CLCERT Random Beacon
Progress Report - September 2017

Alejandro Hevia, Camilo Gómez, Cristián Rojas
{ahevia,cgomez,crirojas}@clcert.cl

Dept. de Ciencias de la Computación (DCC) &
Applied Cryptography and Security Laboratory (CLCERT)
Facultad de Ciencias Físicas y Matemáticas (FCFM)
Universidad de Chile

Abstract

In this report, we briefly describe the current state of the CLCERT Random Beacon project. We detail the project's architecture, modules, and some key design choices, including those steaming from security considerations. The report also discusses some of the work coming ahead for the next period.

The CLCERT Random Beacon was designed to provide a public randomness service which is unpredictable, autonomous, consistent, and externally verifiable. The Beacon seeks to complement similar efforts worldwide by providing a reliable and independent source of randomness.

1 Introduction

The CLCERT Random Beacon was designed to provide a public randomness service which is unpredictable, autonomous, consistent, and externally verifiable.

In this report, we briefly describe the status of this project, including goals and milestones achieved and those which are still in progress. As we detail later in this report, during the first year, some of the key objectives included the overall design of the system, its architecture, and implementation. They are described them in Section 3. In the following section, we discuss the security considerations that determined some of the operational features of the system. We then present some current results on the development of applications of the CLCERT Random Beacon, which were achieved ahead of schedule given the unexpected change of schedule (see next section). Section 6 provides an outline of some of the goals and milestones scheduled for the second year. Finally, we conclude with a detailed comparison of our project's actual accomplishments with the goals and accomplishments established for the period, and some general conclusions.

2 Funds Availability and Schedule Update

Our project was affected by an unexpected delay in the funds availability. Funds were made available to our university only by late August 2017. This had two major effects:

- the generation of several unliquidated obligations, mainly the project member's salaries, and
- the delay in the purchase of the project equipment (hardware).

The first one was not a major obstacle as many university projects start with severe delays, so the situation was not unexpected (although the length of the delay did cause some financial constraints for all team members). The second effect, on the other hand, caused a more significant problem, as the implementation

could not be tested on the actual hardware which would be finally used. The implementation had to be tested on simulated hardware. Nonetheless, the development team took the opportunity to make inroads into goals and milestones scheduled for the second year. The results (goals and milestones) for this period are detailed in Section 7.

In next section, we describe the proposed system architecture for the beacon.

3 System Architecture

Figure 1 describes the architecture of the proposed beacon.

