1}\nu_i\right) \cdot d_j = \sum_{j=1}^{m+2} \Gamma(j,k) d_j = d_k = 0$.
Since this must hold for any $k \in \{1,\dots,m+2\}$, this implies that all d_k ($k \in \{1,\dots,k+2\}$) are zeros. Substituting back the expression of the $\{d_k\}_{k=1}^{m+2}$ this is :

- $< \mathbf{a_L^{(j)}}, \mathbf{2}^n >=< \mathbf{v^j}, \mathbf{1}^l >,\ j \in \{1,\dots,m\}$, **(1)**
- $< \mathbf{a_L} - \mathbf{1}^{n\cdot m} - \mathbf{a_R}, \mathbf{y}^{n\cdot m} >= 0$, **(2)**
- $< \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n\cdot m} >= 0$. **(3)**

Let's denote in the rest of this section $\mathbf{a_L}[k], \mathbf{a_R}[k]$ as the k-th component of the $n\cdot m$ vector $\mathbf{a_L}$ and $\mathbf{a_R}$ respectively.

Then, we can write equation **(2)** equivalently as $\sum_{k=1}^{m\cdot n}(\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k])y^{k-1} = 0$.
This equation must hold for all $y \in \{y_i\}_{i=1}^{n\cdot m}$. It can be rewritten as : $\sum_{k=1}^{m\cdot n}(\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k])y_i^{k-1} = 0$, $i \in \{1,\dots,n\cdot m\}$ **(4)**.
Now, given $j \in \{1,\dots,m+2\}$, let $\{\nu_i\}_{i=1}^{n\cdot m} \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^{n\cdot m} \nu_i y_i^{j-1} = 1$ and $\sum_{i=1}^{n\cdot m} \nu_i y_i^{k-1} = 0\ \forall k \in (\{1,\dots,n\cdot m\}\backslash\{k\})$. In other words, $\{\nu_i\}_{i=1}^{n\cdot m}$ are such that given $j \in \{1,\dots,n\cdot m\}$: $\sum_{i=1}^{n\cdot m} y_i^{k-1}\nu_i = \Gamma(k,j) := \begin{cases} 1, \text{ if } j = k, \\ 0, \text{ otherwise.} \end{cases}$
Then, we take linear combinations of the $n\cdot m$ equations **(4)** as follows :
$\sum_{i=1}^{m\cdot n}\left(\sum_{k=1}^{m\cdot n}(\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k])y_i^{k-1}\right)\cdot\nu_i = \sum_{k=1}^{m\cdot n}\left(\sum_{i=1}^{m\cdot n} y_i^{k-1}\nu_i\right)\cdot(\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k]) =$
$\sum_{k=1}^{m\cdot n}\Gamma(k,j)\cdot(\mathbf{a_L}[k] - 1 - \mathbf{a_R}[k]) = \mathbf{a_L}[j] - 1 - \mathbf{a_R}[j] = 0$.
Hence since the above reasoning must hold for any $j \in \{1,\dots,m\cdot n\}$, **(2)** implies : $\mathbf{a_L}[j] - 1 - \mathbf{a_R}[j] = 0\ \forall j \in [1, m\cdot n]$ or equivalently $\mathbf{a_L} - \mathbf{a_R} - \mathbf{1}^{m\cdot n} = \mathbf{0}^{n\cdot m}$.

We will do the same reasoning to show $\mathbf{a_L} \circ \mathbf{a_R} = \mathbf{0}^{n\cdot m}$.
For this, we can write **(3)** equivalently as $\sum_{k=1}^{m\cdot n} \mathbf{a_L}[k]\cdot\mathbf{a_R}[k]y^{k-1} = 0$.
This equation must hold for all $y \in \{y_i\}_{i=1}^{n\cdot m}$. It can be rewritten as : $\sum_{k=1}^{m\cdot n} \mathbf{a_L}[k]\cdot\mathbf{a_R}[k]y_i^{k-1} = 0$, $i \in \{1,\dots,n\cdot m\}$ **(5)**.
Once again, given $j \in \{1,\dots,m+2\}$, let $\{\nu_i\}_{i=1}^{n\cdot m} \subset \mathbb{Z}_p$ such that : $\sum_{i=1}^{n\cdot m} \nu_i y_i^{k-1} = \Gamma(k,j) = \begin{cases} 1, \text{ if } j = k, \\ 0, \text{ otherwise.} \end{cases}$
Then, we take linear combinations of the $n\cdot m$ equations **(5)** as follows :
$\sum_{i=1}^{m\cdot n}\left(\sum_{k=1}^{m\cdot n} \mathbf{a_L}[k]\cdot\mathbf{a_R}[k]y_i^{k-1}\right)\cdot\nu_i = \sum_{k=1}^{m\cdot n}\left(\sum_{i=1}^{m\cdot n} y_i^{k-1}\nu_i\right)\cdot(\mathbf{a_L}[k]\cdot\mathbf{a_R}[k]) =$
$\sum_{k=1}^{m\cdot n}\Gamma(k,j)\cdot(\mathbf{a_L}[k]\cdot\mathbf{a_R}[k]) = \mathbf{a_L}[j]\cdot\mathbf{a_R}[j] = 0$.
Hence since the above reasoning must hold for any $j \in \{1,\dots,m\cdot n\}$, **(3)** implies : $\mathbf{a_L}[j]\cdot\mathbf{a_R}[j] = 0\ \forall j \in [1, m\cdot n]$ or equivalently $\mathbf{a_L} \circ \mathbf{a_R} = \mathbf{0}^{n\cdot m}$.

Now, we can conclude this proof because we have extracted $\{\gamma_j, \mathbf{v^j}\}_{j=1}^{m} \subset \mathbb{Z}_p \times \mathbb{Z}_p^l$: $V_j = h^{\gamma_j}\mathbf{g}^{\mathbf{v^j}}\ \forall j \in [1,m]$ and we have also extracted $(\mathbf{a_L}, \mathbf{a_R}) \in \mathbb{Z}_p^{n\cdot m} \times \mathbb{Z}_p^{n\cdot m}$ that satisfy :

- $< \mathbf{v^j}, \mathbf{1}^l >=< \mathbf{a_L^{(j)}}, \mathbf{2}^n >,\ \forall j \in [1,m]$,
- $\mathbf{a_R} = \mathbf{a_L} - \mathbf{1}^{n\cdot m}$,
- $\mathbf{a_L} \circ \mathbf{a_R} = \mathbf{0}^{n\cdot m}$,

where $(\mathbf{a_L^{(1)}}||\dots||\mathbf{a_L^{(m)}}) = \mathbf{a_L}$.
The first two equations implies that $\mathbf{a_L^{(j)}} \in \{0,1\}^n$ and thus the first one implies $< \mathbf{v^j}, \mathbf{1}^l >\in [0, 2^n - 1]$, that is $\{\gamma_j, \mathbf{v^j}\}_{j=1}^{m}$ is a valid witness for relation :
$R_2 \equiv \{(g_1,\dots,g_l, h \in \mathbb{G}, V = (V_1,\dots,V_m) \in \mathbb{G}^m; \{\gamma_j, \mathbf{v^j}\}_{j=1}^{m} \subseteq \mathbb{Z}_p \times \mathbb{Z}_p^l) : \left[V_j = h^{\gamma_j} g_1^{v_1^{(j)}} \dots g_l^{v_l^{(j)}} \wedge < \mathbf{v^j}, \mathbf{1}^l >\in [0, 2^n - 1]\right]\}$.

Using the *Forking Lemma* presented in section 3.4 this implies the *Computational Witness Extended Emulation* property (3.3.4) which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The K-selection protocol is in $6 + (2log_2(mn) + 4log_2(l) + 1) = 2log_2(mn) + 4log_2(l) + 7$ moves and has $3 + log_2(mn) + 2log_2(l)$ challenge $(y, z), x, x', log_2(mn)$ challenges for the first "Bulletproofs" inner product argument on input $n = mn$, $log_2(l)$ for the second "Bulletproofs" inner product argument on input $n = l$ and $log_2(l)$ for the "Extended-Schnorr" argument on input $n = l$.

Hence the parameter μ in the statement of the *Forking Lemma* is equal to $3 + log_2(mn) + 2log_2(l)$. the corresponding tree of accepting transcripts has depth equal to $\mu = 3 + log_2(mn) + 2log_2(l)$. The node in the first level has $n_1 = nm^2 + 2mn$ children in the second level of the tree corresponding to the $mn(m + 2)$ pairs of challenges (y, z) for the first challenge $(\{y_j\}_{j=1}^{m \cdot n} \times \{z_j\}_{j=1}^{m+2})$. Each node in the second level has $n_2 = 3$ children each corresponding to a different value of the challenge x $(\{x_1, x_2, x_3\})$. Each node in the second level has $n_3 = 2$ children corresponding to the 2 different challenges x' (x'_1, x'_2). Each node in the third level of our tree is connected to the root of a tree of accepting transcripts for the first "Bulletproofs" inner product argument on input $n = mn$, which is a tree of depth $log_2(mn)$ with 4 children per node (see the "Bulletproofs" paper), hence $n_i = 4$, $4 \leq i \leq 4 + log_2(mn)$.

Then, each node of the last round of the first "Bulletproofs" inner product argument is connected to a tree of accepting transcripts for the second "Bulletproofs" inner product argument on input $n = l$, which is a tree of depth $log_2(l)$ with 4 children per node (see the "Bulletproofs" paper), hence $n_i = 4$, $5 + log_2(mn) \leq i \leq 5 + log_2(ml) + log_2(l)$.

Finally, each node of the last round of the second "Bulletproofs" inner product argument is connected to a tree of accepting transcripts for the "Extended-Schnorr" argument on input $n = l$, which is a tree of depth $log_2(l)$ with 3 children per node (see the "Bulletproofs" paper), hence $n_i = 3$, $5 + log_2(mn) + log_2(l) \leq i \leq 5 + log_2(mn) + 2log_2(l)$.

