UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# ACHIEVING TRANSPARENCY IN PUBLIC DECISION MAKING PROCESSES VIA VERIFIABLE RANDOMNESS

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
MEMORIA PARA OPTAR AL TITULO DE INGENIERÍA CIVIL EN COMPUTACIÓN

CONSTANZA ANDREA CSORI PINTO

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIERREZ GALLARDO
JORGE PEREZ ROJAS
MAURICIO SOLAR FUENTES

Este trabajo ha sido parcialmente financiado por 2016-NIST-MSE-01

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: CONSTANZA ANDREA CSORI PINTO
FECHA: 2019
PROF. GUÍA: ALEJANDRO HEVIA ANGULO

# ACHIEVING TRANSPARENCY IN PUBLIC DECISION MAKING PROCESSES VIA VERIFIABLE RANDOMNESS

This work is motivated by providing accountability to public institutions using creative and practical new combinations of cryptographic and technological tools. The need for accountability arises from multiple fronts in the public discussion. Government agencies are often criticised for not providing enough transparency, while regular citizens want to verify the operation and trustworthiness of their public institutions. A more effective approach is that public institutions offer active mechanisms for transparency which are openly provided to citizens. Our work gives one such mechanism for public institutions. We provide a way to verify the random component of any public decision making processes, thus providing transparency of any decision taken using the output of such components.

The main goal of this thesis work is achieving transparency via verifiable randomness. We start by defining what it means to achieve transparency, and how it impacts a process. We then study verifiable randomness: the concept of a "randomness beacon", how it can be applied, the consequences of using it in any randomised process, and how such processes looks in terms of its interaction with the randomness beacon. We examine the cryptographic tools that are needed to achieve that goal, and identify which implementations of those tools are useful and available. We conclude by extending the case of randomness verification from the initial goal to the case when private data needs to be used in the processes.

An important achievement of this work is the implementation of a working real-life prototype of a randomness-verifiably decision-making process in a public organisation, namely the Contraloría General de la República of Chile (the main public agency in charge of audits in Chile). We worked alongside CGR to create a client that would generate verifiable random selections. We developed a general and extensible engine that enables the creation of this verifiable random selection with a verification bundle. We also implemented a verification engine that receives the verification bundle and checks the correctness of the execution encoded in the bundle. Finally, we also developed a private engine wrapper that adds a privacy layer to any private information that needs to be included in the process.

With this work we hope to show one way of providing accountability by making public random decisions transparent. The work developed can be easily replicated and extended, and it is our hope that it is implemented in other organisations, possibly with new requirements so the engine can grow more robust. We see this work as a stepping stone in working with government agencies using cryptographic tools to provide transparency. Making transparency available to the public, no matter their scientific background was indeed one of our goals.

Ultimately we hope that this work will have some impact on how public transparency is actively delivered in public institutions, and how this process is viewed from the within the institutions.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: CONSTANZA ANDREA CSORI PINTO
FECHA: 2019
PROF. GUÍA: ALEJANDRO HEVIA ANGULO

# ACHIEVING TRANSPARENCY IN PUBLIC DECISION MAKING PROCESSES VIA VERIFIABLE RANDOMNESS

Este trabajo es motivado por la obligación que tienen las instituciones públicas de rendir cuentas usando nuevas combinaciones, creativas y prácticas, de herramientas tecnológicas y criptopgráficas. La necesidad de rendir cuentas nace de múltiples frentes en la discusión pública. Las agencias gubernamentales son frecuentemente criticadas por no proveer suficiente transparencia, mientras que los ciudadanos quieren verificar la operación y confianza de sus instituciones públicas. Un enfoque más efectivo es que las instituciones ofrezcan mecanismos de transparencia activos que estén abiertos a los ciudadanos. Nuestro trabajo da un mecanismo de estas características para las instituciones públicas. Proveemos una forma para verificar el componente aleatorio de cualquier proceso de decisión pública, así dando transparencia a cualquier decisión que se tome a partir de este componente.

El objetivo principal de este trabajo de tésis es lograr transparencia a través de aletoriedad verificable. Comenzamos definiendo que significa lograr transparencia, y cómo esto impacta un proceso. Luego estudiamos aleatoriedad verificable: el concepto de un "faro de aleatoriedad", cómo puede ser aplicado, las consecuencias de usarlo en cualquier proceso aleatorizado, y como ese proceso se identifica en términos de la interacción con el faro de aleatoriedad. Examinamos herramientas criptográficas que son necesarias para alcanzar el objetivo, e identificamos cuales implementaciones de esas herramientas son útiles y están disponibles. Concluimos extendiendo el caso de la verificación de aletoriedad desde la meta inicial al caso donde datos privados necesitan ser usados en los procesos.

Un logro de este trabajo es la implementación de un prototipo en una situación real de procesos con decisiones públicas que involucran aleatoriedad verificable en una organización pública, específicamente la Contraloría General de la República de Chile (principal agencia pública en cargo de auditorías en Chile). Trabajamos junto a CGR para crear un cliente que generaría selecciones a partir de aleatoriedad verificable. Desarrollamos un motor genérico y extensible que permite la creación de estas selecciones junto a un paquete de verificación. También implementamos un motor de verificación que recibe el paquete de verificación y chequea la correctitud de la ejecución codificada en el paquete. Finalmente, también desarrollamos un motor envoltorio privado que añade una capa de privacidad a cualquier información privada que necesite ser incluida en el proceso.

Con este trabajo esperamos mostrar una forma de hacer la toma de decisiones aleatorias transaparentes. El trabajo desarrollado puede ser fácilmente replicado y extendido. Es nuestra esperanza que sea adoptado en otras organizaciones con nuevos requerimientos para que pueda crecer y ser más robusto. Vemos este trabajo como un peldaño importante con agencias gubernamentales usando herramientas criptográficas para proveer transparencia.

*To my mother, my best friend.*

# Acknowledgements

Quisiera agradecer antes de todo a mi madre. Paola, sin ti nada de lo que soy sería posible. Tu apoyo incondicional, tu ejemplo de vida y de perseverancia son trascendentales para mí. Eres el pilar de mi vida y este logro es nuestro. Realmente eres fantástica.

Gracias a mi familia. Sebastián, con tu forma especial de ver el mundo me desafías constantemente a mirar dos veces, estoy siempre orgullosa de ser tu hermana. Mami y Tata, son una inspiración y no podría haber pedido mejores tatas que ustedes. Hugo, tu apoyo constante me ha ayudado a formarme como persona y cuestionarme todo.

Thank you Alex, everything is a bit brighter with you around and the light at the end of the tunnels seem closer.

Gracias Romi, por siempre creer que voy a poder hacer lo que me proponga (y por los infinitos cafés y cervezas).

Gracias Nancy, por escucharme y apoyarme todos estos años y obligarme a seguir aún cuando no he querido.

Gracias a mi profesor guía, Alejandro. Usted confió en mis capacidades cuando creí que nadie lo haría, fue capaz de ver más allá y ayudarme a crecer no solamente como estudiante, sino que como profesional y persona. Y gracias a todo el equipo del CLCERT que apoyaron este trabajo desde su concepción.

Al que me envió en este loco camino, mis infinitas gracias profe Julio.

A todas las personas en Contraloría, especialmente al profesor Cesar Guerrero. Por creer en este proyecto y abrirme las puertas para trabajar con ustedes.

Quiero agradecer a todas mis amigas y amigos que han visto y estado en distintos aspectos de este largo proceso, especialmente a Juanjo, Nacho, Américo, Nico, Karin y Gonzalo.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Transparency and Randomness

Bill is just a regular citizen. Bill was just called up by his national tax agency to submit all his data to be audited. Bill, as a law abiding citizen, is not worried about what will come up from that, but he is curious about why the agency is using up resources that he feels would be much better spent auditing higher profile people. Bill wants to know how he was chosen; he wants to make sure the agency is not hand picking who to audit in order to cover up illegalities.

Now let us think about Joe, he is in charge of managing the tax agency. Joe wants to use up all of his resources wisely, he wants to reassure the public that his agency is doing its best to audit everyone that their resources permit. Joe wants to give the public a way to monitor the correctness of the selection process.

At present, there is a significant number of people around the world who are striving to give transparency to all sorts of human activities. Particularly to those that are in the public eye or carried out by the government. This is an essential step towards giving power and knowledge back to those affected [17]. Indeed, the goal of these initiatives is twofold. First, providing affected users with the tools to understand how these decisions were made. Also, giving decision makers mechanisms a way to achieve accountability for those decisions. Transparency, in this context, emerges as an essential mechanism to achieve this goal.

Yet, providing transparency in decision-making processes must balance several requirements. It must be affordable and feasible to implement. Equally important, it must be meaningful to those who are affected by these decisions. This task is often hard to achieve, as trust is not always implied by transparency as noted by Grimmelikhuijsen in [12]. There are several lines of study [6, 8, 22] that are currently trying to make the transition as easy and smooth as possible. The overall consensus, however, is that transparency is a crucial ingredient towards achieving trustworthy decision-making processes.

Making decision processes fully verifiable is not straightforward, especially in the case of randomised processes. The mere definition of random[1], tells us that it should not be a reproducible part of the process. However, there are some ways around it. The randomness beacon, a NIST[2] backed initiative, provides a constant source of randomness, generated by a trusted organisation. The idea in this thesis is to use the recently launched CLCERT randomness beacon[3] alongside the NIST randomness beacon[4] to help us ensure the fairness and correctness of the random coins used in the decision-making process.

The last important factor in decision-making processes is privacy. There are aspects of decision making which are intrinsically secret, such as the income salary of taxpayers. This creates a significant complication in developing a transparent process whilst not revealing all the information about it.

The problem of mixing accountable processes, verifiable randomness, and private data in public decision making has not been thoroughly studied in the open literature. The primary objective of this thesis is to fill this void by precisely defining what is necessary to achieve transparency in these situations. As part of this effort, we will present a functional prototype implemented at the Comptroller General of Chile (*Contraloría General de la República Chilena*, hereafter CGR), as well as a more in-depth study of a case where private information is involved and needs to be protected.

The thesis raises that any process that uses the beacon can be separated in 3 parts. A full picture of the backend, the engine and the client is shown in Figure 1.1, where it is possible to take a look at what the solution will look like, how each developed library works to solve and how they communicate between each other.

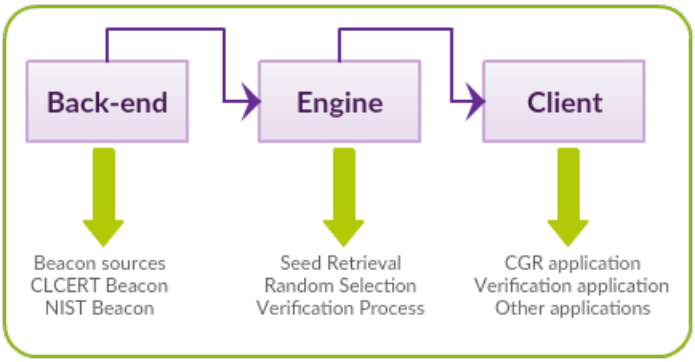


Figure 1.1: Prototype architecture.

[1]As the one given by the Oxford English Dictionary: *Governed by or involving equal chances for each item.*

[2]`https://www.nist.gov/`

[3]`https://beacon.clcert.cl/`

[4]`https://beacon.nist.gov/home`

## 1.2 Hypothesis, Goals, and Methodology

In this section, we state the hypothesis, goals, and methodology underlying the work of this thesis.

### 1.2.1 Hypothesis

*Verifiable randomness can be effectively used to provide transparency to a broad class of practical, randomised decision-making processes.*

### 1.2.2 Main Goal

To identify the characteristics, scope, and parameters of randomised decision making processes in which can be enhanced to provide transparency and to design and implement such a system for a practical case.

### 1.2.3 Specific Goals

1. To identify the types of transparency possible to capture in our setting.

2. To identify some overall design characteristics of an application for publicly verifiable processes.

3. To determine the scope of parameters for the targeted processes.

4. To identify the characteristics of meaningful and usable publicly verifiable processes, that is, which processes can effectively be verified by end users.

5. To have a successfully implemented working prototype for several relevant processes that implement public policy.

### 1.2.4 Methodology

The methodology to achieve the stated goals is the following.