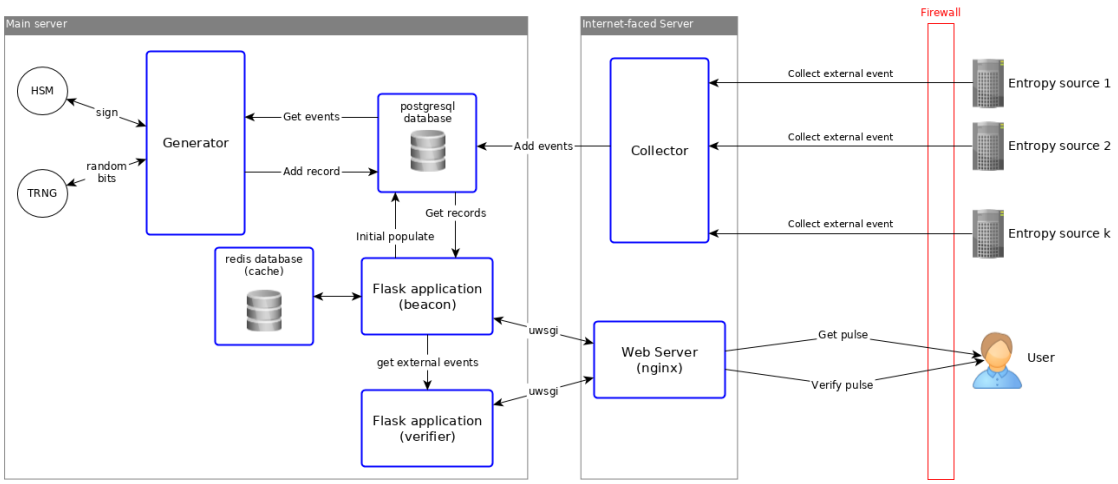

Figure 1: Architecture Diagram

The CLCERT Random Beacon system is composed of seven (7) modules organized into two physical servers. The motivation for this modular design was to follow the *separation of duties*, a standard security practice. Each module is highly specialized, playing an specific role in the system. Concretely, our design seeks that all modules can be decoupled, this is, implemented separately from each other, in such a way that all modules could be replaced or updated without affecting the rest of the system. Currently these modules are developed as Docker containers[1] in order to support a micro-services architecture. This approach allows us to not only deploy the system much faster than using other virtual machine environments, but also update each of the modules without disrupting the whole system. Moreover, the decision of using a container-based design was a direct response to some of the security considerations described later in this document.

In the current section we provide a high level description of each module, how they operate, and how they interact with the rest of the modules of the system. We also discuss some implementation details associated to these modules.

3.1 The Verifiable Beacon: A High-level description

At a high level, our beacon follows a similar approach as the NIST beacon: every 60 seconds, the systems outputs a *pulse*, a 512-bit long string called the "random output". One mayor difference is the beacon's use of sampleable sources. Each minute, the CLCERT *Beacon Collector* module samples several publicly accessible sources in order to extract data which is expected to contain randomized values. Then each sample is processed, stored, and its randomness added into the pool. Data processing (filtering) is explained in Sect. 3.2, and the design and implementation of the data storing process is explained in Sect. 3.4. Adding data to the pool is done by the *Beacon Random Generator* module (Section 3.3). The whole process is designed to be verifiable by any external third party, as long as the party is connected to the mentioned sources.

[1] https://www.docker.com/

The design and security considerations are detailed in a internal white paper (Gómez et al. 2017).

3.2 Beacon Collector

This section provides an outline of the *Beacon Collector*.

As mentioned in the previous section, at the beginning of each minute, the collector module visits a collection of publicly accessible active data sources and extracts (copies) a sample. This data, which we sometimes refer as a data source "event" is then added them to a database. Also the collector performs a conservative estimation of the amount of entropy that each source is producing with each new event. The objective of the collector module is to maximize the total amount of entropy collected on every minute: the system both extracts events from many different sources with small entropy, and well as accessing events from a few sources where each of them is likely to provide a large amount of entropy.

Current Sources

The system is currently extracting events from the following sources:

- **Earthquake Web.** At the 30 second mark each minute, the collector requests data from the following site: http://sismologia.cl/links/por_dia/index.html. The site delivers information about the last "large" earthquakes (those magnitude 2.5 or above) which have happened in the Chilean territory. The data is provided by the "*Centro de Sismología Nacional de la Universidad de Chile*".
- **Earthquake Twitter.** At the 30 second mark each minute, the collector requests the last 30 tweets produced by the twitter account 'Sismos CSN' (https://twitter.com/Sismos_CSN). Note: This is a temporary source *(it will be phase out in the future, as there it is redundant with the previous source).* mainly used for testing correctness of new functionality.
- **Trending Twitter.** At the beginning of each minute, the collector obtains the current top 10 trending topics from Twitter among those labelled as belonging to Chile by Twitter. After that performs a search using those trending topics as keywords and obtains the tweets produced by users in Chile using those keywords between the 15 and 30 seconds mark of the current minute.
- **Radio Stream.** The collector also obtains the raw audio produced by the public signal of 'Radio Universidad de Chile' (http://radio.uchile.cl/) between approximately the 15 and 30 seconds mark within the current minute. The data currently consists of 20 blocks of 4096 bytes each.

There are new sources that we expect to add in the short term. See Section 6.2.

Properties of a good entropy sources

The sources mentioned in the previous section have been selected based on the following criteria.