Therefore, the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = \prod_{i=1}^{3+log_2(mn)+2log_2(l)} n_i = n_1 \cdot n_2 \cdot n_3 \cdot \prod_{i=4}^{3+log_2(mn)} n_i \cdot \prod_{i=4+log_2(mn)}^{4+log_2(mn)+log_2(l)} n_i \cdot \prod_{i=5+log_2(mn)+log_2(l)}^{5+log_2(mn)+2log_2(l)} n_i = m(m + 2)n \cdot 3 \cdot 2 \cdot 4^{log_2(mn)} \cdot 4^{log_2(l)} \cdot 3^{log_2(l)} = N^{KS} = O(m^2 n(mn)^2 l^2 l^{1.58}) = O(n^3 m^4 l^{3.58})$, which is clearly bounded above by a polynomial in the parameter λ. In addition, χ is polynomial time since computing the ν_i amounts each time to solving a system of linear equations in $\mathbb{Z}_p$ for example for the opening of the commitment A : $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ in $\mathbb{Z}_p$, which can be done efficiently. All other operations are also basic polynomial time computations. Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute the ν_i to open a commitment, which occurs only when the determinant of the matrix is 0 which happens only with negligible probability. For example for the opening of the commitment A, the matrix corresponding to the linear system of equation in $\mathbb{Z}_p$ is $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$. The determinant is $x_1 - x_2$. It is zero when $x_1 = x_2$, which occurs with negligible probability $(1/p)$ since x_1 and x_2 are picked uniformly at random.

Chapter 10

Partial Opening

In this section, we create a protocol that allows doing partial opening on votes commitment.
That is given a commitment of the form $V = h^{\gamma} g_1^{v_1} \dots g_l^{v_l}$ for a vote $\mathbf{v} = (v_1, \dots, v_l)$, we create a protocol that allows proving that the j-th selection vote on the ballot: v_j, is equal to a certain bit $b \in \{0, 1\}$.
The prover knowing $\gamma, \{v_i\}_{i=1}^{l}$ (the committed values inside V) sends v_j for a particular value $j \in \{1, \dots, m\}$ and convinces him that V has the form $V = h^{\gamma} g_1^{v_1} \dots g_j^{v_j} \dots g_l^{v_l}$.
We want to do this in a zero-knowledge fashion, the verifier must only learn the values v_j and not the others v_i for $i \neq j$.

Notice that if we divide V by g^{v_j}, we obtain $V/g^{v_j} = h^{\gamma} \prod_{i \neq j} g_i^{v_i}$, that is a commitment that is only in basis h and g_i, $i \in (\{1, \dots, l\} \backslash \{j\})$.
The idea of the proof is that $\mathcal{P}$ proofs $V = h^{\gamma} g_1^{v_1} \dots g_j^{v_j} \dots g_l^{v_l}$ by zero-knowledge proving that he knows $(\gamma, v_1, \dots, v_l)$ such that $V/g^{v_j} = h^{\gamma} \prod_{i \neq j} g_i^{v_i}$. This can be done efficiently using our "Extended-Schnorr Argument". However, we want our protocol to be zero-knowledge and we do not want to leak any information about the witness $(\gamma, v_1, \dots, v_l)$, while the "Extended-Schnorr" argument does not have the zero-knowledge property, so we will blind these values in our protocol with random blinding factors $\mathbf{s} \in \mathbb{Z}_p^{l-1}$.

10.1 Protocol

The following zero-knowledge argument protocol allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that he knows a valid witness for the following relation :

$$R_3 \equiv \{(g_1, \dots, g_l, h \in \mathbb{G}, j \in \{1, \dots, l\}, b \in \{0, 1\}, V \in \mathbb{G}; \gamma \in \mathbb{Z}_p, (v_1, \dots, v_l) \in \mathbb{Z}_p^{l-1}) : V = h^{\gamma} g_1^{v_1} \dots g_l^{v_l} \wedge v_j = b\}.$$

For the Partial Opening protocol, the C.R.S. is $\sigma := (g_1, \dots, g_l, h)$, the statement is $u := (j, b, V)$ and the witness is $w := (\gamma, v_1, \dots, v_l)$.

We first give a formal written description of the protocol and then a diagram to see more visually the exchanges in the protocol.

10.1.1 Formal Description

Partial Opening Protocol

- Input : $(g_1, \dots, g_l, h, V \in \mathbb{G}, j \in \{1, \dots, l\}, b \in \{0, 1\}; v_1, \dots, v_l, \gamma \in \mathbb{Z}_p)$,

- $\mathcal{P}$'s input : $(g_1, \dots, g_l, h, V, j, b; v_1, \dots, v_l, \gamma)$,
- $\mathcal{V}$'s input : $(g_1, \dots, g_l, h, V, j, b)$.

- Output : $\mathcal{V}$ accepts or rejects the proof.

- step 0 : $\mathcal{P}$ and $\mathcal{V}$ (pre)compute $\mathbf{g} := (g_1, \dots, g_{j-1}, g_{j+1}, \dots, g_l) \in \mathbb{Z}_p^{l-1}$.
- step 1 : $\mathcal{P}$ on input $\{\mathbf{v} = (v_1, \dots, v_l), \gamma\}$ computes :

- $\tilde{\mathbf{v}} := (v_1, \dots, v_{j-1}, v_{j+1}, \dots, v_l)$.

- Pick uniformly at random α in $\mathbb{Z}_p$ and $\mathbf{s} \in \mathbb{Z}_p^{l-1}$
- $S := h^{\alpha}\mathbf{g}^{\mathbf{s}}$.

- step 2 : $\mathcal{P}$ sends S to $\mathcal{V}$.
- step 3 : $\mathcal{V}$ picks x uniformly at random in $\mathbb{Z}_p$.
- step 4 : $\mathcal{V}$ sends x to $\mathcal{P}$.
- step 5 : $\mathcal{P}$ computes :

- $\mu := \gamma + \alpha x \in \mathbb{Z}_p$,
- $\mathbf{b} := \tilde{\mathbf{v}} + \mathbf{s}x$.

- step 6 : $\mathcal{P}$ sends μ to $\mathcal{V}$.
- step 7 : $\mathcal{P}$ and $\mathcal{V}$ compute : $P := h^{-\mu}S^{x}Vg_j^{-b}$.
- step 8 : $\mathcal{P}$ and $\mathcal{V}$ engage in an "Extended-Schnorr Argument" for the following relation :

$$R_4 \equiv \{(\mathbf{g} \in \mathbb{G}^n, P \in \mathbb{G}; \mathbf{a}, \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}}\}$$

, on input :

- $\mathbf{g} := \mathbf{g} \in \mathbb{G}^{l-1}$,
- $P := P \in \mathbb{G}$,
- $\mathbf{a} := \mathbf{b} \in \mathbb{Z}_p^{l-1}$,

for $n = l - 1$.
- step 9 : $\mathcal{V}$ accepts the proof if the "Extended-Schnorr Argument" is valid.

N.B.: We assume the Prover has first sent v_j to the Verifier before the protocol begins.
Recall that the "Extended-Schnorr" argument requires to have an input with a size that is a power of 2 so we need to pad the input if $l - 1$ is not a power of 2.

10.1.2 Diagram

Partial Opening Protocol

Prover		Verifier
$\mathbf{g} := (g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_l)$		$\mathbf{g} := (g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_l)$
$\tilde{\mathbf{v}} := (v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_l)$		
$\alpha \xleftarrow{\$} \mathbb{Z}_p$		
$\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^{l-1}$		
$S := h^{\alpha}\mathbf{g}^{\mathbf{s}}$		
	$S \longrightarrow$	
		$x \xleftarrow{\$} \mathbb{Z}_p$
	$\longleftarrow x$	
$\mu := \gamma + \alpha \cdot x$		
$\mathbf{b} := \tilde{\mathbf{v}} + \mathbf{s} \cdot x$		
	$\mu \longrightarrow$	
$P := h^{-\mu} S^x V g_j^{-b}$		$P := h^{-\mu} S^x V g_j^{-b}$
	$\longleftarrow ES(\mathbf{g}, P; \mathbf{b})$	

10.2 Complexity

The Partial Opening protocol has the following complexities :

- **# Communications :** $2log_2(l-1)+3$ elements :
  - $2log_2(l-1)+1$ **Group elements:**
    - S: 1 element in $\mathbb{G}$
    - "Extended-Schnorr" argument with $n = l-1$: $2log_2(l-1)$ elements in $\mathbb{G}$
  - 2 $\mathbb{Z}_p$ **elements :**
    - μ : 1 element in $\mathbb{Z}_p$
    - "Extended-Schnorr" argument with $n = l-1$: 1 element in $\mathbb{Z}_p$
- **# Bases :** $2ml + l$ bases in $\mathbb{G}$:
  - $\mathbf{g_1}, \ldots, \mathbf{g_l}, \mathbf{h_1}, \ldots, \mathbf{h_l}$: 2ml bases
  - $g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_l$: l-1 bases
  - h : 1 base
- **# Exponentiations :**
  - **# Prover :** $5l + 2log_2(l-1) - 5$ exponentiations in $\mathbb{G}$
  - **# Verifier :** $2l + 2log_2(l-1)$ exponentiations in $\mathbb{G}$

  The details are shown in Table 10.1.
- **# Exponentiations in $\mathbb{Z}_p$:**
  - **Prover:** $3log_2(l-1)$ exponentiations in $\mathbb{Z}_p$

  - **Verifier:** $3log_2(l-1)$ exponentiations in $\mathbb{Z}_p$

  The detail can be found in Table 10.1.

- **# Multiplications in $\mathbb{G}$:**
  - **# Prover :** $4l-1$ multiplications
    - S : $l-1$ multiplications
    - P : 3 multiplications
    - "Extended-Schnorr" argument with $n = l-1$: $3l-3$ multiplications
  - **# Verifier :** $l + 2log_2(l-1) + 1$ multiplications:
    - P :3 multiplications
    - "Extended-Schnorr" argument with $n = l-1$: $l + 2log_2(l-1) - 2$ multiplications
- **# Multiplications in $\mathbb{Z}_p$:**
  - **# Prover :** $3l-5$ multiplications
    - μ : 1 multiplication
    - $\mathbf{b}$: $l-1$ multiplications
    - "Extended-Schnorr" argument with $n = l-1$: $2(l-1)-2$ multiplications
  - **# Verifier :** 0 multiplication

	Prover		Verifier	
	# Exp in $\mathbb{G}$	# Exp in Z_p	# Exp in $\mathbb{G}$	# Exp in Z_p
S	l			
P	3		3	
ES(l-1)	$4l + 2log_2(l-1) - 8$	$3log_2(l-1)$	$2l + 2log_2(l-1) - 3$	$3log_2(l-1)$
TOTAL	$5l + 2log_2(l-1) - 5$	$3log_2(l-1)$	$2l + 2log_2(l-1)$	$3log_2(l-1)$

Table 10.1: Complexity of Partial Opening Protocol

As desired the number of exponentiation and multiplications is linear in l and the size of the elements exchanged scales logarithmically in l.

10.3 Theorem 8

The zero-knowledge protocol presented in section 11A for relation R_3 has perfect completeness, perfect special honest-verifier zero-knowledge and computational witness extended emulation.

In the followings subsections, we will show point by point that the Schnorr's protocol satisfies the definitions of *Perfect Completeness*, *Perfect Special Honest-Verifier Zero-Knowledge* and *Computational Witness Extended Emulation* for relation R_3.