For the working prototype:

- Evaluate the scope of this thesis, which types of applications (public decision making processes) can be adopted to provide transparency.
- Assess the scope and parameters used by the applications mentioned above.
- Work with the CLCERT and NIST beacon to build mechanisms to receive random sequences and integrate them into the program.
- Design the characteristics of the program according to the requirements of the CGR.
- Implement the CGR prototype in a working environment.
- Assess the effectiveness of the prototype system throughout the process.

For the research into the generalisation of the verifiable randomness concept:

- Perform exhaustive research into previous work in achieving transparency and outreach into communities using open data (or similar) initiatives.
- Investigate the use of zero knowledge protocols to enhance the privacy of large amounts of citizen data.
- Assess different types and origins of inputs in a generalisation of the possible answers to the problem.
- Assess different types of outputs in a generalisation of a wider content.
- Explore different tools mentioned on previous research to recognise those better suited for each part of the class.
- Assess the feasibility of implementation and the impact of verifiable transparency with private data on a real-life environment.

## 1.3 Thesis structure

1. **Chapter 2: State of the Art and Background.** This chapter reviews the concepts pertinent to the development of the client and applications.
2. **Chapter 3: The Randomness Beacon.** This chapter gives a broad overview of what a randomness beacon is and how it works.
3. **Chapter 4: Using Publicly Verifiable Randomness.** This chapter gives a detailed explanation of the client created for the CLCERT beacon.

4. **Chapter 5: Integrating Public Randomness in Real Cases.** This chapter reviews the integration process between the applications generated from the client and the possible extensions to those.
5. **Chapter 6: Extensions to Private Data.** This section reviews some extensions of the main beacon client into private data.
6. **Chapter 7: Conclusion.** This chapter lists what was done, why it is relevant, the conclusions and details possible future work.

# Chapter 2

# State of the Art and Background

## 2.1 Transparency

There are several general definitions of transparency. Following “Understanding modern transparency“[15] we will use a descriptive definition: Oliver [16] indicates that transparency can be described by three elements: an observer, a means or method for observation and something available to be observed. This work aims to provide more availability of the last two since there are always eager observers.

There is an extensive discussion regarding the importance of delivering transparency to public decision-making processes, which extends far broader than the field of computer science. It involves human and social sciences as shown by Grimmelikhuijsen [12] who gives a broader view of the relevance and motivation behind this endeavour.

## 2.2 Randomness, Verifiability and Accountability

A general, non-mathematical, definition of randomness relates to a state which lacks a pattern or principle of organisation; something unpredictable. As for verifiability, something is considered scientifically verifiable if it can be tested and proven to be true. Being accountable refers to something that is required or expected to justify actions or decisions; something that can be explained or understood.

Accountability is the way this work combines randomness and verifiability.

In a seminal work on accountable algorithms [14], Kroll et al. discuss a wide range of design options for adding accountability in decision making, exploring various directions and with several possible applications. Although broader in scope than what we intend to cover in this thesis, their work provides a guide to follow in what adding accountability involves.

In “Accountable Algorithms" [14], Kroll et al. first described the basic ideas of how the

NIST randomness beacon could be used to include randomness into a program that uses only public data. It presents the main ideas behind the need to provide accountability to simple processes, how that can be done, what tools are the more readily available and exploring different extensions of what could be done to improve it.

This is indeed the starting point of our research, which we plan to extend to other types of programs, including those with private data. More recent work, such as the one by Syta et al. [23] has focused on the problem of generating trustworthy randomness by combining two or more random beacons into a single one.

## 2.3 Privacy

Privacy is usually defined as a state in which one is not observed or disturbed by other people. It is important to note that to achieve privacy, Kroll et al. suggest the use of zero-knowledge protocols [11] as a tool in which secret information is crucial to the process can be hidden.

Kroll et al.'s work does not delve deeper into making this process accessible and easy to apply. We claim that this is imperative to make this approach work in real applications.

One such example is in zcash [21], where efficient ZK tools are used [18] to effectively provide strong privacy deriving from the fact that shielded transactions in zcash can be fully encrypted on the blockchain, yet still be verified as valid under the consensus of the network rules by using zk-SNARK proofs. We plan to follow this approach while taking into account the characteristics of the particular private data present in the decision-making process we consider. This means that we will use efficient and practical cryptographic tools to ensure the privacy of the data that is entrusted to the process.

## 2.4 Is Verifiable Randomness Practical and Useful?

One primary concern left unaddressed in previous research is how to generalise the transparency approach. Identifying the type and characteristics of processes where transparency can be successfully implemented in practice, is also an open challenge. Also researching the tools that are currently available to obtain a successful implementation is a major open question in the field.

# Chapter 3

# The Randomness Beacon

## 3.1 The Idea Behind a Randomness Beacon

The idea that supports a randomness beacon is simple: true randomness is hard to get. To have a reliable public source of randomness is incredibly important, there is a latent need for randomness all day in every field. To have verifiable, autonomous, consistent, and unpredictable randomness would offer peace of mind to different applications in dire need of it.

## 3.2 NIST Beacon

NIST, the American National Institute of Standards and Technology, is part of the U.S. Department of Commerce where its mission is to promote innovation and industrial competitiveness. It also has active programs for encouraging and assisting industry and science to develop and use these standards. Under this line they have different projects.

NIST implemented a source of public randomness as one of their projects. The service uses two independent commercially available sources of randomness. The Beacon is designed to provide unpredictability, autonomy, and consistency [13]. Unpredictability means that users cannot algorithmically predict bits before they are made available by the source. Autonomy means that the source is resistant to attempts by outside parties to alter the distribution of the random bits. Consistency means that a set of users can access the source in such a way that they are confident they all receive the same random string.

In an effort to provide transparency and trustworthiness when using the beacon, NIST have encouraged the creation of different beacons around the world. This is done with the thought in mind that if one of the sources is untrustworthy, the others will make it so you can trust in them as a whole.

## 3.3 CLCERT Verifiable Beacon

Under the umbrella of NIST, CLCERT of the University of Chile developed an independent beacon to be a part of the worldwide community for providing randomness.

The beacon generates new randomness every minute by providing a 512-bit random value (or "pulse"). The 512 bits randomness value is generated by an algorithm which provides a mathematical proof that no entity can effectively manipulate or predict the random output. The computation is done using a slow hash function that makes unfeasable that the Beacon administrators can compute the result significantly earlier than the rest of the public.

The generation algorithm consists of three phases.

1. Collection of external entropy: Happens during the first 30 seconds. A large collection of data is recovered from different inherently random sources (as shown in Figure 3.1: Twitter, Radio, National seismologic services, etc).
2. Use of Local Entropy: Using a physical device, a TRNG (True Random Number Generator), local entropy is generated.
3. Cryptographic functions and Publication: The external entropy, the local entropy, and some additional information of the system are linked into a single value and signed by the Beacon administrators. The signature is concatenated and the result passed through a slow hash function. The process begins again at the start of the next minute.

In Figure 3.1 we can see how the CLCERT beacon works and how it is able to provide the qualities it ensures.

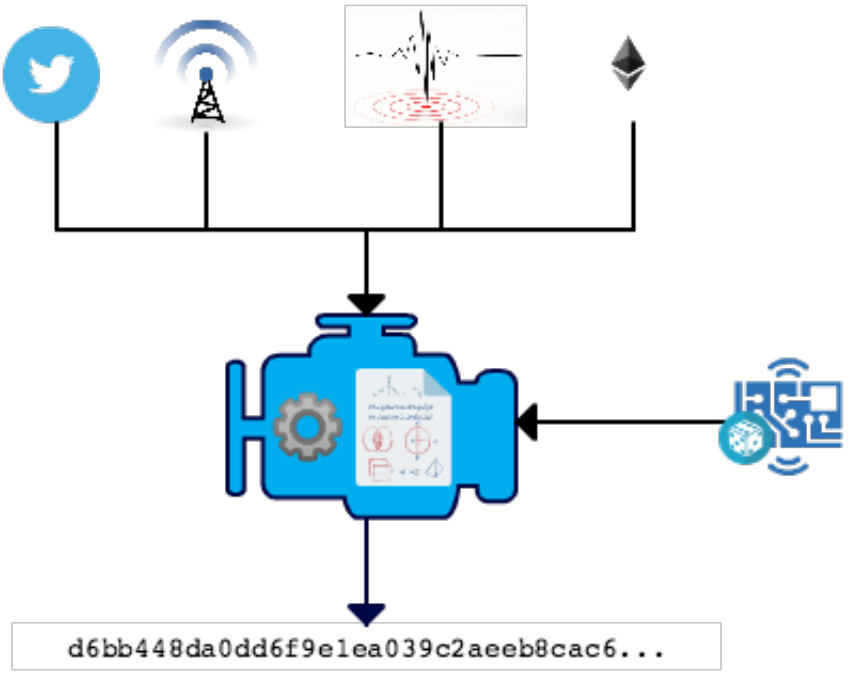


Figure 3.1: How the CLCERT beacon works

# Chapter 4

# Using Publicly Verifiable Randomness

With the implementation of the CLCERT beacon at Universidad de Chile, there was a need to create a client that was versatile and sufficient to use. The motivation behind the use of a publicly verified library comes from the necessity that different organisms be accountable and transparent.

Accountability becomes a necessity in an age where every decision has to be accounted for. For example, in Chile the school system needs to randomly assign a school in a district to a list of students. Because there are more students than spots available and while every parent is invested in providing their child with the best possible school, the public organism in charge has to have some way of proving to them, and to anyone interested, that there was no foul play or undue influence on their part.

The advantage of using a system like this is that it takes away the burden of having to prove correctness from the organism trying to deliver transparency.

Relying on a mathematically provable application that functions on top of a certified random seed, on a robust pseudo-random generator, allows every party involved in the process the chance to feel (and be) reassured to take action in the most transparent way possible.

As an example: consider the case when an auditing agency needs to select a certain number of people to audit. Given that they have limited resources to go ahead with the actual auditing, they have to choose in some way how they're going to spend those resources. One way to provide transparency, whilst keeping the decision fair, is to make this decision public for auditing as well.

There are three parts to the developed library. The **seed retrieval** process, the **random selection** process and the **verification** process of correctness. They are depicted in Figure 4.1.

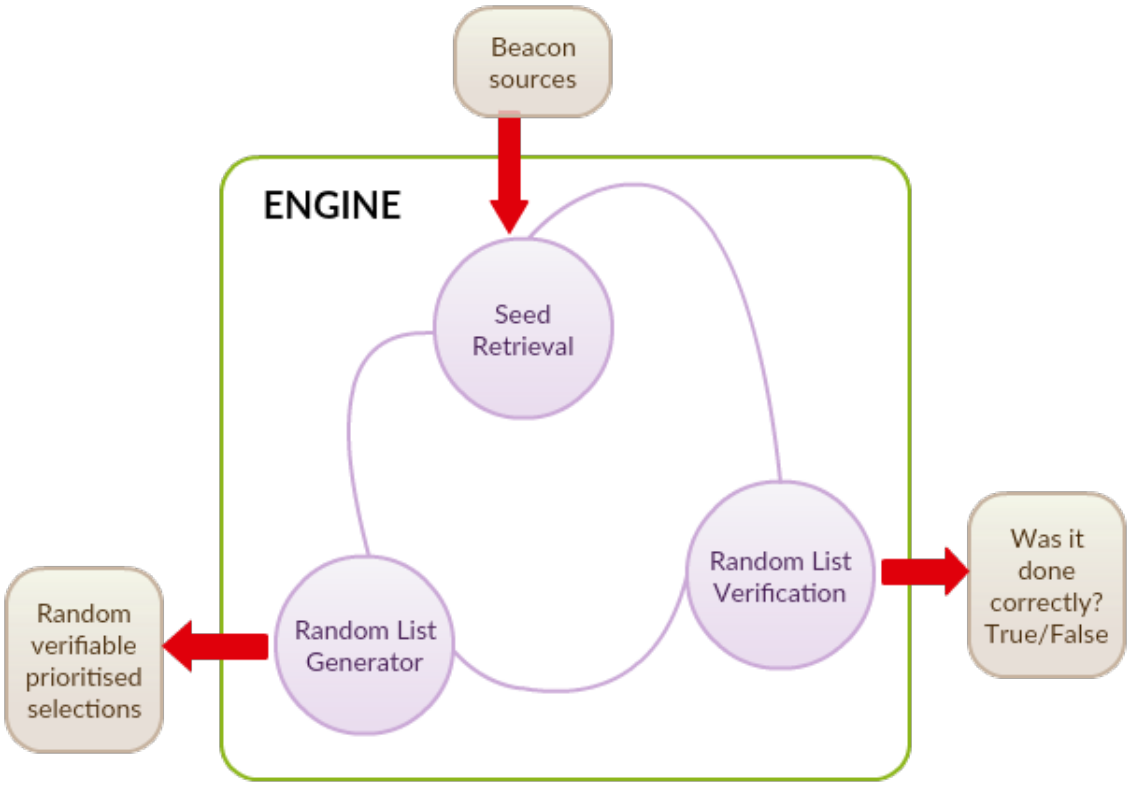


Figure 4.1: Client packages, outputs, and inputs

Each part will be explained in the following sections.