- **'Fixed Star'**: If a person consults the source from a location (say, Europe) which is geographically distant from Chile, the source must provide exactly the same information as that seen by (say) a person that consults the source at the same time from Asia. In theory, we want this property hold independently from where the query was made.[2] This property is required to guarantee universal verifiability of the beacon.
- **Trustworthy**: Each sources must be controlled by a trustworthy organizations, one that in practice has little (negligible) motivation to provide malicious information. By "malicious" we mean any data that do not follow the natural distribution of the source, in particular, any value produced with the goal of altering or manipulating the result of the random number generator. The correctness of our beacon, however, will not rely on this property.

[2]The name comes from the fact that 'fixed stars' do not appear to move relative to each other no matter from where in earth they are measured.

- **Potentially high-entropy source**: The randomized process behind each source must be such that it is likely to produce a significant amount of entropy. Examples are data from twitter, blockchain, and newspapers. This property is important in order to produce random outputs with some minimum levels of entropy.
- **Availability**: Each source must be (at least in principle) available to be queried when needed. Ideally, the source's uptime most be high, being able to query it at frequent intervals (say, each minute).
- **Dynamic Information**: The information provided by the source must change with the highest frequency possible. Combined with 'availability', this property seeks to maximize the chances that new entropy is collected from the source.

The following section describes some concrete aspects of the beacon implementation.

Implementation

The Beacon Collector is an application developed on Python 3. To maximize parallelism, for each active source of entropy, the collector launches processes in parallel that retrieve the raw data provided by each source.

3.3 Beacon Random Generator

At the 30 second mark of each minute (after the collector module has completed), the generator module retrieves all the external events collected during the current minute. It also obtains a local random value generated by a TRNG hardware in the previous minute. After collecting those values, the generator module computes the hashed digest of all of them, signs it, and then produces the random output using the algorithm explained in the following section. The random output value is then added to the database.

Values used for Random Output Generation

To produce a preliminary output value Y, the random generator module concatenates the following data:

- Hashed version of the application.
- Frequency of random outputs generation.
- Hashed public key (certificate) of the application.
- Timestamp of current random output that is being generated (beginning of next minute).
- Local random value generated by TRNG in the previous minute.
- Hashed values of all external events collected in the current minute.
- Previous random outputs generated (previous minute, previous day, previous month and previous year).
- Pre-commitment of a new local random value generated by a TRNG hardware (the value itself will be used in the following minute).
- Status code of the current random output being generated (start of a chain, start of a skip-list or normal).

Formal specification of the Random Output Generation

The output value `Output` is generated from Y as follows $S = \texttt{Sign}_{\texttt{sk}}(Y)$, $C = Y||S$ and $\texttt{Output} = \texttt{SHA3}_{512}(C)$. where sk is the signature generation key stored in a HSM hardware module, and `SHA3` is the standard SHA3 cryptographic hash function.

3.4 Beacon Databases

The CLCERT Beacon utilizes two databases. The main database, implemented using PostgreSQL, is accessed using SQLAlchemy[3]. The second database is a cache, and it is implemented using Redis[4]. Data stored in both databases includes the following:

- **External Events**: data from all external events gathered by the collector module are stored in the main database and copied to the cache database. In particular, all raw data is stored only for one hour in the cache database. All hashed values and associated metadata (timestamps, source ids, entropy estimation and correlated entries) are stored in the main database. The main database is periodically backed up offline.
- **Entropy Sources**: A description and any specific configuration data associated to each external sources is also stored in the main database. Sources can be updated and added throughout the lifecycle of the project.
- **Random Outputs (Records)**: Each random output computed by the generator module is stored alongside their respective signatures.
- **Local Random Values**: Each random output generated by the local RNG hardware is stored in the main database, including each pre-commitment performed for each of them.

3.5 CLCERT Beacon Web Interface (Viewer)

The system includes a web interface to allow easy access to all random outputs produced by the generator module. Each output is associated to a *record*, a unit of information associated to this output. The most recent record is called the *current record*.

Record Information

Any record shown in the main page (viewer) displays the following information:

1. `Version`: current application's version.
2. `Frequency`: frequency of new random values generation.
3. `Time`: timestamp of current record being showed.
4. `Previous`: random output value of the previous minute.
5. `Signature`: signature of the current random output value.
6. `Output Value`: current random output value.
7. `Status`: a flag that indicates whether the output was correctly generated or if it was some any problem during collection or generation.