10.3.1 Perfect Completeness

In order to show *Perfect Completeness*, assume the prover $\mathcal{P}$ follows honestly the protocol knowing the valid witness $(\gamma, v_1, \ldots, v_l) \in \mathbb{Z}_p^l$ for the relation $R_3 \equiv \{(V, h, g_1, \ldots, g_l \in \mathbb{G}, j \in \{1, \ldots, l\}, b \in \{0,1\};$
$\gamma \in \mathbb{Z}_p, (v_1, \ldots, v_l) \in \mathbb{Z}_p^{l-1}) : V = h^\gamma g_1^{v_1} \ldots g_l^{v_l} \wedge v_j = b\}$.
We prove the interaction between $\mathcal{P}$ and $\mathcal{V}$ will result in an accepting conversation.
That is : $P\left[< \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) >= 1 | \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow \mathcal{A}(\sigma), (\sigma, u, w) \in R_3\right] = 1$.

The "Partial Opening" argument results in an accepting conversation as soon as the "Extended-Schnorr" argument is valid. We will thus prove that the verifier will accept the "Extended-Schnorr" argument to prove the protocol is complete.

To see this, notice that if $\tilde{\mathbf{v}} = (v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_l)$ and $V = h^\gamma \prod_{i=1}^{l} g_i^{v_i}$, then we have :
$V = g_j^b h^\gamma \prod_{i=1, i\neq j}^{l} g_i^{v_i} = g_j^b h^\gamma \mathbf{g}^{\tilde{\mathbf{v}}}$, or equivalently : $V g_j^{-b} = h^\gamma \mathbf{g}^{\tilde{\mathbf{v}}}$.
Therefore, $P = h^{-\mu} S^x V g_j^{-b} = h^{-(\gamma+\alpha x)} (h^\alpha \mathbf{g}^{\mathbf{s}})^x h^\gamma \mathbf{g}^{\tilde{\mathbf{v}}} = h^{\gamma+\alpha x-(\gamma+\alpha x)} \mathbf{g}^{\tilde{\mathbf{v}}+\mathbf{s}x} = h^0 \mathbf{g}^{\mathbf{b}} = \mathbf{g}^{\mathbf{b}}$, since $\mathcal{P}$ has computed μ and $\mathbf{b}$ as $\mu = \gamma + \alpha x$ and $\mathbf{b} = \tilde{\mathbf{v}} + \mathbf{s}x$.

Consequently, the "Extended-Schnorr Argument" interaction between the prover and the verifier at **step 8** will be accepted by the verifier since the "Extended-Schnorr Argument" protocol is complete. And the whole "Partial Opening" argument will be accepted by $\mathcal{V}$, which thereby concludes the proof of *Perfect Completeness*.

10.3.2 Perfect Special Honest-Verifier Zero-Knowledge

To show *Perfect Special Honest-Verifier Zero-Knowledge* (SHVZK) we build explicitly an efficient simulator that given the C.R.S. and a statement ($\sigma := (h, g_1, \ldots, g_l) \in \mathbb{G}^{l+1}$ and $u := (j \in \{1, \ldots, l\}, b \in \{0, 1\}, V \in \mathbb{G})$ respectively) produces indistinguishable transcript (in the sense that the transcript will have identical distribution) from a transcript resulting from the true interaction between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. That is :

$$P\left[\mathcal{A}(tr) = 1 \;\middle|\; \begin{array}{l} \sigma \leftarrow Setup(1^\lambda), \\ (u, w, \rho) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u, \rho) \rangle \end{array}\right] = P\left[\mathcal{A}(tr) = 1 \;\middle|\; \begin{array}{l} \sigma \leftarrow Setup(1^\lambda), \\ (u, w, \rho) \leftarrow \mathcal{A}(\sigma), \\ tr \leftarrow \mathcal{S}(\sigma, u, \rho) \end{array}\right].$$

The simulator pseudo-code is as follows :

Partial Opening Simulator

- Input : $(g, h, u, V, \mathbf{g}, \mathbf{h}, b)$.
- Output : transcript identically distributed to a true interaction between $\mathcal{P}$ and $\mathcal{V}$.

- step 0 : Compute $\mathbf{g} := (g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_l) \in \mathbb{Z}_p^{l-1}$.
- step 1 : Pick x, μ uniformly at random in $\mathbb{Z}_p^*$ and $\mathbf{b}$ uniformly at random in $\mathbb{Z}_p^{l-1}$.
- step 2 : Compute $S = \left(h^\mu V^{-1} g_j^b \mathbf{g}^{\mathbf{b}}\right)^{x^{-1}}$.
- step 3 : Given the witness $\mathbf{b}$ run the "Extended-Schnorr Argument" (using the verifier's randomness once again) protocol to get the transcript S^{ES}.
- step 4 : Output : $(S; x; \mu; S^{ES})$.

An honest prover interacting with an honest verifier will produce independent random-looking values $\mathbf{b}, \mu$ given x is chosen independently and randomly.
The simulated S is fully defined by : $h^\mu \mathbf{g}^{\mathbf{b}} = V g_j^{-b} S^x$, which is ensured by computing S accordingly.
S^{ES} is valid transcript of a true interaction between $\mathcal{P}$ and $\mathcal{V}$.
We can thus conclude that the transcript obtained with our simulator pseudo-code will be identically distributed to the transcript of a true protocol interaction between honest $\mathcal{P}$ and $\mathcal{V}$ with independently uniformly selected challenges. This shows our protocol proof has the *Perfect Special Honest-Verifier Zero-Knowledge* property.

10.3.3 Computational Witness Extended-Emulation

In this section, in order to prove the *Computational Witness Extended-Emulation*, we construct an efficient extractor χ that uses χ^{ES}, the extractor of the "Extended-Schnorr" argument as a subroutine and a polynomial number of $N^{PO} := 2 \cdot N^{ES}(l-1) = O(N^{ES}(l-1))$ transcripts: 2 transcripts for the challenges x and $N^{ES}(l-1)$ for the "Extended-Schnorr Argument" to extract a valid witness $(\gamma, v_1, \ldots, v_l)$ for relation $R_3 \equiv \{(h, g_1, \ldots, g_l \in \mathbb{G}, j \in \{1, \ldots, l\}, b \in \{0, 1\}, V \in \mathbb{G}; \gamma \in \mathbb{Z}_p, (v_1, \ldots, v_l) \in \mathbb{Z}_p^{l-1}) : V = h^\gamma g_1^{v_1} \ldots g_l^{v_l} \wedge v_j = b\}$.

This allows us to prove, using the Forking Lemma 3.4, the *Computational Witness Extended Emulation* 3.3.4 property.

Using the "Extended-Schnorr Argument" extractor χ_{ES}, we can extract a witness $\tilde{\mathbf{v}} = (\tilde{v}[1], \ldots, \tilde{v}[l-1])$ such that $\mathbf{g}^{\tilde{\mathbf{v}}} = P := h^{-\mu} g_j^{-b} V S^x$.

Now, consider two valid transcripts with different challenges $x : x_1, x_2 \in \mathbb{Z}_p$.
We have :

$$\mathbf{g}^{\tilde{\mathbf{v}}_\mathbf{i}} = P_i = h^{-\mu_i} g_j^{-b} V S^{x_i}, i \in \{1, 2\}.$$

Let $\nu_1, \nu_2 \in \mathbb{Z}_p$ such that $\sum_{i=1}^2 \nu_i = 1$ and $\sum_{i=1}^2 \nu_i x_i = 0$. Using these, we can compute : $\prod_{i=1}^2 [P_i]^{\nu_i}$ in two ways :

- $\prod_{i=1}^2 \left[h^{-\mu_i} g_j^{-b} V S^{x_i}\right]^{\nu_i} = h^{-\sum_{i=1}^2 \nu_i \mu_i} g_j^{-b \sum_{i=1}^2 \nu_i} V^{\sum_{i=1}^2 \nu_i} S^{\sum_{i=1}^2 \nu_i x_i} = h^{-\sum_{i=1}^2 \nu_i \mu_i} g_j^{-b} V^1 S^0 = h^{-\sum_{i=1}^2 \nu_i \mu_i} g_j^{-b} V$
- $\prod_{i=1}^2, \left[\mathbf{g}^{\tilde{\mathbf{v}}_\mathbf{i}}\right]^{\nu_i} = \mathbf{g}^{\sum_{i=1}^2 \nu_i \tilde{\mathbf{v}}_i}$.

Hence, since both sides are equal we must have: $h^{-\sum_{i=1}^2 \nu_i \mu_i} g_j^{-b} V = \mathbf{g}^{\sum_{i=1}^2 \nu_i \tilde{\mathbf{v}}_i} \Leftrightarrow V = h^{\sum_{i=1}^2 \nu_i \mu_i} g_j^b \mathbf{g}^{\sum_{i=1}^2 \nu_i \tilde{\mathbf{v}}_i}$.
We can thus write $V = h^\gamma g_1^{v_1} \ldots, g_l^{v_l}$, with $\gamma = \sum_{i=1}^2 \nu_i \mu_i$ and $v_k = \begin{cases} \sum_{i=1}^2 \nu_i \tilde{v}_i[k], \text{ if } k < j, \\ b, \text{ if } k = j, \\ \tilde{v}_i[k-1], \text{ if } k > j. \end{cases}$.
We have thus extracted a valid witness $v_1, \ldots, v_l$ such that $v_j = b$ and $V = h^\gamma g_1^{v_1} \ldots g_l^{v_l}$.

Using the *Forking Lemma* presented in section 3.4 this implies the *Computational Witness Extended Emulation* property (3.3.4) which concludes the proof.