## 4.1 Seed Retrieval Process

Retrieving the truly random value used for the seed is a crucial step in successfully delivering transparency using this method. The random value is retrieved from the beacon sources and processed. This is time to start thinking about what will be needed for verification later on. The seed retrieval package manages several scenarios in which the output value is harvested.

### 4.1.1 Requirements

- The randomness extraction has to be able to communicate with different beacons.
- It should be able to easily add more beacon sources as they become available.
- It needs to connect to each source securely, with confidentiality and auditability.
- It should deal with foreseeable exceptions generated by the response.
- It should indistinctly receive responses regardless of their format.
- It should be able to combine output values from different sources.
- It should verify the integrity of the source.

– There should be an output that can be used as a seed.
– The output should be standard to use.

### 4.1.2 Design

There was a division of each case scenario currently available. The package has four different scenarios, depending on the availability of the beacons as seen in Figure 4.2.

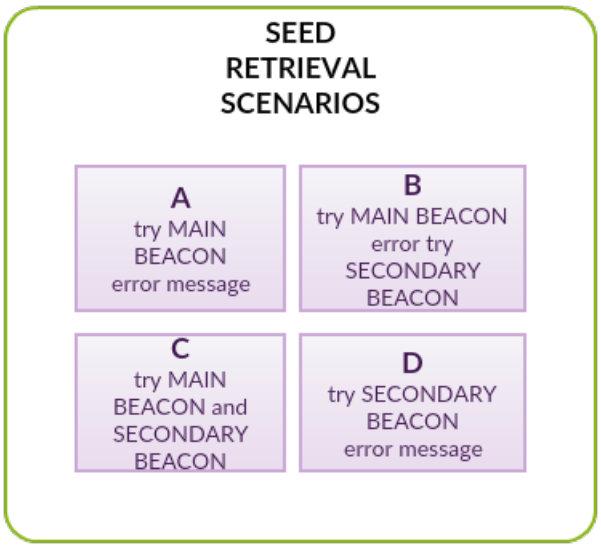


Figure 4.2: Scenarios in Seed Retrieval considering only two beacons (main and secondary).

Each scenario manages a different case of seed retrieval.

#### Scenario A

In this scenario, the seed is retrieved from the data given by the beacon hosted by CLCERT and Universidad de Chile. We connect to the next possible response after the timestamp we took from the moment the Seed Retrieval object was created. If there is no response for 10 minutes, an exception is thrown letting the user know that there was not a correct response available. In case of failure the user of the library should change scenario of use or try again at a later time when the desired beacon is available again.

The response from the CLCERT beacon comes in JSON format, and the package is apt to manage that type used in the first version of its development.

#### Scenario B

In this scenario, the seed is also retrieved from the data given by the beacon hosted by CLCERT and Universidad de Chile. It has the same behaviour, but it only differs from

Scenario A when the 10 minutes of waiting for a correct response passes. In this case, an attempt is made to retrieve a seed from the beacon hosted by NIST, using Scenario D.

### Scenario C

In this scenario, the seed is a combination, XOR for example, of two different sets of data. We attempt to take data from both available beacons so far (CLCERT and NIST) by using the created scenarios (A and D, respectively), and after each seed is harvested, a combination operation is performed. If either of the two sets of data is unavailable, only one is used, and a log of this is recorded in the bundle package that the Seed Retrieval object renders when the user asks for it.

It is important to note that each beacon (until the moment this thesis was written) uses a different configuration to format their response data. Consequently, our system must implement some protocol to parse each beacon output.

### Scenario D

In this scenario, the seed is retrieved from the data response given by the beacon hosted by NIST. It behaves like Scenario A; it asks for the next available response from what the current timestamp is and waits for a maximum of 10 minutes for a correct response on NIST's part. After that time has passed without a response, an exception is thrown warning the user. In case of receiving the exception, the user of the library must choose another scenario that uses another beacon or wait until the desired beacon is available again.

The response from the NIST beacon comes in XML format; the developed library is apt to manage that type of response since it is the first working version of the NIST beacon.

### Other Scenarios

By the time this thesis was written and the library implemented, there were only two beacons in operation. In the future, when there are more available to be used, there can be scenarios where there are more back up beacons in case there is a failure or there can be more inputs to be used for the combination operation.

The extended scenarios could be to generalise the ANDs and ORs, choose `k` out of the total `N` beacons available, choose any beacon randomly or just choose some of them.

As stated before, the combination operation used was an XOR, this is by no means the only one that can be used in this case, but for simplicity it was the only one used.

### 4.1.3 Implementation Specifications

The primary goal when developing the seed extraction part of the engine was to make it interchangeable. The final user does not need to know or worry about how to use all the different beacon sources available, so the primary concern was to produce a well-documented and straightforward package.

Currently, there are 4 different scenarios, and they all extend from the main class that wraps them into a single seed extractor.

Each scenario manages the communication with the beacon and the behaviour previously described. Each one has two constructors, one that uses the current timestamp and one that gets the timestamp as an input. This is made so the verification of the seed can be made possible.

The different methods present in each scenario have been designed to be used for verification once the bundle is given in the selection process package of the engine.

There is also another class that manages a secure connection to the beacon response. It uses the native Java library, HttpClient. In Figure 4.3 is possible to get the whole picture.


Figure 4.3: UML diagram, SeedExtraction and scenarios

## 4.2 Selection Process

The crucial part of this engine is the selection process. In this process, the goal is to use the seed to create a new random subsection of the input data list. Using a seed to generate a pseudo-random selection means that the step is reproducible given the seed. This reproducibility allows for third-party observers to recreate the process.

### 4.2.1 Requirements

- It must not depend on what the potentially selected values are, but it must depend on a key identifier associated to the client, so different clients generate different pseudo-random sequences even if started from the same seed.
- It must give versatile options for output.
- It needs to generate an unpredictable sequence for anyone who lacks the seed.
- It has to return all the information needed to replicate the procedure.
- It must be able to receive priorities and weight the random output on those priorities.
- It must receive a configuration for the prioritised output.
- It must be easy to extend to different types of output needed.
- It must be self contained. This means that it should do all the operations that are needed for the desired output within the library.

### 4.2.2 Design

Given the requirements raised for the selection, several design choices were made at different stages of the development process.

#### Using a PRNG

Truly random sequences are difficult to generate, but demand for them is increasing exponentially. This means that a faster and more approachable way has to be found. For this purpose there are pseudorandom generators (suggested and developed by Blum and Micali and Yao [1]). They are efficient deterministic programs that expand a randomly selected `k`-bit seed into a much longer pseudorandom bit sequence that is indistinguishable in polynomial time from an (equally long) sequence of unbiased coin tosses [10].

Pseudorandom number generators (PRNGs) are algorithms that generate a sequence of numbers whose properties approximate the properties of sequences of random numbers. The PRNG-generated random sequence is not truly random, because it is completely determined by an initial value, called the seed of the PRNG.

For this work we used the cryptographically secure pseudo-random number generator Fortuna [7] designed by Ferguson and Schneier. Being cryptographically secure means that

Fortuna can be used in many cryptographic applications including digital signatures, authentication protocols and data encryption, which is the reason it was chosen for this design. The implementation used was developed by Ingo Bauersachs [1].

There are three parts to Fortuna. The **generator** takes a fixed-size seed and generates arbitrary amounts of pseudorandom data. The **accumulator** collects and pools entropy from various sources and occasionally reseeds the generator. Finally, the **seed file** control ensures that the PRNG can generate random data even when the computer has just booted. In Figure 4.4 we can see how each part of Fortuna interacts with each other.

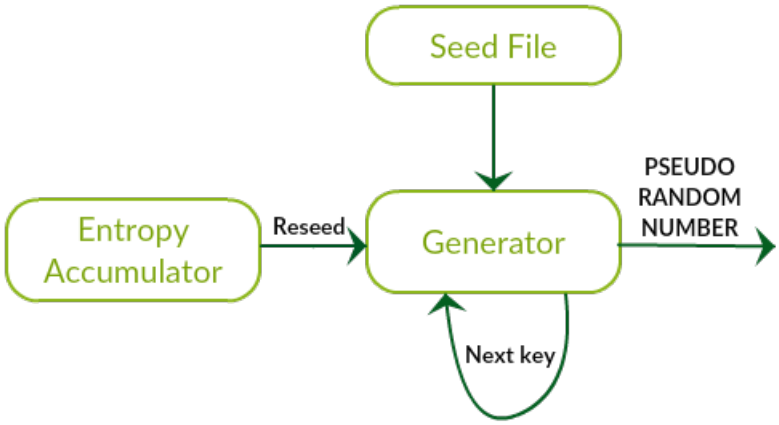


Figure 4.4: Quick overview of the *Fortuna* PRNG

### How Much Randomness?

The beacon implementation generates full-entropy bit-strings and posts them in blocks of 512 bits every 60 seconds, the time it takes the beacon to get enough entropy. The decision on how much randomness from the 512 bits to use was made based on how many entropy bits were needed by Fortuna to operate: this was 10 bytes of entropy.

### Specialisation of the Pseudo Random Sequence of the Client

As the clients that use the libraries grow, there can be a time when more than one client wants to use the beacon response as a seed for their random selection. This problem gave way to the design decision of using an identifier per client.

The identifier can be any sequence. For the development of this work an identifier was chosen and given to CGR, the studied client.

To generate the new sequence the secure hash algorithm, SHA 256 is used. Each value is converted to byte arrays, then the SHA 256 of the concatenation of the byte arrays is calculated. In Figure 4.5 there is a depiction of how it works.

[1] `https://github.com/jitsi/bccontrib`

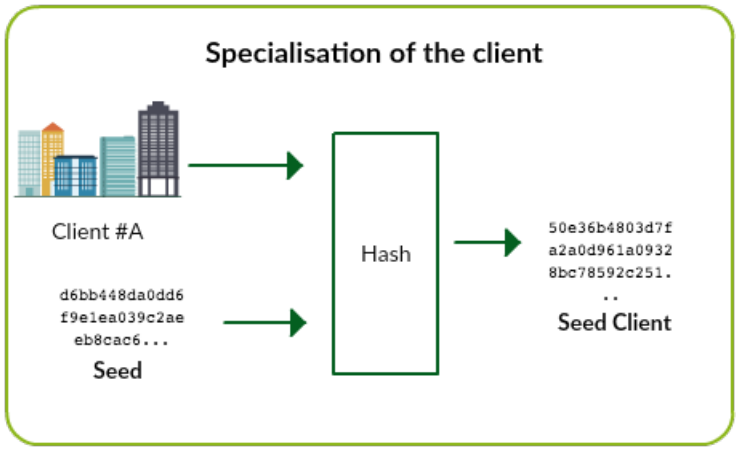


Figure 4.5: Each client gets their own seed based on their client identifier.

### Dynamically Extending the Pseudorandom Sequence

There are two main types of selection: **simple** and **composite**, as seen in Figure 4.6. Only these two types were developed given that they fulfilled the necessities of the main client as well as a vast array of situations.

The **simple type**, refers to a standard selection process with only one list for input and a value to be extracted for output.

The **composite type**, refers to a prioritised selection process in which there is more than one list as an input, and a configuration is needed to determine the output.

The simple type is a special case of the composite type and it was generated to deliver simplicity in use.