Current API

The CLCERT beacon also provides an API (Application Programming Interface). Currently implemented calls are listed below. To access them, users submitting a request need to specify the record generation time in POSIX format[5].

Current Record (or next closest)

```
/beacon/1.0/pulse/<timestamp>
```

Previous Record

[3] http://www.sqlalchemy.org/

[4] https://redis.io/

[5] Number of seconds since midnight UTC, January 1, 1970 (see http://en.wikipedia.org/wiki/Unix_time for more information and http://www.epochconverter.com for an online timestamp converter.)

`/beacon/1.0/pulse/previous/<timestamp>`

Next Record

`/beacon/1.0/pulse/next/<timestamp>`

Get Record by Id

`/beacon/1.0/pulse/id/<id>`

If no record is found for a given request, a 404 response is returned.

Last Record Generated

`/beacon/1.0/pulse/last`

Start of Chain Record

`/beacon/1.0/pulse/start-chain/<timestamp>`

Raw External Events Values for Record by Id

`/beacon/1.0/raw/id/<id>`

Tentatively, queries to the database are only able to obtain raw external events for records generated in the last 60 minutes. This design decision is motivated by security concerns, namely preventing *denial of service attacks* (DoS). Whether this value can be extended to longer periods will be evaluated in the system testing phase, after deployment.

3.6 Beacon Record's Web Verifier

One key functionality of the CLCERT Random beacon was that each output value generated by the beacon should be externally verifiable. This means that any user who was not involved during the computation (generation) of a random output for a given timeframe, can obtain guarantees that the process indeed followed the specified procedure and no external source was able to influence the outcome (the random output) more than any other external source. In general, the recommended strategy to perform this verification procedure is to use an open source application that, recomputes the output vale starting from the public data from the external sources and the committed local random value. The process must verify if a given output value for a given timeframe can be recomputed using only the public data extracted from each of the active sources as well as the local random value committed in the previous timeframe.

To simplify the user experience, the system will also include a simple web interface (currently in development) to perform the same verification. The system will also provide a mechanism to verify the correctness of any raw data associated to each of the random outputs generated in the last hour. The mechanism will allow any external user to recover all raw data extracted by each one of the sources (for a fixed window of time) so the user can compare it with her own sampling of the same source.

4 Security considerations

The system architecture of the proposed beacon considers two servers: an Internet-facing server which contains the collector and public web server, and a main server which contains the rest of the beacon's components. The system restricts access to the main server, which cannot be accessed directly from the Internet, and communication is only allowed between with the Internet-facing server. This design was the outcome of several security considerations, among them, the standard *CIA Triad* security model (Perrin 2008).

Confidentiality

The system was designed to guarantee that the *High-Security Module* (HSM) data cannot be accessed externally. Also, the main server cannot be accessed externally, as it contains sensitive data. All access to this data is mediated by the Internet-facing server. As an additional security measure, the system does not store user data.

Integrity

The system was designed to guarantee that the data stored is not tampered or destroyed. This will be guaranteed via network segregation and mediated access, as well as frequent database backups.

Also, data sources must be treated as tainted so they will be sanitized before storage to prevent both potential SQL injection attacks, as well as data verification abuses (ie. potential Cross-Site Scripting attacks).

Availability

The system must contain recovery mechanisms that impede the downtime of both the web server and the collector. This will be considered using orchestration tools like Kubernetes, and server anti-DDoS protections.

Other Concerns

To complement the project's security program, the following tasks are also being performed:

1. Application Threat Modeling, based on Microsoft's Security Development Lifecycle (Shostack 2014).
2. Semiautomatic internal and external software vulnerability analysis (McGraw 2006).
3. Server stress tests to address potential (D)DoS vulnerabilities.