If you want more details on why we can apply here the *Forking Lemma* here are the explanations.
The partial opening protocol is in $2 + (2log_2(l) + 1) = 3 + 2log_2(ml)$ moves and has $1 + log_2(l)$ challenges : x and the $log_2(l)$ challenges from the "Extended-Schnorr" argument. Hence the parameter μ in the statement of the *Forking Lemma* is equal to $1 + log_2(l)$. The corresponding tree of accepting transcripts has a depth equal to $\mu = 1 + log_2(l)$. The node in the first level has $n_1 = 2$ children in the second level of the tree corresponding to the 2 values of the challenge x (x_1, x_2). Each node in the second level of our tree is connected to the root of a tree of accepting transcripts of the "Extended-Schnorr" argument, which is a tree of depth $log_2(l)$ with 3 children per node (see the "Bulletproofs" paper), hence $n_i = 3$, $i \geq 2$
Therefore, the total number of nodes in the tree is $N = \prod_{i=1}^{\mu} n_i = \prod_{i=1}^{1+log_2(l)} n_i = n_1 \cdot \prod_{i=2}^{1+log_2(l)} = 2 \cdot \prod_{i=2}^{1+log_2(l)} 3 = 2 \cdot 3^{log_2(l)} = 2l^{1.58} = N^{PO}$, which is clearly bounded above by a polynomial in the parameter λ (the constant polynomial $l^2 = O(N^{PO})$ for example). In addition, χ is polynomial time since computing the ν_i amounts to solving a system of linear equations in $\mathbb{Z}_p$: $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, which can be done efficiently. All other operations are also basic polynomial time computations.
Finally, χ succeeds except with negligible probability, which allows using the *Forking Lemma*, because it only fails when we cannot compute ν_1, ν_2, which occurs only when the determinant of the matrix $\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$ is 0 (the linear system of equations has no solution then), or equivalently when $x_1 - x_2$ is zero that is when $x_1 = x_2$, which happens only with negligible probability $(1/p)$ since x_1 and x_2 are picked uniformly at random in $\mathbb{Z}_p$.

Chapter 11

Implementation

11.1 Overview

We implemented in *Python 3.8.8* the protocols *Logarithmic Proof*, *Min-Max K-selection proof protocol* and *Partial Opening*, described in the previous sections in accordance with the ElectionGuard specifications v.1.1 [11]. The implementation can be found on GitHub.
The group $\mathbb{G}$ is a cyclic group of r-adic residues modulo p.
The modulus p is a 4096-bit prime number that can be represented in hexadecimal form as follows :

```
p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
    93C467E3 7DB0C7A4 D1BE3F81 0152CB56 A1CECC3A F65CC019 0C03DF34 709AFFBD
    8E4B59FA 03A9F0EE D0649CCB 621057D1 1056AE91 32135A08 E43B4673 D74BAFEA
    58DEB878 CC86D733 DBE7BF38 154B36CF 8A96D156 7899AAAE 0C09D4C8 B6B7B86F
    D2A1EA1D E62FF864 3EC7C271 82797722 5E6AC2F0 BD61C746 961542A3 CE3BEA5D
    B54FE70E 63E6D09F 8FC28658 E80567A4 7CFDE60E E741E5D8 5A7BD469 31CED822
    03655949 64B83989 6FCAABCC C9B31959 C083F22A D3EE591C 32FAB2C7 448F2A05
    7DB2DB49 EE52E018 2741E538 65F004CC 8E704B7C 5C40BF30 4C4D8C4F 13EDF604
    7C555302 D2238D8C E11DF242 4F1B66C2 C5D238D0 744DB679 AF289048 7031F9C0
    AEA1C4BB 6FE9554E E528FDF1 B05E5B25 6223B2F0 9215F371 9F9C7CCC 69DDF172
    D0D62342 17FCC003 7F18B93E F5389130 B7A661E5 C26E5421 4068BBCA FEA32A67
    818BD307 5AD1F5C7 E9CC3D17 37FB2817 1BAF84DB B6612B78 81C1A48E 439CD03A
    92BF5222 5A2B38E6 542E9F72 2BCE15A3 81B5753E A8427633 81CCAE83 512B3051
    1B32E5E8 D8036214 9AD030AA BA5F3A57 98BB22AA 7EC1B6D0 F17903F4 E22D8407
```

The group $\mathbb{Z}_p$ is defined as the subgroup modulo q. q is a 256-bit prime such that $p-1$ is a multiple of q and $r = \frac{p-1}{q}$ is a cofactor of q. It can be represented under hexadecimal as follows :

```
q = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFF43

r = 1 00000000 00000000 00000000    00000000 00000000 00000000 00000000 000000BC
    93C467E3 7DB0C7A4 D1BE3F81 0152CB56 A1CECC3A F65CC019 0C03DF34 709B8AF6
    A64C0CED CF2D559D A9D97F09 5C3076C6 86037619 148D2C86 C317102A FA214803
    1F04440A C0FF0C9A 417A8921 2512E760 7B2501DA A4D38A2C 1410C483 6149E2BD
    B8C8260E 627C4646 963EFFE9 E16E495D 48BD215C 6D8EC9D1 667657A2 A1C8506F
    2113FFAD 19A6B2BC 7C457604 56719183 309F874B C9ACE570 FFDA877A A2B23A2D
    6F291C15 54CA2EB1 2F12CD00 9B8B8734 A64AD51E B893BD89 1750B851 62241D90
    8F0C9709 879758E7 E8233EAB 3BF2D6AB 53AFA32A A153AD66 82E5A064 8897C9BE
    18A0D50B ECE030C3 432336AD 9163E33F 8E7DAF49 8F14BB28 52AFFA81 4841EB18
    DD5F0E89 516D5577 76285C16 071D2111 94EE1C3F 34642036 AB886E3E C28882CE
    4003DEA3 35B4D935 BAE4B582 35B9FB2B AB713C8F 705A1C7D E4222020 9D6BBCAC
    C4673186 01565272 E4A63E38 E2499754 AE493AC1 A8E83469 EEF35CA2 7C271BC7
    92EEE211 56E617B9 22EA8F71 3C22CF28 2DC5D638 5BB12868 EB781278 FA0AB2A8
    958FCCB5 FFE2E5C3 61FC1744 20122B01 63CA4A46 308C8C46 C91EA745 7C136A7D
    9FD4A7F5 29FD4A7F 529FD4A7 F529FD4A 7F529FD4 A7F529FD 4A7F529F D4A7F52A
```

We generate 33.024 generators for the group $\mathbb{G}$. This allows to use parameters up to $l = 128$ selections and a batch of $m = 128$ votes in the 0-1 logarithmic proof and $l = 128$ selections, a batch of $m = 128$ votes and a selection limit of $n = 128$ in the K selection proof. These values should be acceptable for a practical use.

As it is usually done, we use the Fiat-Shamir heuristic to make our protocols non-interactive by computing the challenges as a hash of all the public parameters and all the elements communicated by the Prover.[16]

The protocols in the following sections are implemented on an Intel i7-1165G7 processor, using the Python 3.8.8 interpreter, and gmpy2 2.1.5.

11.2 Logarithmic 0-1 proof

11.2.1 Execution time

Figure 11.1 represents the execution time of the logarithmic protocol for different values of the parameters l and m.
We can see that the execution time is linear with the parameter $l \cdot m$ (with l the number of candidates and m the number of votes). If we consider that exponentiations are the most time-consuming operations, this is in accordance with the theoretical time complexity for exponentiations computed at section 8.3 which was linear in $l \cdot m$: $16lm + 16l + m + 4log_2(lm) + 6log_2(l) - 11$ (prover) $+ 8lm + 7l + m + 2log_2(lm) + 4log_2(l) + 2$ (verifier) $= 24lm + 13l + 2m + 6log_2(lm) + 12log_2(l) - 9$.
We use the number of exponentiations in $\mathbb{G}$ as an approximation for the time complexity. This is a realistic assumption in the sense that we measured that exponentiations were taking between 80 and 95% of the total execution time.

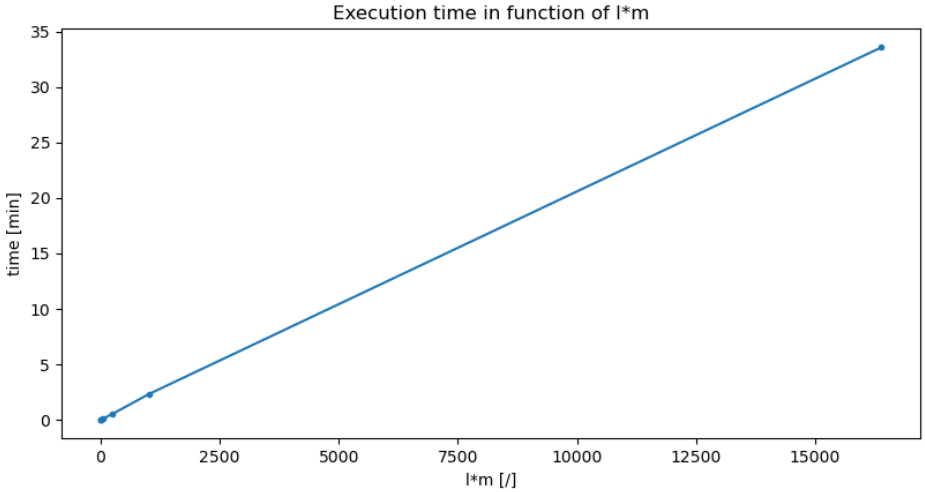

Figure 11.1: Execution time of the logarithmic proof

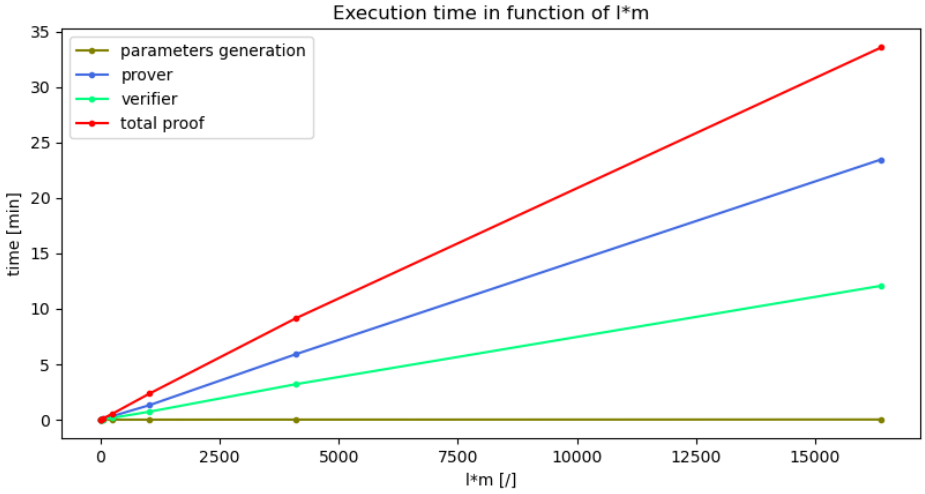

Figure 11.2: Detailed execution time of the logarithmic proof

The protocol involves 3 steps: the parameters generation, the proof generation and the proof verification. The parameters generation consists of the loading from a file of the required generators and the computation of a multi-Pedersen commitment on randomly chosen votes. Figure 11.2 shows the time required for each of these steps and their relation to the total execution time for the three steps.
The parameters' generation is almost constant time as it only requires drawing random exponents and computing the

commitment. The most time-consuming part is the loading of all the generators.As expected the proof verification time is faster than the proof generation time. This is a desired property in an election as the proof will typically be generated only once for each prover (or batch of provers) and then all proofs will have to be verified at least once but often several times by the verifiers (which will often be the election's authorities)