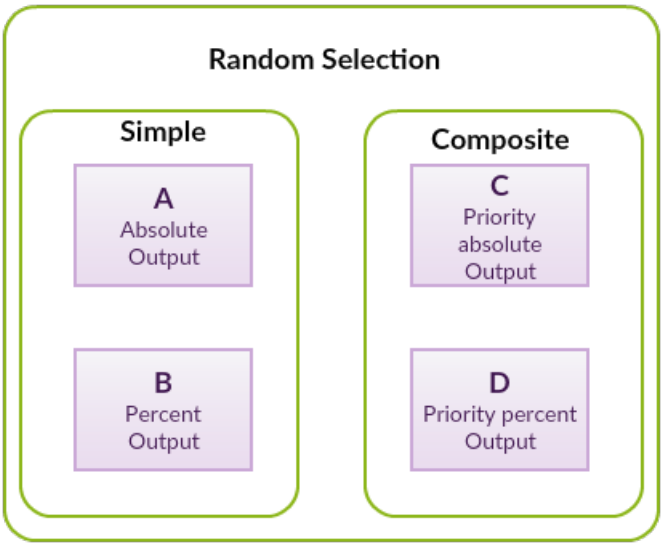


Figure 4.6: Types in Random Selection

### Simple Type

The simple type is the most straightforward one. It is designed to be general enough that it fits most cases when there are no restrictions to the selection, and there is only a need for a simple, verifiable selection within an initial list. The basic functioning is depicted in Figure 4.7.

Within the *simple type* there is **type A** and **type B**; each receives an input list of identifiers and a single desired output.

**Type A** receives an integer value as the desired output. This number represents an absolute quantity of rows in the original list to be selected in the output. **Type B** receives an integer value as the desired output as well. In this case, the number represents the percentage of the total of rows in the original list to be selected as the output.

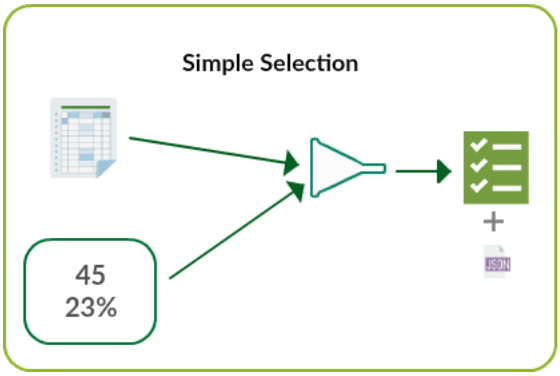


Figure 4.7: Simple type: Basic functioning

#### Composite Type

The composite type takes the configuration of the output one step further than the simple type. It is designed to increase what the user can outline in the final output. It allows for different priorities associated to weigh the final selection process. The basic functioning is depicted in Figure 4.8.

Within the *composite type* there is **type C** and **type D**. Each receives an input list of identifiers, an input list of priorities and a configuration array of integer values for desired outputs.

**Type C** receives, in its configuration array, all integer values associated with absolute values for each priority. **Type D** receives, in its configuration array, all integer values associated to percentages of quantity for each priority.

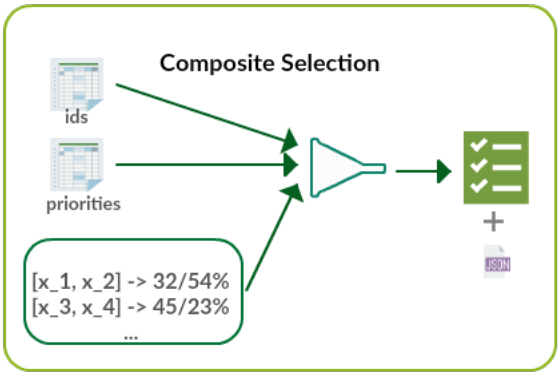


Figure 4.8: Composite type: Basic functioning

### 4.2.3 Implementation Specifications

The development was incremental and iterative, given that the entire engine is connected and feeds other parts of the system.

In both types there is a parent class designed to manage the selection part, which holds the values needed to make the selection. Then there is each extended class that depends on its parent to make the decision.

The subordinate classes take care of the creation part of the process, assigning the different values that will be needed after to make the selection.

Each subordinate class has two different constructors. One is the primary constructor, designed to communicate with the beacon sources and wait for a seed to populate the PRNG. The second constructor is for the verification; once there is already a seed to verify, the constructor receives the data extractor as a parameter to emulate the same selection.

Each constructor is also in charge of verifying the correctness of the boundaries of the

values and throw an adequate exception if something goes wrong.

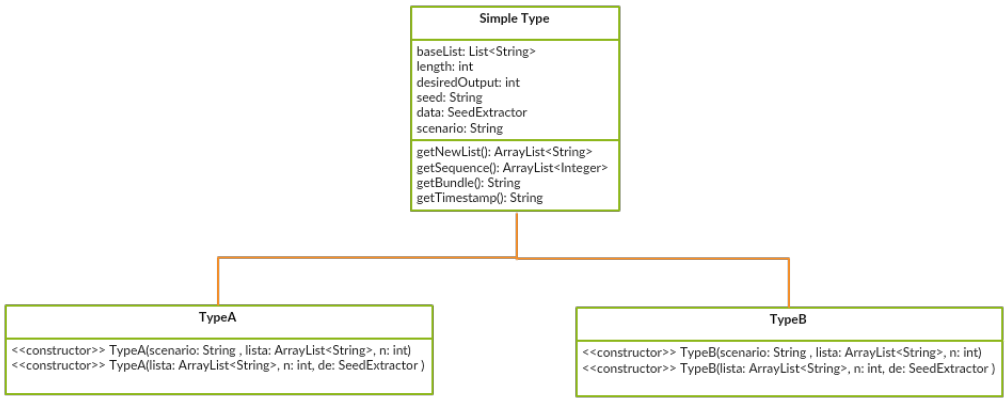


Figure 4.9: UML diagram, Simple

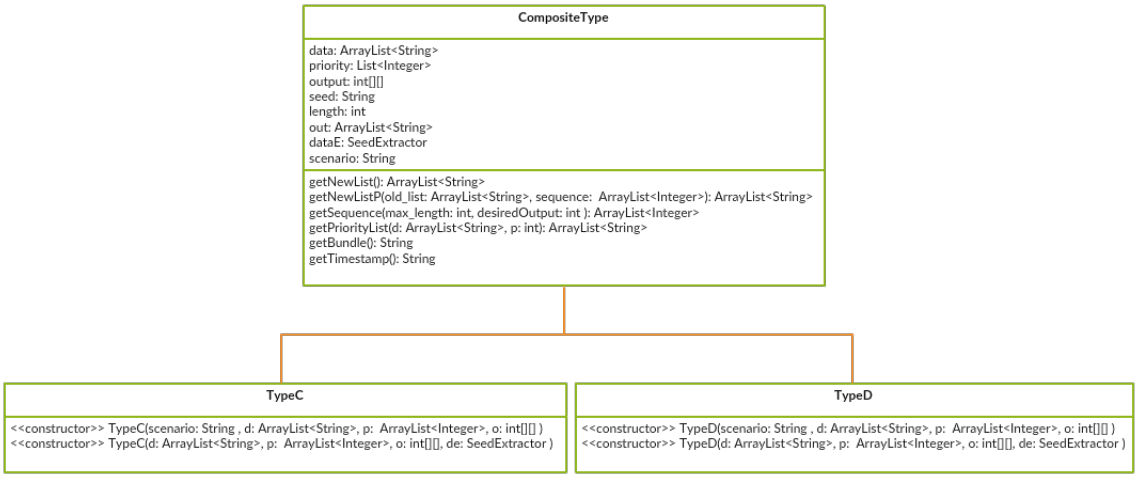


Figure 4.10: UML diagram, Composite

As we can see in Figures 4.9 and 4.10 there are similarities in the design and use. A user calling any of the selection processes is like this:

```
TypeC c = new TypeC(scenario, ids, priorities, conf);
ArrayList<String> randomOutputList = c.getNewList());
```

The process is to create the object of the desired selection type and then call the method `getNewList()` to obtain a result.

There is also the method `getBundle()` available in every selection type. This method returns a String that contains the needed configuration to replicate the process. This string is in JSON format. This configuration is sufficient, alongside the input list, to account for the whole process. The configuration values for both the Simple and Composite types can be seen in Tables 4.1 and 4.2 respectively.

| Name | Value |
|---|---|
| Seed | Used for the PRNG to generate numbers, it is the one given by the seed retrieval part of the engine. |
| Timestamp | It is the timestamp used for the retrieval of the seed. |
| SelectionType | It states wether it is type A or type B. |
| DesiredOutput | It is the absolute number of desired output, when in verification every process is treated as an absolute quantity. |
| Scenario | It is the scenario used in the seed retrieval part of the process. |

Table 4.1: Bundle format for simple type

| Name | Value |
|---|---|
| Seed | Used for the PRNG to generate numbers, it is the one given by the seed retrieval part of the engine. |
| Timestamp | It is the timestamp used for the retrieval of the seed. |
| SelectionType | It states wether it is type C or type D. |
| OutputConfiguration | It is the array used in the configuration. With the mapped priority and the absolute quantity desired for output. When in verification, every quantity is treated as absolute. |
| Scenario | It is the scenario used in the seed retrieval part of the process. |

Table 4.2: Bundle format for composite type

In the composite type, the output configuration array is a two-dimensional integer array. In the first dimension it has the priority number, and in the second dimension, it has the quantity (absolute or percentage) of the desired output for that priority.

### 4.2.4 General Cases

In the previous sections we have explained the specifications for this implementation of a particular program that takes the input and is able to generate an output that can be verified. This is not the only possible way; a general approach to the solution is given in Figures 4.11 and 4.12, for the selection and verification processes respectively.

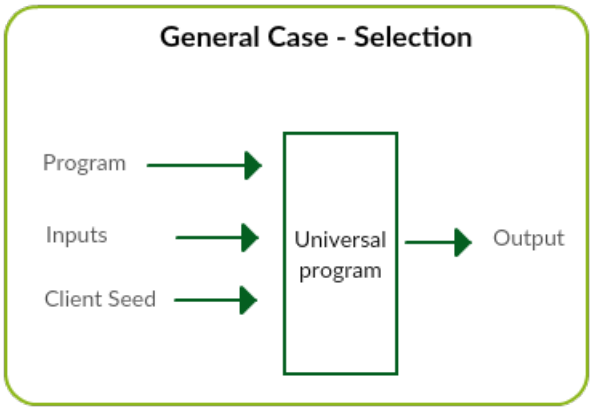


Figure 4.11: General case, selection.

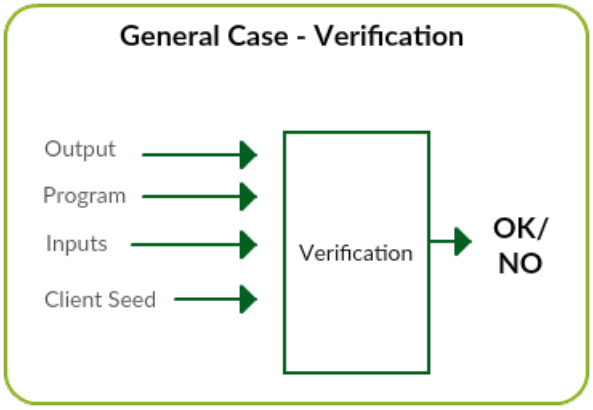


Figure 4.12: General case, verification.

## 4.3 Verification Process

A relevant focus of the engine is its ability to verify the veracity of the result at a later point. This is possible since the sources the engine uses are independent and reliable in keeping their data stored. This is what makes sense in using these beacon sources for verifiable randomness. It is possible to capture the seed if the timestamp used was saved.

It is important to note that it is possible to find out right away if there was any tampering with the responses given by the beacon since they are signed, and this is checked to assure its integrity.

### 4.3.1 Requirements

– It must be able to reproduce the exact process as it was done the first time.

– The parameters needed are to be fixed.
– The output must be `True` or `False`.

### 4.3.2 Design and Implementation

The design of the package was derived from the development of the previous two packages described.

Given the processes that were already developed, it was necessary to take the configuration bundle created in the output and perform the same process to check whether the results were the same or not.