11.2.2 Proof size

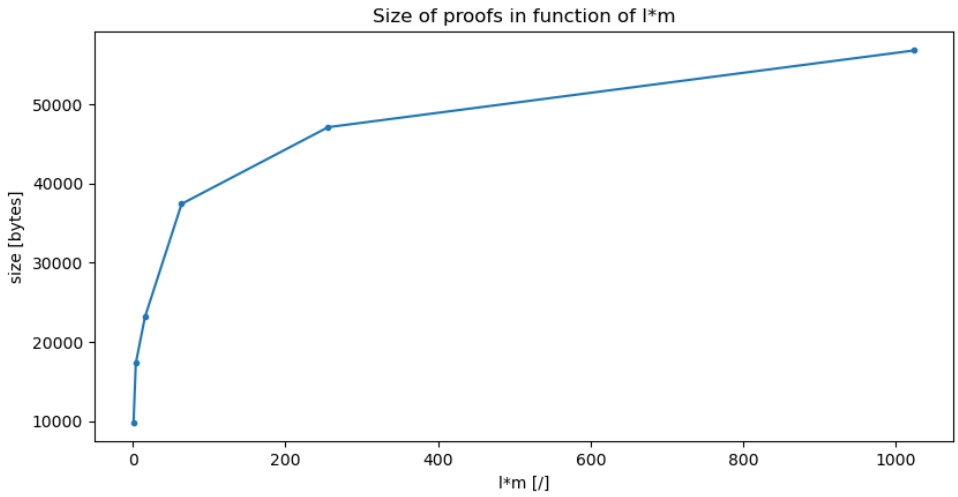

Figure 11.3: Proof size

Figure 11.3 shows the size of the elements transmitted during the proof. The size is clearly logarithmic which agrees with the theoretical complexity size computed in section 8.3 : $2log_2(ml) + 4log_2(l) + 12$.
The size of the proofs is consequently very short. For more than 16000 selections ($l = m = 128$), the size of the proof is only around 50 kBytes.
We have thus succeeded in obtaining what we were looking for building compact proofs in order to conduct Risk Limiting Audits.

11.3 Multibatching Min-Max Selection proof

11.3.1 Execution time

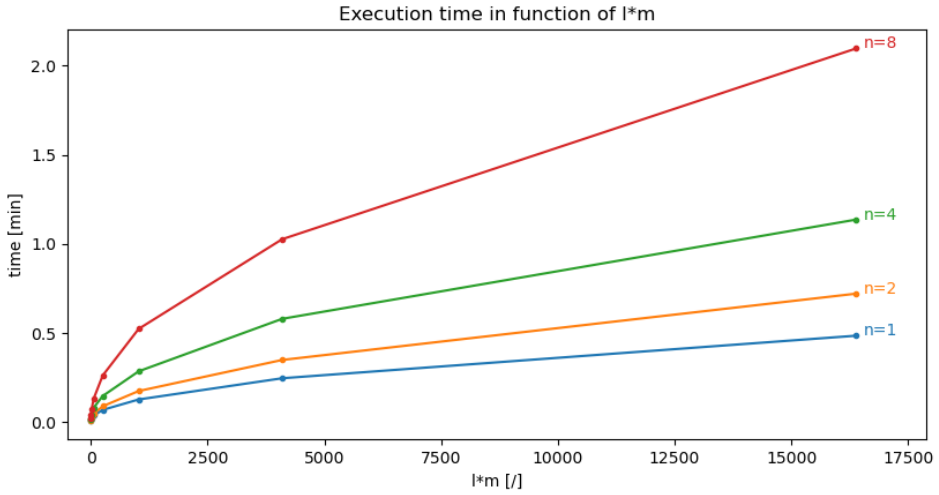

Figure 11.4: Total execution time of the K-selection roof

Figure 11.4 shows the evolution of the execution time of the multibatching min-max selection proof for $n \in [1, 2, 4, 8]$ and $l \cdot m \in [1 \cdot 1, 2 \cdot 2, \ldots, 128 \cdot 128]$ with $l = m$. As we can see the curve is not linear with respect to the term $l \cdot m$. This is in accordance with the theoretical prediction. Indeed, recall that in section 9.5 we had predicted the following complexity : $15mn+13l+m+4log_2(mn)+6log_2(l)-7$ (prover) $+ 7mn+6l+m+2log_2(mn)+4log_2(l)+10$ (verifier) $= 22mn + 19l + 2m + 6log_2(mn) + 10log_2(l) + 3 = O(mn + l)$. We have thus a linear term with $n \cdot m$ and a linear term with l but no linear term with respect to $l \cdot m$. This explains the behaviour observed on the graph. Since we chose $l = m$ to generate the proof notice that we have $m = \sqrt{m \cdot m} = \sqrt{m \cdot l}$. And thus a complexity of $O(nm) = O(n \cdot \sqrt{l \cdot m})$. This explains the behavior of the graph which looks like a square root curve.
The linearity remains in the fact that the execution time increases linearly with n.

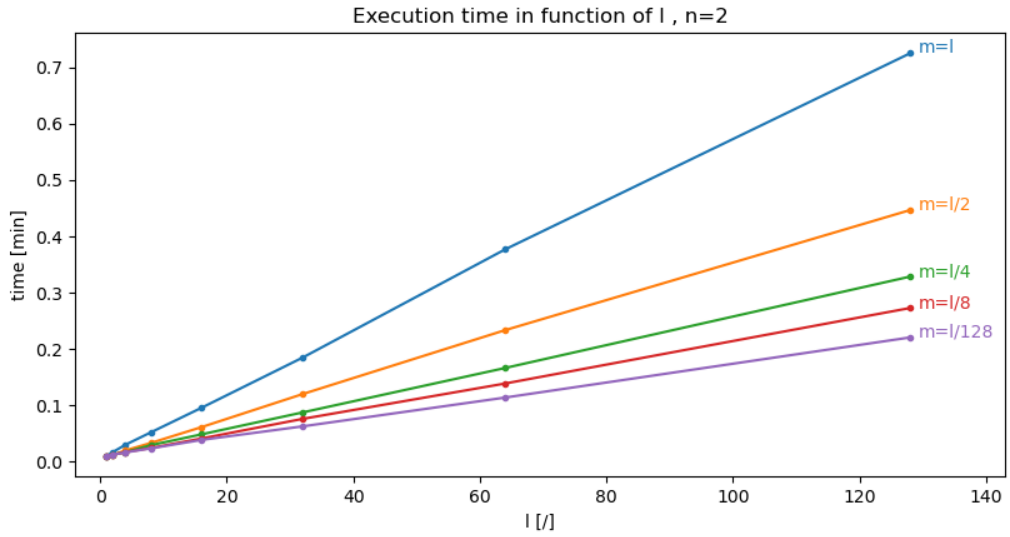

Figure 11.5: Execution time for multiple ratios between l and m of the K-selection proof protocol

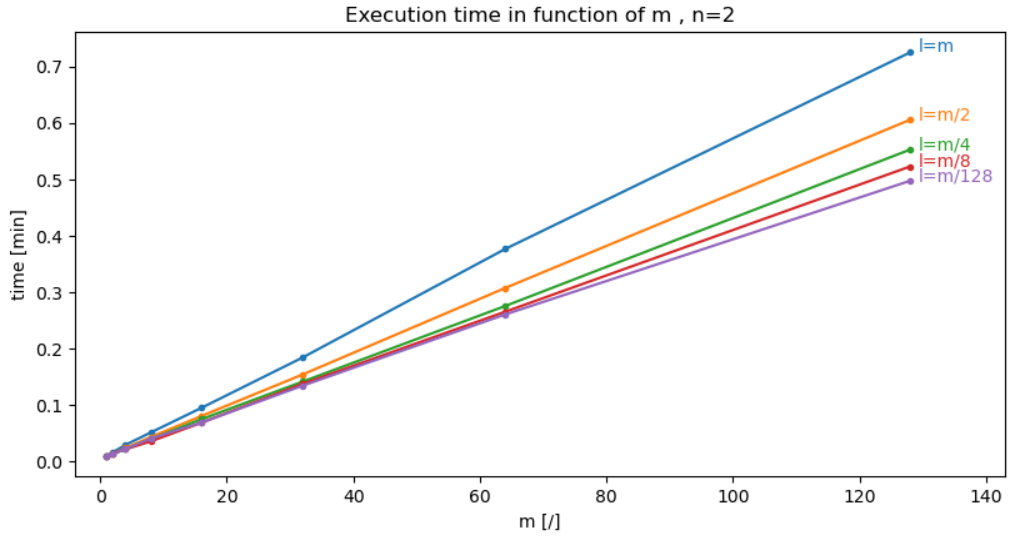

Figure 11.6: Execution time for multiple ratios between l and m of the K-selection proof protocol

Figures 11.5 and 11.6 represent the execution time of the protocol with different values for the ratio between l and m and with n fixed to 2. $n = 2$ allows to choose between 0 and $2^n - 1 = 3$ selections which is a realistic value. We confirm from the two graphs that the execution time is linear with respect to l and m (but not with respect to $l \cdot m$).
We can also deduce by the fact that the execution time is higher when $\frac{l}{m} < 1$ compared to when $\frac{l}{m} > 1$ for a fixed value of the product $l \cdot m$ that the factor m has a bigger impact on the execution time. This could have been once

again predicted by the theoretical complexity that the term m has more impact as we had computed a complexity of $\sim 22mn + 19l = 44m + 19l$ when $n = 2$.

11.3.2 Proof size

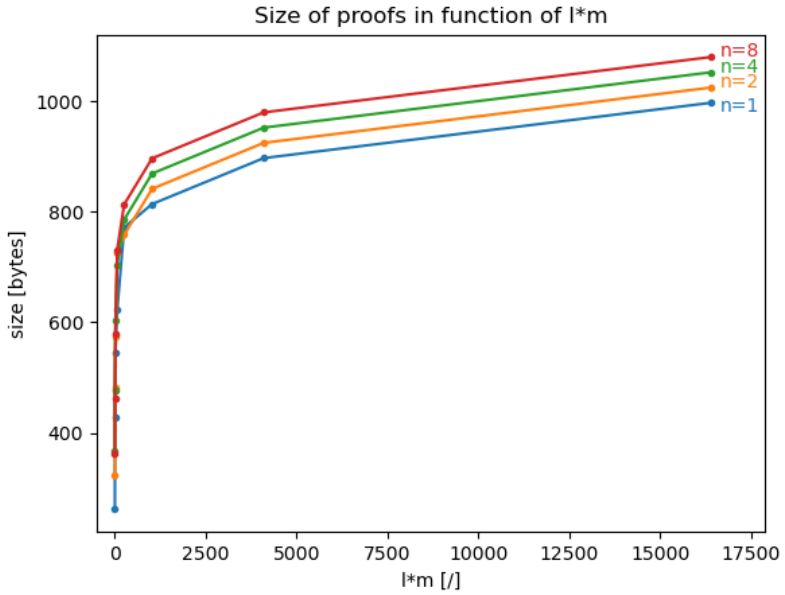

Figure 11.7: Proof Size for the K-selection Protocol

As expected, the size of the element exchanged is logarithmic in $n \cdot m$ (Figure 11.7). The number of elements communicated was indeed of $2log_2(mn) + 4log_2(l) + 18$ elements in Section 9.5.
The size of the proofs is consequently very short. For more than 16000 selections ($l = m = 128$) and a selection limit of 255 ($n = 8$), the size of the proof is only 1 kBytes.

11.4 Partial Opening

In this section, we will analyse the performance of the Partial Opening protocol.

11.4.1 Execution time

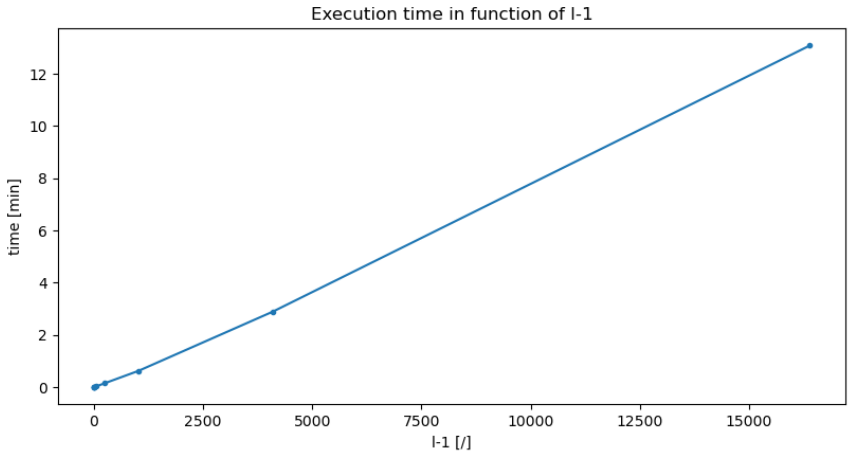

Figure 11.8: Total execution time of the Partial Opening protocol

Figure 11.8 shows that the execution time of the Partial Opening protocol is linear in l as predicted by the theoretical computation: $5l + 2log_2(l-1) - 5$ (prover) $+ 2l + 2log_2(l-1)$ (verifier) $= 7l + 4log_2(l-1) - 5$ (cfr. Section 10.2).

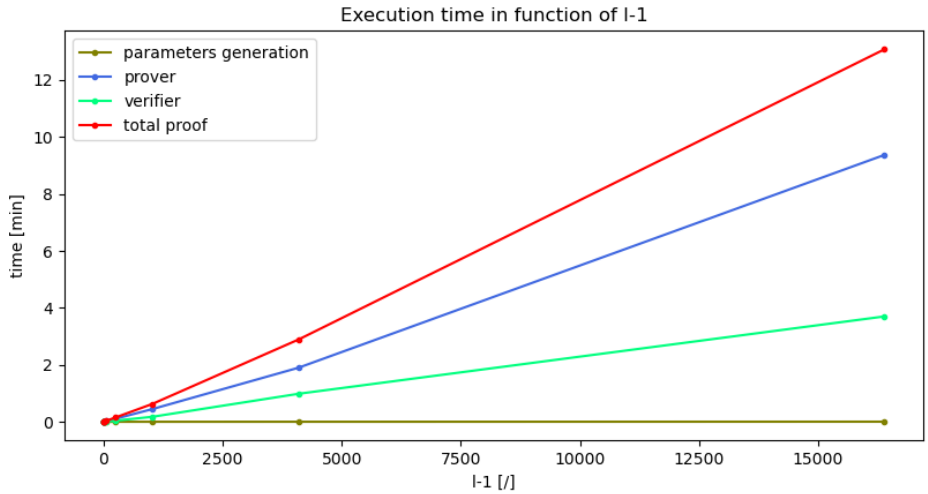

Figure 11.9: Generation time, execution time of the Prover and the Verifier for the Partial Opening protocol

Once gain, the execution time of the prover is almost 2 times longer than the one for the verifier.
As explained above this is a desired property in our case.

11.4.2 Proof size

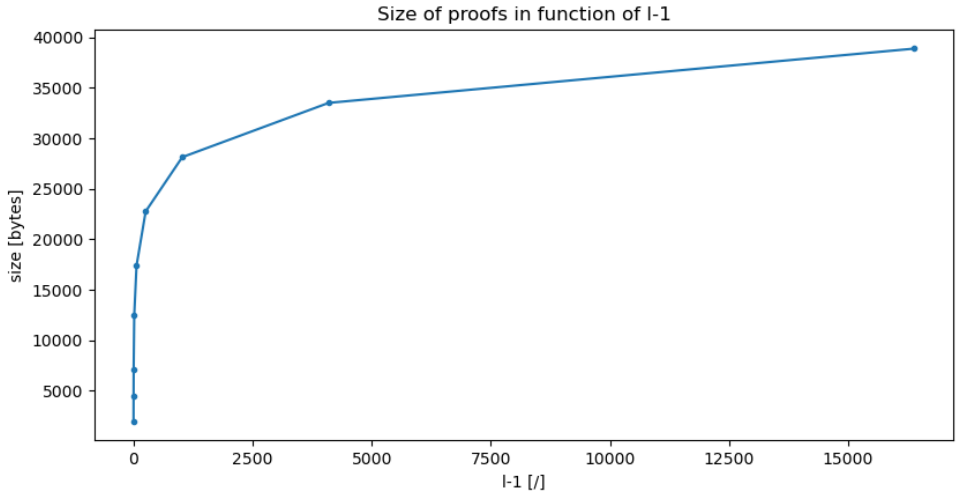

Figure 11.10: Proof Size for the Partial Opening Protocol

The size of the elements exchanged is logarithmically increasing with l (Figure 11.10).
This is in accordance with theory ($2log_2(l-1)+3$ computed at section 10.2) and all in all this shows that we have well succeeded in achieving our initial goal which was to construct a proof system of Risk Limiting Audits that allow to a have compact proofs.

11.5 Exponentiations with precomputations

We managed to decrease drastically the proof size but the execution time remains too high. As the most time-consuming part of the implementations consists in the exponentiations, we tried to reduce it by using a fixed-based exponentiation technique that takes advantage of a precomputed table containing some powers of the base that will be multiplied together to obtain the result [9]. As the base elements should be known in advance to precompute the tables, we applied this technique only on the generators $(g, h, u, g_1, \ldots, g_l, h_1, \ldots, h_l, \mathbf{g_1}, \ldots, \mathbf{g_l h_1}, \ldots, \mathbf{h_l}$. The number of powers precomputed is $\lceil \frac{256}{k} \rceil \cdot 2^k$ and depends on a parameter k.

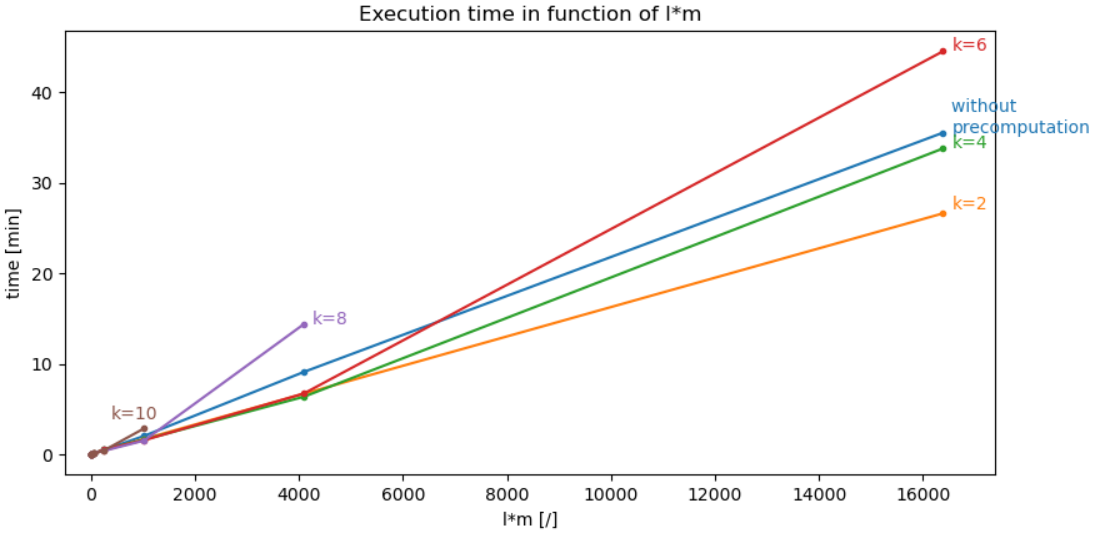

Figure 11.11: Execution time of the Prover and the Verifier for the Logarithmic 0-1 Protocol

Figure 11.11 shows the execution time of the Prover and the Verifier of the Logarithmic 0-1 proof (without the generation time). We can clearly see a reduction in the execution time using precomputation compared to when it is not used. For $k = 2$ and $n = m = 128$, there is a $\sim 33\%$ reduction corresponding to more than 10 minutes.
Using $k = 4$ or $k = 6$ remains interesting until $l = m = 64$ but the gain disappears for $l = m = 128$. This is because the precomputation table becomes too big. Indeed, when $l = m = 128$ and $k = 6$, the number of elements in the table for one generator is $\lceil \frac{256}{6} \rceil \cdot 2^6 = 2756$ and there are 16384 such tables to compute for all the generators. This can take a lot of space in memory and make the computation slower. In the same order of idea, on a computer with 16GB RAM, it is not possible to run the protocol with $k = 8$ and $l = m$ higher than 64 or with $k = 10$ and $l = m$ higher than 32.
The best value for the parameter k should be chosen in function of the memory capacity of the devices.