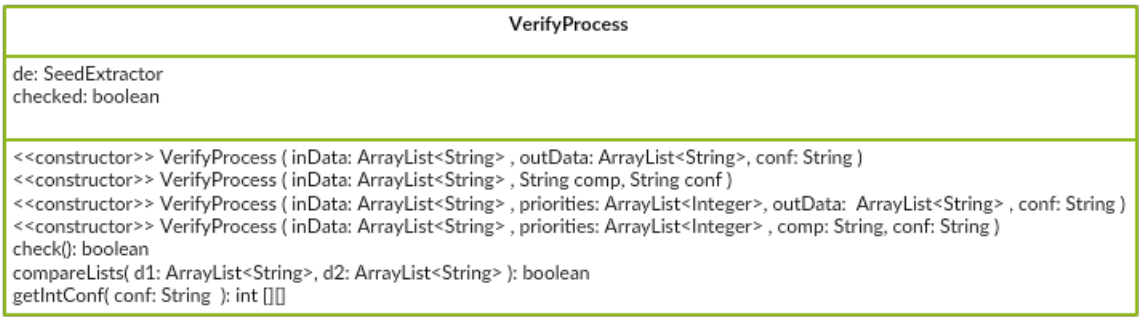


Figure 4.13: UML diagram, VerifyProcess

In Figure 4.13 is observed that there are two auxiliary methods, one to verify the response lists and another to transform the *String* configuration into the *int* array needed, and the public one `check()` that stores the value of the verification.

These four different constructors check a different angle of verification and cover the grounds for usage:

- Verify an **entire list** that is the result from a **simple** process (typeA or typeB).
- Verify a **single value** that is the result from a **simple** process (typeA or typeB).
- Verify an **entire list** that is the result from a **composite** process (typeC or typeD).
- Verify a **single value** that is the result from a **composite** process (typeC or typeD).

The package is implemented and designed to only differ in use by creating the object differently. A final user can always find out which process to create by accessing the information in the configuration bundle and create the appropriate object.

The Boolean function that shows the result is the same since it is a class variable that stores that value at the moment of creation.

Any final user of the engine would use this verification process in the following way:

```
SeedExtractor extractor = new SeedExtractor("A", timestamp);
TypeC tc = new TypeC(ids, priorities, conf, extractor);
VerifyProcess c = new VerifyProcess(ids, priorities,
                  initialOutput, tc.getBundle());
c.check() // This returns true or false.
```

# Chapter 5

# Integrating Public Randomness in Real Cases

It is of utmost importance to clarify why verifiable randomness provides transparency, and why a public organisation would want to use it.

Whenever an organisation (public, private or of any kind) launches a random selection using public randomness, there is more than just a choice of a particular engine with which to do this. By choosing a verifiable source, such as the one designed in this work, they commit to the output value. Every client using an approach like this one commits to the veracity of the entire process.

The commitment is possible because the back end sources are reliable in their output. If the engine fails to confirm the correct signing of the output, it does not use it. It is up to the public to see if there has been any tampering with the output, and every single pulse is linked to ensuring correctness.

Besides the reliability that the back end source provides, there is the intrinsic part of the design at the moment of fulfilling a request for public randomness. The client either launches a process in this instant, or it programmes the next launch. In any case, the outcome of the procedure is sealed, and it will be public for everyone to check in the moment of launch and in posterity.

This small yet powerful concept yields importance to the decision of using such a system to make random, particularly public, procedures.

CGR is taking a big step towards providing transparency to all the public that they serve. By allowing conceptual research to take place in one of their newer systems, they are leaping into the future by giving a tool to be held accountable.

So far the development explained in the previous chapter has been referred to as the **engine**. As presented in the introduction to this thesis, Figure 5.1 illustrates the abstraction for further understanding.

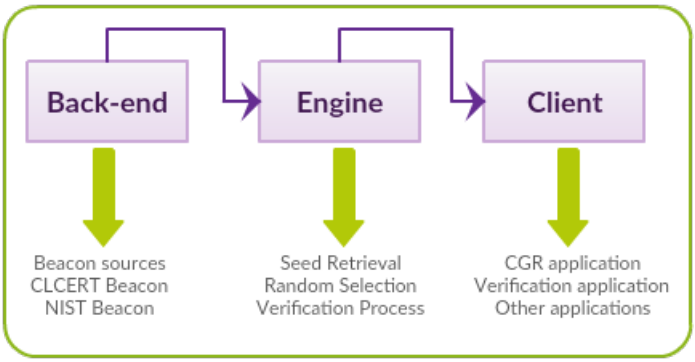


Figure 5.1: Full picture design.

## 5.1 Comptroller General of Chile

The Comptroller General of Chile is the superior organism for the auditing of the Administration of the State. It is constitutionally autonomous from the Government of Chile. The Chilean constitution backs its functions.

In late 2016, new legislation came into effect. Law 20.880 deals with probity and amongst many things forces the declaration of interests and assets from all public employees that fulfil certain criteria. This law also allows for audits of the declarations, a previously forbidden side for CGR. This declaration is commonly referred as **DIP**, the Spanish acronym for *Declaración de interés y patrimonio*.

The exact criteria of the people mandated to declare is complicated and irrelevant for this research. It is significant to remark that it involves all high ranking officials that are employed by the state, elected by the voting system or merely employed by a public organism.

In most cases, there is one month for all declarations to be sent. After this, there is an assessment of the correctness of the declaration itself, and then it is up to CGR to select which ones are to be audited. The selection of people that declare provides context for the importance of this particular selection.

In 2017, approximately 85.000 declarations were sent. It is of no surprise that CGR does not have the ability to audit all declarations. At the same time, the CGR needs to enforce the law that states that any declaration may be audited.

There are two sides to the selection process and as the organism auditing not only the Government but every public branch, it is necessary to state the primary goals of this selection.

1. Any individual that sent its declaration can be sure that the auditing selection was fair and transparent.
2. Any member of the public can trust in the selection process, no individual was out of the pool to be chosen from this.

With those two primary goals in mind, and the ever present mission of CGR to deliver transparency and accountability, the design for their client application was made. This design is explained in the next section.

### 5.1.1 Design

When discussing the design and implementation of the client for the problems presented there were several contingencies to be dealt with.

There are two principal scenarios of selection to be considered:

- First, there is a group of people, and the random selection applies to all of them. As seen in Figure 5.2.

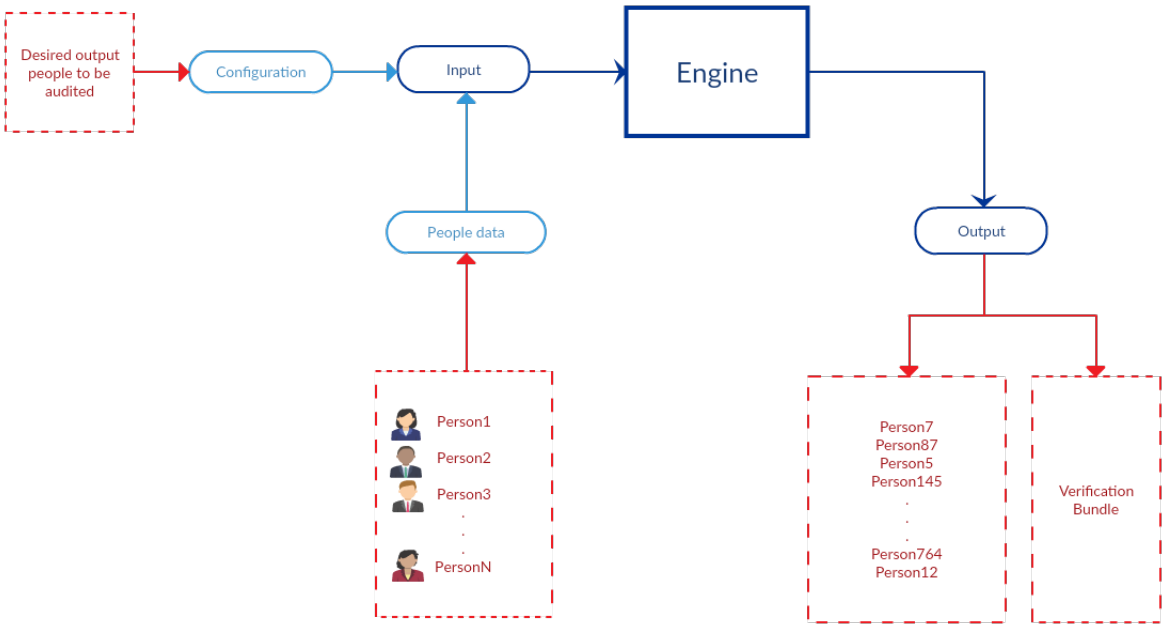


Figure 5.2: Simple random selection

- Second, there is a risk value assigned to each person in the group, and the random selection applies differently to each range of priority. As seen in Figure 5.3.

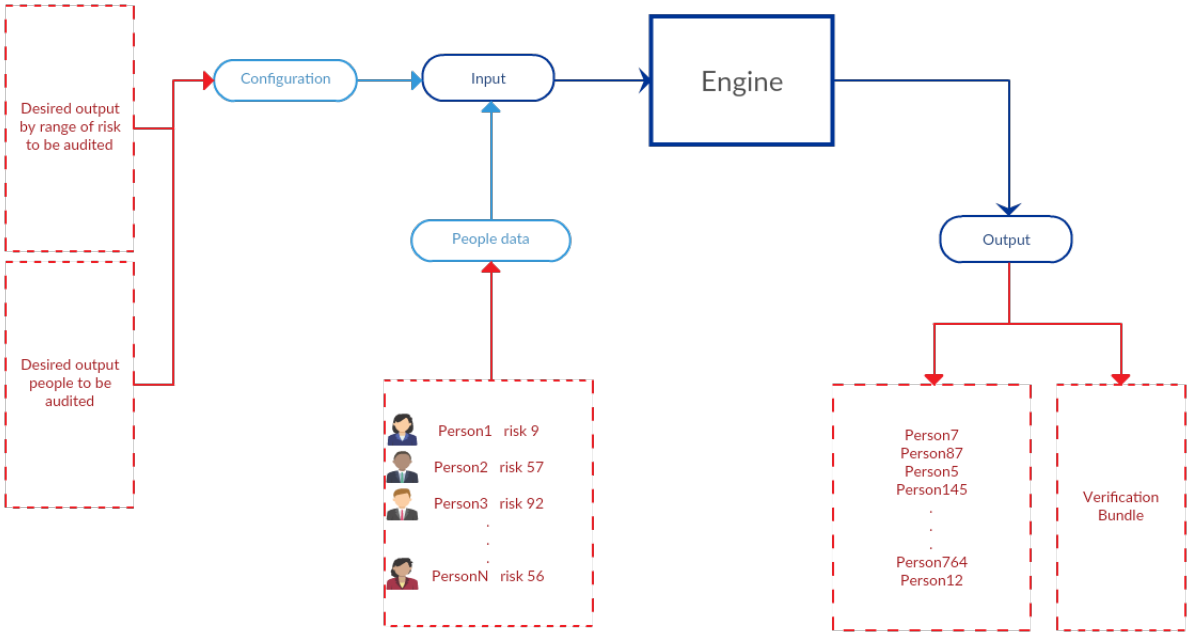


Figure 5.3: Random selection with risk factors

The first case is uncomplicated. CGR has to map a single identifier to a person in the list of declarations, and that is used as an input to the process. The engine generates an output list and a verification bundle.

The second case deals with the use of risk factors. These risk factors are calculated from several elements present in the declaration and are continually changing. Given the limited amount of time and resources available to audit all of the declarations, the need arose to select in ranges.

It is prevalent that everyone is exposed to the possibility of being audited, no matter how low his or her risk factor is. This means finding a fair way to do this while remaining an impartial decision. That is where the ranges come in to place. In the application, the ranges are not set and are malleable as a configuration input alongside the list of identifiers and risk values.

In Figure 5.4 is an example of how this model would work in a simple case.