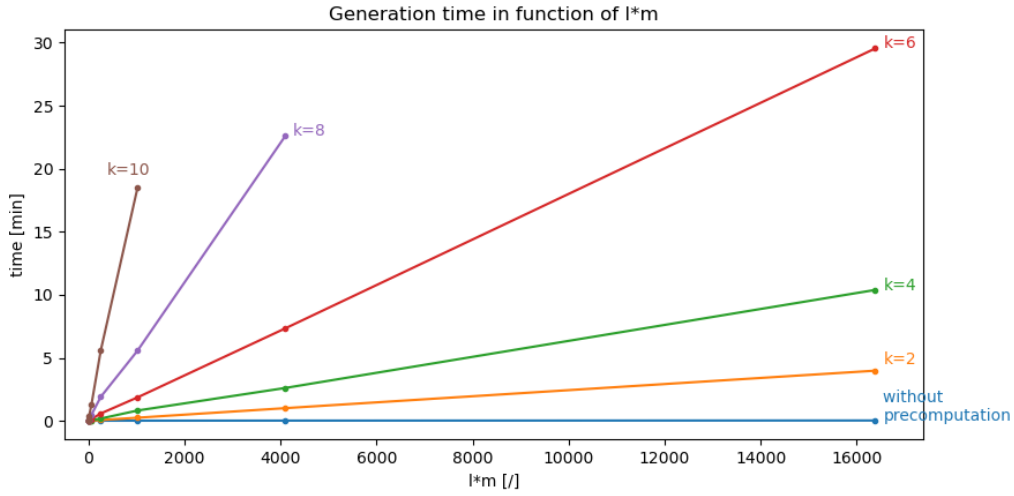

Figure 11.12: Generation time for the Logarithmic 0-1 Protocol

The generation time includes in this case the time to load the generators and to create the precomputation tables. As expected, the generation time grows linearly with $l \cdot m$ for a fixed value of k but grows exponentially with k (Figure 11.12). However, having a big generation time should not be too problematic in the sense that the generation only happens once while the protocol can be reused as much as needed afterwards.
Considering both the execution time and the generation time, choosing $k = 2$ seems to be a good compromise in this case.

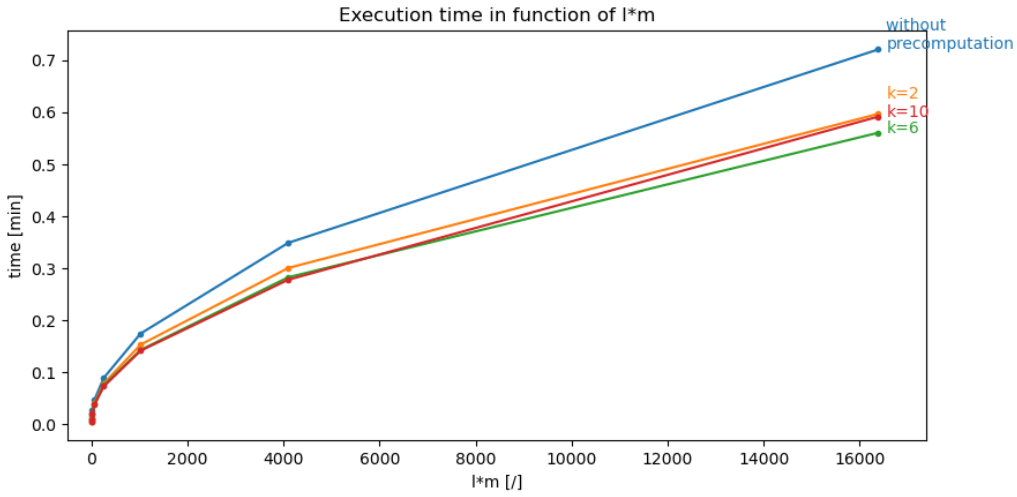

Figure 11.13: Execution time of the Prover and the verifier for the K-Selection Protocol

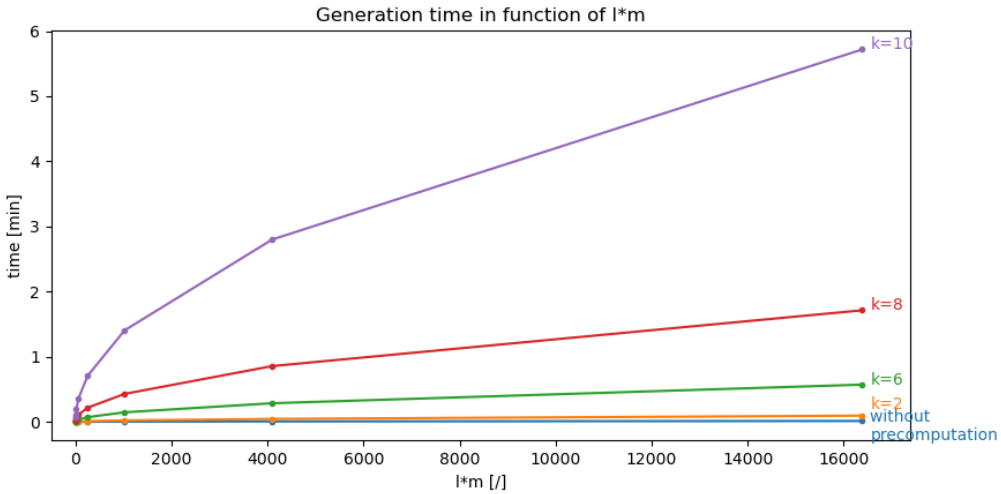

Figure 11.14: Generation time for the K-selection Protocol

A similar behaviour can be observed for the K-selection Protocol in Figures 11.13 and 11.14 but this time all the values of k up to 10 lead to better execution time than without precomputation table .

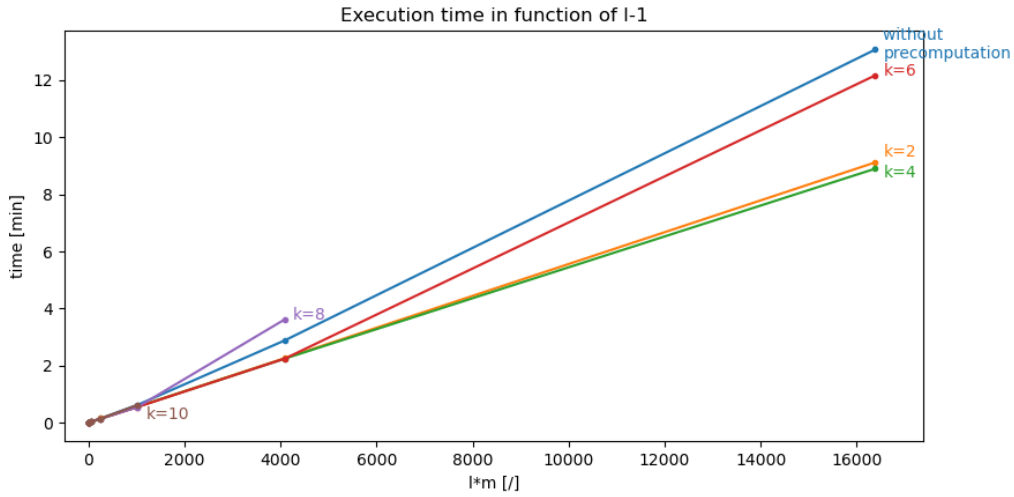

Figure 11.15: Execution time of the Prover and the Verifier of the Partial Opening protocol

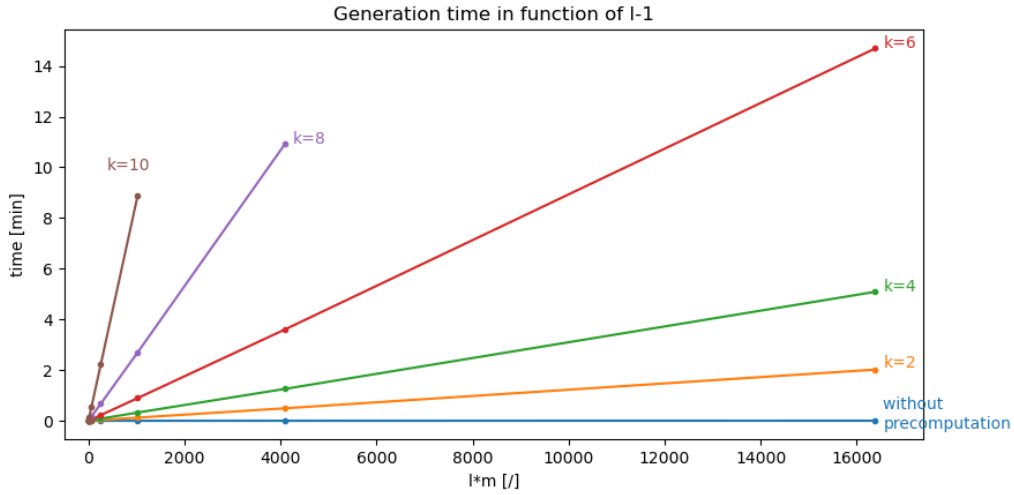

Figure 11.16: Generation time of the Partial Opening protocol

For the Partial Opening Protocol, the fixed based exponentiation presents an advantage until $k = 6$ (Figures 11.15 and 11.16).

One way to further reduce the execution time could be to optimize the number of generators for which a table is precomputed instead of computing a table for all of them.

Conclusion

In this master thesis, we have presented several zero-knowledge protocols that we developed, together with their associated security proofs and their complexity.

We have shown through implementations and measurements that the practical performance of our protocols matches the theoretically predicted complexities.
In particular, the main goal which was to obtain protocols with proofs of compact logarithmic size is reached.

The handling of large data volumes when conducting Risk Limiting Audits was an obstacle in several cases to the deployment of ElectionGuard in the United States. Our work allows to overcome this limitation by making a significant contribution to the field of secure and efficient auditing in the context of voting systems.
In addition, our cryptographic protocols might find a practical use within the framework of Microsoft's ElectionGuard and can thus have a real world impact by enabling more secure and private voting in the future.

However, several questions naturally arise from our work and several improvements remain possible.
The first question that arose naturally is the compatibility issue with the nowadays version of ElectionGuard which currently uses ElGamal votes ciphertexts instead of the multi-Pedersen votes commitments used in our work.
Another question is the time efficiency of our protocols. They have been thought to be very data-efficient but they have not been optimized for the computational time point of view. The possibility to develop algorithms that would be logarithmic both in proof size and computational time remains an open question.

A lot of optimizations are still possible including in the implementations and the algorithms themselves.
In the implementations, it might be possible to combine several computations, especially at the verification step as explained in the "Bulletproofs" paper from the Stanford University.
In our paper, we have considered treating "Bulletproofs" as a black box. The main advantage is that it simplifies a lot the reasoning but it might be more efficient by not doing so: for example, by doing the "Extended-Schnorr" protocol and "Bulletproofs" together several computations steps could be saved. In the same order of idea, it might be possible to combine the K-selection and the 0-1 protocols in one single protocol which could allow to combine several computations step together.