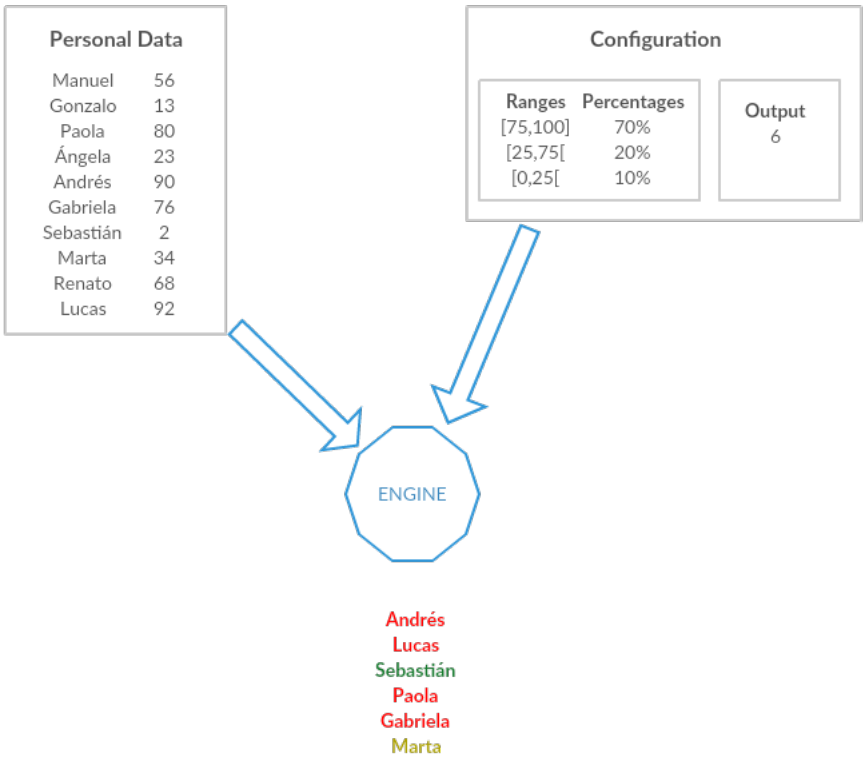


Figure 5.4: Example of risk value model

### 5.1.2 Development and Implementation

This client application was developed to be integrated immediately with the current systems working at CGR. The project was closely monitored and worked on with the development teams at CGR and under their quality standards.

The client was developed using Java 9, JSF 2.2 with Primefaces 6.2 in a Tomcat server. The engine is implemented as a jar file and all libraries used are freely licensed under the *GNU Lesser General Public License* [1].

## 5.2 Verification Application

The most crucial part of the process is the verification side. It is what delivers transparency and by definition should be available to any member of the public that wishes to test the accuracy of the selections made by any of the applications that use the engine.

An impartial organism must host the verification application. In our case, the University of Chile alongside CLCERT also provides this service (in theory, any trustworthy organisation

[1] `https://www.gnu.org/licenses/gpl-3.0.en.html`

could do it). It uses the verification process part of the engine to compute the same random selection that is indicated by the input and configuration files provided.

The verification client is generalised; it was not designed to work only for the CGR client. It is designed to receive the necessary files to compute the selection process and then compare that result to the one being tested against.

It needs the original input file(s) used for the random selection process, the configuration bundle resulting from that process and the output that the user wishes to compare.

For example, in the case of a prioritised selection, the user would need to provide a set of inputs. The input file with the identifiers; the input file with the associated priorities; the configuration file containing the timestamp used, scenario, type of selection, etc.; so it could compare it with a single identifier that they wish to know if it was chosen or the entire output list, to check the correctness of the selection process in full.

## 5.3 Other Applications

There are many possible client applications to be implemented using the studied engine. There is a wide range of areas in which this concept can be applied to impact citizens all around the world.

The characteristics that a process needs to have to be eligible to be implemented are fairly simple:

1. To have a random selection within their processes.
2. That the people designing it have the desire to make their operations and actions accountable to an external group of people.

Openness of the data is not required, which will be discussed in the next chapter. It is not necessary to disclose all information in order to maintain transparency.

Here are only three examples of possible applications. These examples vary in scope and capacity, but they all point to the nature of a random selection that will be publicly scrutinised.

The *diversity visa lottery in the US* was instituted in 1995 and offers a chance for people from qualifying countries to get a green card. It was established by the Immigration Act of 1990[2]. After all eligible candidates are sorted through, they randomly select 50000 recipients to be given an immigration visa.[3]

*Public school auditing.* In many countries, including Chile[4], students apply to several

[2] `https://immigration.laws.com/immigration-act-of-1990`

[3] This example comes from "Accountable Algorithms", published by Kroll et al.

[4] Law 20.845 mandates that the selection cannot be discriminatory. `https://www.leychile.cl/Navegar?idNorma=1078172`

schools that they desire to enter. Since the capacity is not enough to cater to every applying student, the governing body makes a random selection among all applicants.

*Randomly chosen members of a polling station.* In Chile, there is a national vote every two years. In every polling station, there is an ample number of voting tables. Those voting tables are overseen by at least three people that are chosen from the pool of people that vote at that table by a committee. This pool of people is narrowed down to an unspecified number and then the supervisors are randomly selected. This appointment is binding, and the law enforces it.

As was stated before, these are concrete examples. It is relevant to note that they do not necessarily have to be in a government organisation.

# Chapter 6

# Extensions to Private Data

## 6.1 Motivation

There is a more challenging case side when considering verifiability in random selection processes. If the process depends on data that cannot be publicly exposed, achieving transparency is not straightforward. Yet having transparency and privacy is still possible. Consider the following case: there is an agency that gives social service aid to people based on their income bracket. Since funds are limited, beneficiaries are randomly selected from among all qualifying individuals. The organisation needs, however, to prove that their selection process is correct.

What is the difference between this problem and the problem stated in previous chapters? Here the agency can not reveal the household income or citizens private and sensitive information.

We will use cryptographic tools to get around this restriction.

## 6.2 Characteristics and Scope

There are two roles that we are interested in; the **organisation** that needs to execute a process and the **verifier** that needs to check that process.

More concretely, we want an algorithm that given certain data, some of it private, generates a proof. This proof will be taken by another algorithm as input and will produce a result: either the proof is correct or it is not.

Since in this case the data used in the process needs to be kept private, the current implementation of the engine client is not sufficient to hide this data. Being in the private data domain the work of the engine client is now much harder. It may seem that the engine client is not able to hide the private information as the proof is public. It indeed might sound

like an impossible task, but recent advances in cryptography have allowed for increasingly more sophisticated tools to accomplish this.

**On using advanced cryptography vs. developing applications that target realistic needs:** It is certainly true that more sophisticated cryptographic tools that may allow private information processing like the one described here (for example, using secure multiparty computation [9]) could be designed. Indeed, we could have focused our research on designing and developing them. The original goal in our work, however, was to provide a practical solution that could be deployed in the short term in real (government) organisations, following these organisations' computations and design paradigms. These organisations were dealing with transparency issues already, so any solution could not wait for new architecture or computation paradigms. That is why our work in the rest of this chapter shows, by taking some selected advances cryptographic tools, how to combine them in a realistic service that meets the organisation's demands of transparency in a practical solution.

Taking into consideration the proposed architecture model, it is important to note that modifications need to be made to the engine; the solution mentioned in previous chapters. In practice, this means we will design a new engine which will be able to manage private data.

In the following section, we will show the necessary cryptographic tools to implement this solution.

## 6.3 Cryptographic Tools

### 6.3.1 Commitment Scheme

The notion of commitment is crucial to almost all modern cryptographic protocols. A commitment is a digital safe, a place to store values in such a way that we cannot lie about those values in the future (if challenged). In plain words, making a commitment means that a player in a protocol is able to choose a value from a set and commit to their choice in such way that they can no longer change their mind. The key part is that they do not have to reveal their choice [4].

Let us use an example, say Alice bets Bob he cannot pick the best football team in the world. So at the start of the tournament, Bob writes his choice for the winner. He writes it down on a piece of paper and puts it in a safe to be kept secret and gives it to Alice but keeps the key. He can decide to open it at the end of the tournament or not, but he can not change the choice he made at the start. He committed to the chosen value.

There are two basic properties that are essential to any commitment scheme:

1. Since Bob gave away the safe, he can not change his choice. As a consequence, when the box is opened everyone knows that he made that choice at the start of the tournament.
2. When Alice receives the safe, she cannot tell what is inside unless Bob decides to give her the key.

A formal mathematical formulation of these properties can be found in Appendix A

### 6.3.2 Zero Knowledge Proofs

Zero knowledge proofs were first introduced in the paper "The Knowledge Complexity of Interactive Proof Systems" by Goldwasser, Micali and Rackoff [11]. The notion of zero knowledge is simply defined as a method that allows you to assert the truth of a statement without revealing how you know the secret that notes the statement true, or what that secret is. One party can interact with another party and provide proof of knowledge without revealing their confidential data.

Let us give a classic example[1]. Let's suppose Alice is colourblind and Bob has two cards, one green and one red. Alice insists that they are the same so Bob wants to convince her that the cards are in fact different colours. Bob says to Alice that she should hold the cards behind her back, then Alice shows one to Bob and hides it again. Bob says to choose one of the cards again to show him and he will tell her whether she changed the one she chose before. Given enough repetitions of this experiment Alice, who understands a little about probabilities, is convinced that the cards are different colours.

The former example illustrates the three basic properties of a zero knowledge proof:

1. **Completeness**: If the statement is true and the cards are different, Alice will be convinced of this fact given the proof provided by Bob.
2. **Soundness**: If the statement is false and the cards are the same, Alice cannot be convinced by Bob about the different colours.
3. **Zero-knowledge**: If the statement is true, Alice does not find out the colours of the cards, she is only sure about the fact that they are different. She has learned nothing from this exchange.

The first two properties are basic in an interactive proof system. An interactive proof system consists of a prover and a verifier. The goal of the prover, Bob, is to convince the verifier, Alice, of the validity of an assertion. The objective of the verifier, Alice, is to accept or reject the assertion based on the information gathered in the exchange.

A formal mathematical formulation of these properties can be found in Appendix A.

There is also another type of zero-knowledge proof, **non-interactive ZK proofs**. Say Bob and Alice live in different countries, to prove to her the same statement he would give Alice all of the exchanges in advance, together with the statement to prove. Alice on her own can assert the validity of the statements and the proof without having to be in the same room as Bob. They only need to agree previously on a set of rules to use [20, 2].

[1] `http://www.wisdom.weizmann.ac.il/~/oded/poster03.html`

### 6.3.3 Range Proofs

As one of the many applications that zero knowledge proofs have, there is the one that it is used in this work. Being able to give a proof that a secret lies within a specific interval, this is called a **range proof**. Strictly speaking range proofs are a form of commitment validation that allow anyone to verify that a commitment represents an amount within a specified range, without revealing anything else about its value.

For example, using range proofs Bob is able to prove to Alice that he is old enough to be in his thirties, without specifying exactly what year he was born.

Recently a new range proof was developed. Peng and Bao published their efficient range proof scheme within the *Ethereum*[2] research and development team. Their scheme only needs a constant cost, therefore the new range proof scheme is more efficient than the existing range proof schemes. Its security is as strong as the existing solutions [19].

In section 6.4.3 we provide more details on the implementation we use. A formal mathematical formulation of these properties can be found in Appendix B

## 6.4 Architecture and Implementation

### 6.4.1 General Design

The private engine that creates the proofs inserts itself in the middle of the architecture of the solution. Using the beacon data for a random selection, we take each private tuple, and give those two to the client to make the required selection. At the end the verifier would have the same data it had before to verify the selection was made correctly and it would also be able to verify the validity of the proofs. But this time, they will not have access to the plain value of the priority.

This can be seen in Figure 6.1.