Bibliography

[1] *Another Step in Testing ElectionGuard*. https://blogs.microsoft.com/on-the-issues/2020/02/17/wisconsin-electionguard-polls/. Feb. 2020.

[2] Josh Benaloh et al. "VAULT-Style Risk-Limiting Audits and the Inyo County Pilot". In: *IEEE Security & Privacy* 19.4 (July 2021), pp. 8–18. ISSN: 1558-4046. DOI: 10.1109/MSEC.2021.3075107.

[3] Jonathan Bootle et al. *Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting*. Cryptology ePrint Archive, Paper 2016/263. https://eprint.iacr.org/2016/263. 2016. URL: %5Curl%7Bhttps://eprint.iacr.org/2016/263%7D.

[4] Benedikt Bunz et al. *Bulletproofs: Short Proofs for Confidential Transactions and More*. San Francisco, CA: IEEE, May 2018, pp. 315–334. ISBN: 978-1-5386-4353-2. DOI: 10.1109/SP.2018.00020.

[5] David Chaum and Torben Pryds Pedersen. "Wallet Databases with Observers". In: ed. by Ernest F. Brickell. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1993, pp. 89–105. ISBN: 978-3-540-48071-6. DOI: 10.1007/3-540-48071-4_7.

[6] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". In: *Advances in Cryptology — CRYPTO '94*. Ed. by Yvo G. Desmedt. Vol. 839. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 174–187. ISBN: 978-3-540-58333-2. DOI: 10.1007/3-540-48658-5_19.

[7] *Democracy Forward - Protecting Democracy | Microsoft CSR*. https://www.microsoft.com/en-us/corporate-responsibility/democracy-forward.

[8] Henri Devillez, Olivier Pereira, and Thomas Peters. *How to Verifiably Encrypt Many Bits for an Election?* 2022.

[9] Henri Devillez, Olivier Pereira, and Thomas Peters. *How to Verifiably Encrypt Many Bits for an Election?* Cryptology ePrint Archive, Paper 2022/1051. https://eprint.iacr.org/2022/1051. 2022. URL: https://eprint.iacr.org/2022/1051.

[10] *EAC Report Final.Pdf*. https://www.electionguard.vote/images/EAC%20Report%20Final.pdf. (Visited on 06/02/2023).

[11] *ElectionGuard - Official Specifications*. https://www.electionguard.vote/spec/. (Visited on 05/16/2023).

[12] *ElectionGuard - Overview*. https://www.electionguard.vote/spec/web/1_Overview.

[13] *ElectionGuard - What Is ElectionGuard?* https://www.electionguard.vote/.

[14] *ElectionGuard Glossary*. https://www.electionguard.vote/overview/Glossary/#contest. (Visited on 06/02/2023).

[15] *ElectionGuard Glossary*. https://www.electionguard.vote/overview/Glossary/#selection-limit. (Visited on 06/02/2023).

[16] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by Andrew M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194. ISBN: 978-3-540-47721-1.

[17] Jens Groth. "Linear Algebra with Sub-linear Zero-Knowledge Arguments". In: ed. by Shai Halevi. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 192–208. ISBN: 978-3-642-03356-8. DOI: 10.1007/978-3-642-03356-8_12.

[18] Yun-Xing Kho, Swee-Huay Heng, and Ji-Jian Chin. "A Review of Cryptographic Electronic Voting". In: *Symmetry* 14.5 (May 2022), p. 858. ISSN: 2073-8994. DOI: 10.3390/sym14050858.

[19] Adi Shamir. "How to Share a Secret". In: *Communications of the ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176.

[20] *Structure and Processes*. https://www.electionguard.vote/concepts/Structure_and_Processes/. (Visited on 06/02/2023).

[21] *What is ElectionGuard?* https://www.electionguard.vote/. (Visited on 06/02/2023).

Appendices

Appendix A

In this section, we prove the following statement :

$$\sum_{k=1}^{m}(\mathbf{a_L^{(j)}}[k] - v_j^{(k)})z^{k+1} + z < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m > + < \mathbf{a_L^{(j)}}, \mathbf{a_R^{(j)}} \circ \mathbf{y}^m >= 0$$

$$\Leftrightarrow < \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >= \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + \delta(y, z)$$

PROOF:

$$< \mathbf{a_L^{(j)}} - z \cdot \mathbf{1}^m, \mathbf{y}^m \circ (\mathbf{a_R^{(j)}} + z \cdot \mathbf{1}^m) + z^2 \cdot \mathbf{z}^m >$$

$$=< \mathbf{a_L^{(j)}}, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} > + < \mathbf{a_L^{(j)}}, \mathbf{y}^m \circ (z \cdot \mathbf{1}^m) > + \sum_{k=1}^{m} < \mathbf{a_L^{(j)}}, (\mathbf{0}^{k-1}||1||\mathbf{0}^{m-k})z^{k+1} > - z < \mathbf{1}^m, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} >$$

$$- z < \mathbf{1}^m, \mathbf{y}^m \circ (z \cdot \mathbf{1}^m) > - \sum_{k=1}^{m} < \mathbf{1}^m, (\mathbf{0}^{k-1}||1||\mathbf{0}^{m-k})z^{k+2} >$$

$$= \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + \delta(y, z) = \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + (z - z^2) < \mathbf{1}^m, \mathbf{y}^m > - \sum_{k=1}^{m} z^{k+2}$$

$$\Leftrightarrow \sum_{k=1}^{m} < \mathbf{a_L^{(j)}}, (\mathbf{0}^{k-1}||1||\mathbf{0}^{m-k}) > z^{k+1} + < \mathbf{a_L^{(j)}}, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} > + z < \mathbf{a_L^{(j)}}, \mathbf{y}^m >$$

$$- z < \mathbf{1}^m, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} > \cancel{- z^2 < \mathbf{1}^m, \mathbf{y}^m >} \cancel{- \sum_{k=1}^{m} z^{k+2}}$$

$$= \sum_{k=1}^{m} \overline{v}_j^{(k)} z^{k+1} + z < \mathbf{1}^m, \mathbf{y}^m > \cancel{- z^2 < \mathbf{1}^m, \mathbf{y}^m >} \cancel{- \sum_{k=1}^{m} z^{k+2}}$$

$$\Leftrightarrow \sum_{k=1}^{m} (< \mathbf{a_L^{(j)}}, (\mathbf{0}^{k-1}||1||\mathbf{0}^{m-k}) > - \overline{v}_j^{(k)}) z^{k+1} + < \mathbf{a_L^{(j)}}, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} > +$$

$$z \left(\underbrace{< \mathbf{a_L^{(j)}}, \mathbf{y}^m >}_{=<\mathbf{y}^m, \mathbf{a_L^{(j)}}>} - \underbrace{< \mathbf{1}^m, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} >}_{=<\mathbf{y}^m, \mathbf{a_R^{(j)}}>} - \underbrace{< \mathbf{1}^m, \mathbf{y}^m >}_{=<\mathbf{y}^m, \mathbf{1}^m>} \right)$$

$$= \sum_{k=1}^{m} (\mathbf{a_L^{(j)}}[k] - \overline{v}_j^{(k)}) z^{k+1} + z < \mathbf{a_L^{(j)}} - \mathbf{1}^m - \mathbf{a_R^{(j)}}, \mathbf{y}^m > + < \mathbf{a_L^{(j)}}, \mathbf{y}^m \circ \mathbf{a_R^{(j)}} >= 0$$

which concludes the proof.

Appendix B

In this section, we prove the following statement :

$$\sum_{j=1}^{m}(< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >)z^{j+1} + < \mathbf{a_L} - \mathbf{1}^{n\cdot m} - \mathbf{a_R}, \mathbf{y}^{n\cdot m} > z + < \mathbf{a_L}, \mathbf{a_R} \circ \mathbf{y}^{n\cdot m} >= 0$$

$$\Leftrightarrow < \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} >= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y, z)$$

PROOF:

$$< \mathbf{a_L} - z \cdot \mathbf{1}^{n\cdot m}, \mathbf{y}^{n\cdot m} \circ (\mathbf{a_R} + z \cdot \mathbf{1}^{n\cdot m}) + \sum_{j=1}^{m}(\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} >$$

$$=< \mathbf{a_L}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} > + < \mathbf{a_L}, \mathbf{y}^{m\cdot n} \circ (z \cdot \mathbf{1}^{m\cdot n}) > + \sum_{j=1}^{m} < \mathbf{a_L}, (\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} > -$$

$$z < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} > -z < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} \circ (z \cdot \mathbf{1}^{m\cdot n}) > - \sum_{j=1}^{m} < \mathbf{1}^{m\cdot n}, (\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+2} >$$

$$= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + \delta(y, z) = \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + (z - z^2) < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} > - \sum_{j=1}^{m} < \mathbf{1}^n, \mathbf{2}^n > z^{j+2}$$

$$\Leftrightarrow \sum_{j=1}^{m} < \mathbf{a_L}, (\mathbf{0}^{(j-1)\cdot n}||\mathbf{2}^n||\mathbf{0}^{(m-j)\cdot n})z^{j+1} > + < \mathbf{a_L}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} > + z < \mathbf{a_L}, \mathbf{y}^{m\cdot n} >$$

$$- z < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} > \cancel{-z^2 < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} >} - \cancel{\sum_{j=1}^{m} < \mathbf{1}^n, \mathbf{2}^n > z^{j+2}}$$

$$= \sum_{j=1}^{m} < \mathbf{v^j}, \mathbf{1}^l > z^{j+1} + z < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} > \cancel{-z^2 < \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} >} - \cancel{\sum_{j=1}^{m} < \mathbf{1}^n, \mathbf{2}^n > z^{j+2}}$$

$$\Leftrightarrow \sum_{j=1}^{m}(< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >)z^{j+1} + < \mathbf{a_L}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} > +$$

$$z\left(\underbrace{< \mathbf{a_L}, \mathbf{y}^{m\cdot n} >}_{=<\mathbf{y}^{m\cdot n}, \mathbf{a_L}>} - \underbrace{< \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} >}_{=<\mathbf{y}^{m\cdot n}, \mathbf{a_L}>} - \underbrace{< \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} >}_{=<\mathbf{y}^{m\cdot n}, \mathbf{1}^{m\cdot n}>}\right)$$

$$= \sum_{j=1}^{m}(< \mathbf{a_L^{(j)}}, \mathbf{2}^n > - < \mathbf{v^j}, \mathbf{1}^l >)z^{j+1} + z < \mathbf{a_L} - \mathbf{1}^{m\cdot n} - \mathbf{a_R}, \mathbf{y}^{m\cdot n} > + < \mathbf{a_L}, \mathbf{y}^{m\cdot n} \circ \mathbf{a_R} >= 0$$

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**