[2] `https://ethereum.org/`

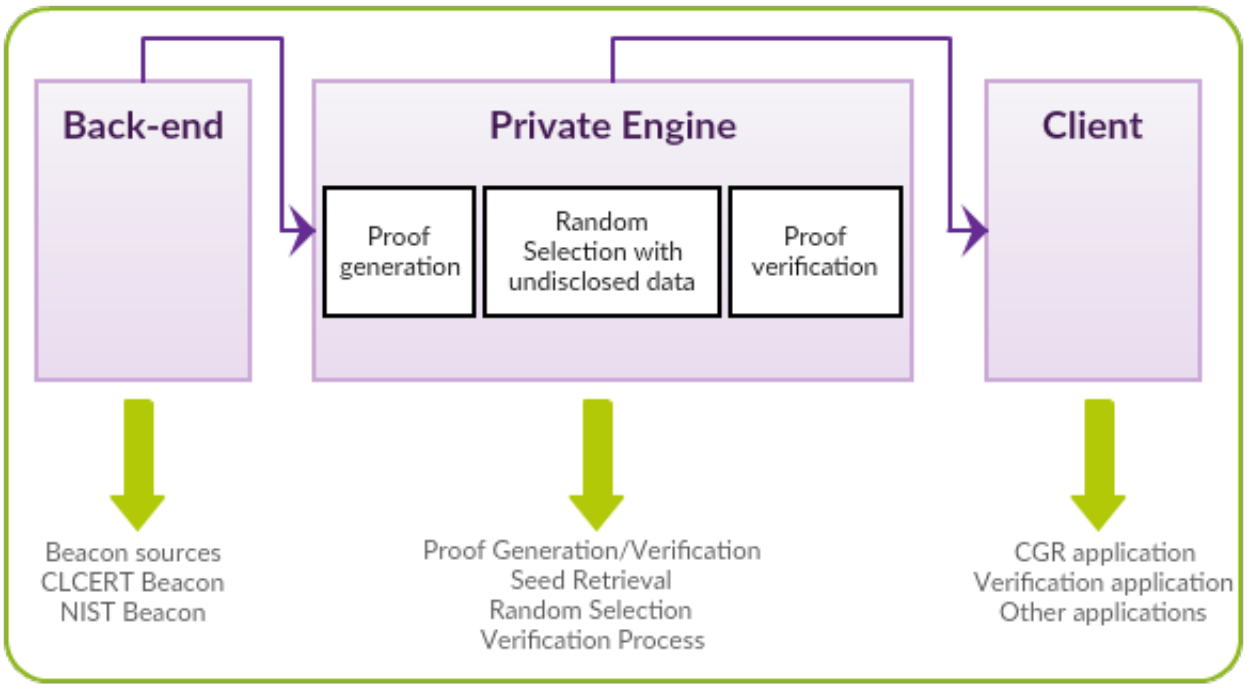


Figure 6.1: Modified architecture of the solution

One of the primary goals was to have a product that could and would be used in organisations, by putting together current unexplored theoretical research into an easy to integrate, usable bundle. Part of the same goal was to create this product to produce a lasting impact on transparency in public organisations, whilst using remarkable academic developments.

In the newly proposed architecture is composed by three parts:

- **Back-end**: it consists of each of the beacon sources that can be used to provide randomness. Each runs separately; this is important to provide trustworthiness. In case one of the beacons becomes compromised, there are still others that are commanded by separate entities.
- **Private engine**: it consists of the proof generation, the random selection using the encrypted values and the following proof verification. The proof generation is held by the trusted entity (organisation) that will perform the random selection. The random selection is done publicly using the previously encrypted values. The proof verification is performed anywhere where the verification script runs, and this is standalone so it could be verified by any member of the public.
- **Client**: it consists in each application that uses the engine to make a random selection. It is hosted by each service that will make the random selection. The application verification is hosted, ideally, by a trusted third party external from the organisation that made the random selection. This is not critical, since the script is standalone for verification.

### 6.4.2 A Building Block

As a part of the development and implementation of new cutting-edge technologies, a zero-knowledge library was created within the `go-ethereum` open source library[3]. The `go-ethereum` library is under the *GNU Lesser General Public License* and, under the license to modify it, it was adapted to allow for each of the scenarios presented in the next section. The range proof implemented in this library was originally designed and used to implement anonymous transactions in Ethereum. We adapt this range proof for our application in this work.

The library uses commitments and zero knowledge proofs. A commitment scheme was used to *hide* the private values, and the zero-knowledge range proof schemes modified from the *Ethereum* library were used to allow for these values to be placed for selection.

The algebra of commitments is thus far limited to the current implementation of the scheme, simple mathematical operations. The implementation uses the Fujisaki-Okamoto commitment scheme, which is an extension of the Pedersen commitment scheme to the RSA modulus. We can find in [19], all mathematical formulations for this scheme and for the range proof used.

### 6.4.3 Proof Generation

In this section, we explain how the range proof implementation of the go-ethereum library works, and how we use it in our design.

We proceed to take what we want to hide and use it as an input in the proof generator. The design is shown in Figure 6.2.

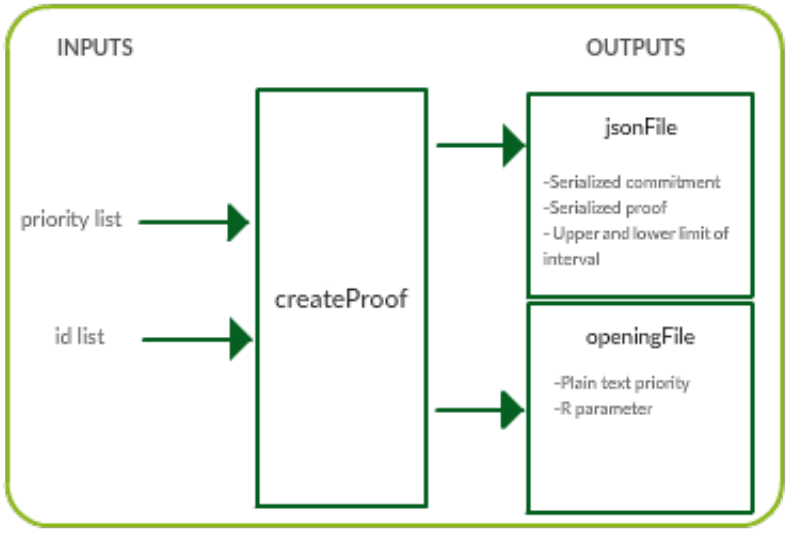


Figure 6.2: Inputs and outputs for `createProofs`

[3] `https://github.com/ethereum/go-ethereum`

To start we have the same initial list of priorities. This is what we want to hide from the public. The action of creating the commitments and the proofs **is done by the trusted organisation**.

We take the list of `ids` and the list of priorities. We create a `privateType`, the current implementation is limited to 3 different intervals. We create a closed range for every interval, an empty list of `privateTuples` and we call the function `createProofs` and give it the priority list.

What does `createProofs` do? For every priority we create a commitment, and a zero knowledge closed range proof that the commitment belongs to that particular range.

This is how `createProofs` generates the list:

1. Create an object `secretOrderGroup`.
2. Get the parameters N, G and H from the group. These are the parameters needed to create the proof.
3. Create a JSON object `jsonFile` that will contain the public result.
4. For every integer in the list generate a `TTPMessage` object. Creating a `TTPMessage` requires the priority we do not want to reveal, as a BigInteger, and the `secretOrderGroup`.
5. Then locate where the message is in the intervals and do two things:
   (a) Calculate the `range Proof`.
   (b) Create a `PrivateTuple` with the commitment, the proof, and the limits.
6. If the proof creation was successful, then add the `privateTuple` to the list that contains all the created `privateTuples`.
7. Get R from the `secretOrderGroup`.
8. Save R and the plain text priority to a JSON file `openingFile`. This is done in case the organisation needs to open the commitments.

The previous description requires several objects and parameters. They will be explained in the following section.

An example of the public JSON file `jsonFile` is shown in Figure 6.3.

```
{ ⊟
   "tuples":{ ⊟
      "0":{ ⊟
         "serializedProof":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLlJhbmdlUHJvb2acjb2ESK6
         "upper":"30",
         "lower":"0",
         "serializedComm":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLkNvbW1pdG1lbnQAWgkySbtN
      },
      "1":{ ⊟
         "serializedProof":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLlJhbmdlUHJvb2acjb2ESK6
         "upper":"30",
         "lower":"0",
         "serializedComm":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLkNvbW1pdG1lbnQAWgkySbtN
      },
      "2":{ ⊟
         "serializedProof":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLlJhbmdlUHJvb2acjb2ESK6
         "upper":"30",
         "lower":"0",
         "serializedComm":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLkNvbW1pdG1lbnQAWgkySbtN
      },
      "3":{ ⊟
         "serializedProof":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLlJhbmdlUHJvb2acjb2ESK6
         "upper":"30",
         "lower":"0",
         "serializedComm":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLkNvbW1pdG1lbnQAWgkySbtN
      },
      "4":{ ⊟
         "serializedProof":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLlJhbmdlUHJvb2acjb2ESK6
         "upper":"30",
         "lower":"0",
         "serializedComm":"rO0ABXNyACRjb20uaW5nLmJsb2NrY2hhaW4uemsuZHRvLkNvbW1pdG1lbnQAWgkySbtN
      }
   }
}
```


Figure 6.3: Example of public JSON of proofs

There are 6 objects and parameters used within the implementation that we further clarify now.

### 1) secretOrderGroup

The generator, an *object*, for the `secretOrderGroup` receives a certain bit length. For the generation we create the safe primes and we find the generators.

That is how we create a `secretOrderGroup` that contains:

- A BigInteger `N`, a *parameter*, is large composite that defines a group $\mathbb{Z}_{\mathbb{N}}^*$
- A BigInteger `g`, a *parameter* and a generator for G is a cyclic subgroup of $\mathbb{Z}_{\mathbb{N}}^*$ with a large order. The order of this subgroup is secret.
- A BigInteger `h`, an element of the group generated by `g`. This is a generator for a subgroup of `G` with large order, as needed for KDLCGSO.

### 2) Commitment

The `commitment`, an *object*, consists of the `secretOrderGroup` that was used to create it and the commitmentValue.

### 3) TTPMessage

The `TTPMessage`, an *object*, is the message from the trusted third party to the prover. This message contains:

- The commitment: `c`, the public part that can be published. This should be signed by the trusted third party.
- `X`: the secret number to hide in the proof (e.g. the priority value).
- `Y`: the secret key.

### 4) RangeProof

The range proof, an *object*, represents the generated proof that the hidden value is within the `closedRange` that it is said to be in.

### 5) PrivateTuple

The `privateTuple`, an *object*, is what will be made available publicly. The tuple contains:

- The commitment, `c`
- The Range proof
- The upper limit, a *parameter*
- The lower limit, a *parameter*

### 6) PrivateType

The `privateType`, an *object*, consists of the created type to manage and create each of the proofs so it communicates with the entire library. It holds the main methods that generate the ranges, the commitments and the list of proofs.

This is also the place where we create a file that saves the values of R and the plain text `ID` (value to be hidden at the start). As a result, the trusted organisation that creates the commitments can open them if the need ever arises.

### 6.4.4 Verification

For the public verification we have the public JSON file that was created before (example in Fig. 6.3), That in itself has all the data necessary to be verified.

We create a `PublicVerification` object with the JSON file. In it we take each tuple and create yet again a list of `PrivateTuples` with each commitment, proof and the interval limits.

We can validate a single tuple or the entire list of proofs in a batch. The validation occurs in the `HPAKERangeProof` class. As seen in the paper [19] we only need the publicly available data to validate correctness. These are the parameters `N`, `g`, `h` and the commitment value `c`.

We deliver a simple true or false value in the case that the proof is valid and the commitment does belong to the interval that it is said to belong.

## 6.5 Limitations

There are several limitations to the developed implementation. They are listed and explained next.

We are only providing transparency to the selection process, there is currently no way of assuring that the data was not manipulated prior to using it as input.

The system works if the prover inputs the correct values, it does not say where it falls and which part of the process was not correct. Currently it only provides a boolean value that represents correctness.

For simplicity the parameters used for the verification process (generators) are created by the *Ethereum* code before the proof. There is ongoing research to generate these parameters from the random beacon and this would eliminate the dependency from their generated values.

## 6.6 Open Problems

Given the limitations of the current tools available, it was not possible to reach all the scope that was intended at first. The applications that stayed out of scope that are part of future work are the following:

1. Provide visibility and transparency to the priority formula by calculating it using commitment algebra.
2. Hide the interval range by further modifying the *ethereum* library.

The library used is very interesting, albeit its original purpose was that of cryptocurrency and blockchain use. The possibilities are unexplored, which is important because it is an attainable, usable tool that can increment security and improve transparency.

One of the goals was to find a real life use case for this implementation. Therefore it is interesting to continue to find different use cases in which our model could be applied or extended to.

# Chapter 7

# Conclusion

This work had several parallel goals. First, we sought to study how verifiable randomness can be used to provide transparency in public organisations. Second, we looked to design and implement a particular case in a real life scenario. And finally, we wanted to further our research to see how incorporating private data into the mix would turn out.

In this work, motivated by the creation and development of the public randomness beacon hosted by the University of Chile, we studied the conceptual verifiable randomness and its use to provide transparency in public processes. We also examined options of possible parameters and identified and analysed which were relevant. We studied the limitations that arose from the selections and what each meant for a possible implementation. Once the design was underway, there was a constant re-examination of the correct application.

Second, to design and implement a practical use case we searched for a suitable organisation. We found the Comptroller General of Chile (CGR) and we proposed a new way to integrate something they strive for, providing transparency, with our implementation. Working with CGR meant meeting their requirements and rising to the high standard that such an institution demands. In this work, we developed a general engine to use verifiable randomness. This work is the first approach and can be replicated throughout different agencies and organisations.

Third, we considered the case of providing transparency for public processes when private data was involved. We first looked at what it meant to include private data, its implications and consequences and identify publicly available tools which could be adopted to add the new functionality into our existent implementation. For the design and tools, we evaluated: possible ranges, possible succinct proofs, distribution of the samples, type of commitments and how broadly each case would be defined. We found the work [19] of Peng and Bao in efficient range proofs developed in the ethereum environment, an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. We adapted and further implemented it as an engine component in our architecture design.

One of the most important requirements of this work was to have a real practical use case, are where the impact could be felt, and where the philosophy of the creation of all these tools

came to fruition. The client at CGR would have an impact on real people and, hopefully, on more organisations that would follow through. While providing transparency was the goal of this research, understanding the parameters and limitations was a useful byproduct of our work.

As for the extension to private data, there is still much to be done. However, there is a working prototype, ready to be assessed and implemented in the future. But also we hope our work motivates further research on transparency strategies.

The importance of delivering transparency only becomes more prevalent with this study and with our application in use at the government. We hope other institutions become interested in pursuing the same path. Making the academic research readily available so new tools are developed was also an important goal.

As a conclusion, even though the amount of transparency delivered by our application was not easily measured, we finally deemed the hypothesis correct. The availability of verifying tools for the public is one path towards a more open and responsible society. One where individuals can hold the organisations that surround them accountable and those organisations welcome such accountability.

## 7.1 Future Work

The work presented in this thesis can be extended in different paths.

Architecture: It is possible to improve trustworthiness in the selection by increasing the number of beacons used at the same time. It may also be interesting to study how to combine the beacons. For example, evaluating strategies so we could use only two of five beacons available, or only using those beacons that have the best rate of response, or those geographically restricted.

Engine: The scenarios in the engine can be expanded to consider a wider set of requirements. For example, we could consider other *business logic* such as selections from a set with non-uniform distributions or under different constraints. As the system is implemented in more organisations, it will become more robust and likely acquire more capabilities.

New Engine: Recent implementations of results [18] [3] on cryptographic zero knowledge proofs could be added into the cryptographic library to expand the private scenarios allowed at the moment. As an example, we might want to hide the range boundaries in the range proofs. Another possible abstraction to the engine concept would be to develop a compiler that given a set of parameters, would be able to automatically generate a dedicated engine to suit the needs of a particular implementation.

# Bibliography


[1] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM journal on Computing*, 13(4):850–864, 1984.

[2] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[3] R. Chaabouni, H. Lipmaa, and B. Zhang. A non-interactive range proof with constant communication. In *International Conference on Financial Cryptography and Data Security*, pages 179–199. Springer, 2012.

[4] I. Damgård. Commitment schemes and zero-knowledge protocols. In *School organized by the European Educational Forum*, pages 63–86. Springer, 1998.

[5] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 125–142. Springer, 2002.

[6] J. Fairbanks, K. D. Plowman, and B. L. Rawlins. Transparency in government communication. *Journal of Public Affairs*, 7(1):23–37, 2007.

[7] N. Ferguson and B. Schneier. *Practical cryptography*, volume 23. Wiley New York, 2003.

[8] A. M. Florini. Increasing transparency in government. *International Journal on World Peace*, pages 3–37, 2002.

[9] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[10] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom generators. *SIAM Journal on Computing*, 22(6):1163–1175, 1993.

[11] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[12] S. G. Grimmelikhuijsen. Transparency of public decision-making: Towards trust in local government? *Policy & Internet*, 2(1):5–35, 2010.

[13] J. Kelsey. The new randomness beacon format standard: An exercise in limiting the power of a trusted third party. In C. Cremers and A. Lehmann, editors, *Security Standardisation Research*, pages 164–184, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04762-7.

[14] J. A. Kroll, S. Barocas, E. W. Felten, J. R. Reidenberg, D. G. Robinson, and H. Yu. Accountable algorithms. *U. Pa. L. Rev.*, 165:633, 2016.

[15] A. Meijer. Understanding modern transparency. *International Review of Administrative Sciences*, 75(2):255–269, 2009.

[16] R. Oliver. *What is Transparency?* What Is the What Is . . . Series. Mcgraw-hill, 2004. ISBN 9780071435482. URL `https://books.google.cl/books?id=sqZssG77k5AC`.

[17] L. A. Pacek. Transparency in government. Technical report, Retrieved 15/5/2018 from https://scholar. google. com/scholar, 2011.

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

[19] K. Peng and F. Bao. An efficient range proof scheme. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 826–833. IEEE, 2010.

[20] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Annual International Cryptology Conference*, pages 433–444. Springer, 1991.

[21] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.

[22] J. Stiglitz. Transparency in government. *The right to tell*, page 27, 2002.

[23] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford. Scalable bias-resistant distributed randomness. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 444–460. IEEE, 2017.

# Appendix A

## A Zero-Knowledge Protocols: Mathematical Formulations

Let $L$ be an NP-language defined by a NP relation $R$, that is

$$L_R = \{x \in \{0,1\}^* \mid \exists w \in \{0,1\}^* \text{ such that } R(x,w) = 1\}$$

For any $x \in L_R$ the value $w$ is called a *witness* for the membership of $x$ in $L_R$. Relation $R$ can be computed in polynomial time and therefore the length of $w$ is polynomially related to the length of $x$.

Let $P$,$V$ be interactive PPT algorithms [1] , and $x, w$ values in $\{0,1\}^*$. The pair $(P,V)$ is called an interactive protocol. We denote $\langle P(x,w), V(x)\rangle$ as the output of algorithm $V$ after $V$ runs on input $x$ while interacting with $P$ on input $(x,w)$. Value $x$ is called *the common input*. In what follows, when we refer to negligible functions we mean they are negligible as a function of the length of the common input $|x|$.

**Definición A.1** The protocol $(P,V)$ is called an *interactive proof* for language $L$ if it satisfies the following properties:

**Correctness:** If $x \in L$ then $Pr\left[\langle P(x,w), V(x)\rangle = 1\right] \geq 1 - \varepsilon(|x|)$ where $\varepsilon(n)$ is a function negligible on $n \in \mathbb{N}$.

**Soundness:**[2] If $x \notin L$ then for all malicious prover $P^*$, $Pr\left[\langle P^*(x), V(x)\rangle = 1\right] \leq \varepsilon(|x|)$ where $\varepsilon(n)$ is a function negligible on $n \in \mathbb{N}$.

[1] In general, one can define interactive proofs for even computationally unbounded provers. Our choice of PPT here is because we want to capture protocols that can be *implemented* in computers.

# Appendix B

## A The Range Proof Protocol[1]

This section follows the explanation from the work by Peng and Bao in [19]. The main idea of the range proof scheme is as follows.

- We employ the secret order principle and the CBPKCGSO technique (computational bindingness through proof of knowledge in cyclic groups with secret order) to make the secret integer computationally binded and proof of non-negativity of an integer easier.
- An integer $m$ is in a range $a, a+1, ..., b$ if and only is $(m-a+1)(b-m+1)$ is positive.
- If $w^2(m-a+1)(b-m+1)$ is positive, $(m-a+1)(b-m+1)$ is positive.
- To prove that $w^2(m-a+1)(b-m+1)$ is positive, it is divided into three shares $m_1, m_2, m_3$, whose sum is $w^2(m-a+1)(b-m+1)$. Moreover, $m_3$ is a square and thus non-negative. If both $sm_1 + m_2 + m_3$ and $m_1 + tm_2 + m_3$ are positive where $s$ and $t$ are random positive integers, then $w^2(m-a+1)(b-m+1)$ is positive with an overwhelmingly large probability.
- As the information revealed from $sm_1 + m_2 + m_3$ and $m_1 + tm_2 + m_3$ about $m$ is statistically negligible except showing that $m$ is in the range $a, a+1, ..., b$, they can be published without compromising statistical privacy.
- 
  - We show achievement of soundness in the new range proof according to Theorem 1, proven by Damgård and Fujisaki in [5]
  - We intuitively argue that the revealed information in our range proof is so trivial that the proofs transcript can be simulated by a party without any knowledge of any secret such that the simulating transcript generated by the party is statistically indistinguishable from the real proof transcript.

Proof that a commited number is a square is denoted as $SQR(\chi, r|g, h|y)$, which proves knowledge of integers $\chi$ and $r$ such that $y = g^{\chi^2} h^r mod N$.

The fukisaki-Okamoto commitment function is employed. We employ two large security parameter $k_1$ and $k_2$. $k_1$ is much smaller that $k_2$. However, $k_1$ is large enough that $1/(k_1-1)$is a negligible probability. A recomendation for their values is that $k_1$ is large but much smaller

[1] An efficient range proof scheme, by Peng and Bao [19]

than the order of $G$ (e.g. 160 bits long) and $k_2$ is larger than the order of $G$ (e.g. longer than 1024 bits).

A secret integer $m$ is comited to in $c = g^m h^r mod N$ where $r$ is a random integer in $Z_{k_2}$. $m$ is in interval range $a, a+1, ..., b$. A party with knowledge of $m$ and $r$ has to prove that the message committed in $c$ is in $a, a+1, ..., b$. The proof protocol and the corresponding verification are as follows.

1. The prover calculates and publishes $c_1 = c/g^{a-1} \bmod N$and $c_2 = g^{b+1}/c \bmod N$.
2. He calculates and publishes $c' = c_1^{b-m+1} h^{r'} \bmod N$ and publicly gives a proof
$$EL(b-m+1, -r, r' | g, h, c_1, h | c_2, c').$$
Where $r'$ is a random integer in $Z_{k_2}$.
3. He randomly chooses integers $w$ and $r''$ respectively from $Z_{k_2} - 0$ and $Z_{k_2}$ and publishes
$$c'' = c'^{w^2} h^{r''} mod N.$$
He publicly gives a proof
$$SQR(w, r'' | c', h | c'').$$
4. He randomly chooses three non-negative integers $m_1$, $m_2$ and $m_4$ smaller than $w^2(m-a+1)(b-m+1)$ such that
$$m_1 + m_1 + m_3 = w^2(m-a+1)(b-m+1)$$
$$m_3 = m_4^2$$
He randomly chooses $r_1, r_2, r_3$ to satisfy $r_1 + r_2 + r_3 = w^2((b-m+1)r + r') + r''$ and publishes
$$c_1' = g^{m_1} h^{r_1} \bmod N$$
$$c_2' = g^{m_2} h^{r_2} \bmod N$$
$$c_3' = g^{m_3} h^{r_3} \bmod N.$$
He publicly gives proof
$$SQR(m_4, r_3 | g, h | c_3')$$
5. A verifier randomly chooses and publishes integers $s$ and $t$ in $Z_{k_1} - 0$.
6. The prover randomly publishes
$$x = sm_1 + m_2 + m_3$$
$$y = m_1 + tm_2 + m_3$$
$$u = sr_1 + r_2 + r_3$$
$$v = r_1 + tr_2 + r_3$$
7. Besides verification of the proofs, the following equations have to be publicly verifies
$$c_1 = c/g^{a-1} \bmod N$$
$$c_2 = g^{b+1}/c \bmod N$$
$$c'' = c_1' c_2' c_3' \bmod N$$
$$c_1'^{s} c_2' c_3' = g^x h h_u \bmod N$$
$$c_1' c_2'^{t} c_3' = g^y h^v \bmod N$$
$$x > 0$$
$$y > 0$$

Any one can publicly perform the verification. Once all the integers involved in a verification equation are available, the equation is verified. If a verification fails, the proofs protocol fails and stops. The proof protocol succeeds if and only if all the verification conditions are satisfied.