5 Developed Applications

5.1 Public Auditing Application

We are currently working with the Contraloría General de la República o CGR (the Chilean equivalent to the Government Accountability Office in the US) to provide more transparency to some of the audit processes they must carry out. In particular, there is a list of high-profile roles and institutions that CGR must audit by law. Moreover, each member on this list has a priority factor (calculated according to risk parameters defined by law). Unfortunately, there is not enough staff to audit every single person in the list, not even those with top priority, so the CGR must resort to select who to audit by randomizedly sampling the list. But proving that this randomized process is fair and unbiased has proven to be hard. It is often the case that those that are chosen to be audited claim that the process was biased, and the selection was not really random but carefully done in order for them to be selected.

In this context, our team is working with the CGR, to generate a system that will take sequences from more than one random beacon and combine them to create a seed for a pseudo random generated number to make a selection for any selection process assigned to CGR. The goal is to provide a system that outputs the list of the people selected for audit along with a "correctness certificate". This certificate, awarded by the University and posted in a public manner, guarantees that the randomness was fairly computed and in a no way influenced by the CGR. Verifying this certificate is simple and can be done either on the CGR web page or by running an open-source standalone application which has access to all public data generated by CGR

and CLCERT. This process is already completed although the access to the random beacon is still simulated (as the service is only running in a testing environment).

From the point of view of the CGR, adapting their systems to utilize a public randomness service (either the standalone one provided by CLCERT or the distributed one assembled from several beacon providers worldwide) allow them to improve the trustworthiness and transparency of their processes, taking away any suspicion of political bias. CGR also has used this opportunity to improve the overall effectiveness and reliability of their processes.

In the future, we expect to expand the system to cover new types of operations beyond the auditing process mentioned above. Processes including private data, for example, where although the data is not meant to be public, it may be critical in determining the output of the selection (say if it used in the computation of the priority factor). Our goal is to provide a system that guarantees that the outcome is fairly and robustly computed following the specification.

6 Future Work

In this section, we describe our plans for the second year of the project.

6.1 Source Entropy Estimation

Our work so far has show the importance of entropy estimation, namely how much entropy is being recollected from each of the active sources. Without it, we can hardly validate our claims that the distribution induced by the process that produces the random output (generated by our beacon) is such that there is approximately 512 bits of entropy on each sample. We are working on a mechanism to estimate the amount of entropy associated each source using some statistical processes, which should help us to evaluate if the source is reliable or not. The same system should also allow us to flag each random output with an estimation of how close to uniform outputs should be. This may allow users to decide if the random output is useful or not.

6.2 New Sources

Currently the team is analyzing whether more sources can be incorporated to the system, specifically the following:

- Public Blockchain information (those associated to Bitcoin and/or Ethereum)
- Financial data
- Astronomical data (from several sources in Chile).
- Public TV channel signals
- Weather data

6.3 Deployment in Real Scenario

We expect to receive the final equipment needed for the deployment of the beacon (servers, RNG hardware, HSM board, etc.) in the short term. Then, we will do our first deployment of the server side of the project. The first version will focused on improving the design of the system, namely, providing a cryptographically sound generation of the random output, starting from all data gathered from active sources. This version will also test our network security assumptions, hopefully improving in terms of the practical robustness of the system. User interface will not be a priority in this upcoming version but it will be considered in the subsequent version.

6.4 Applications

The Beacon will develop and deploy the following applications, implemented in the upcoming period:

- **Education**: The Chilean Primary Education System must produce a fair and random assignment of the new students in each of the schools in Chile. We are exploring options to use the random number generated by this beacon in order to produce the randomness needed on this assignment.
- **Science Simulations**: Science simulations often work by first sampling certain distributions (eg. Poisson, Normal, etc). We are investigating how to create a simple mechanism to produce such distributions using the randomness generated by the beacon, thus allowing sound but easy replication of randomized experiments published in scientific papers.
- **Miscellaneous**: What if we need to pick a random number between 1 and 100? Can I randomly select 10 different people out of 50? For these and other similar use cases, we are designing an application which can provide easy solutions to these problems (and similar ones) in a verifiable way using the beacon.

7 Current Period - Achieved Goals

In this section, we provide an analysis of the progress of the project in terms of objectives and milestones scheduled for the current period (first year) as presented in the project proposal.

7.1 Network design and deployment

The design of the architecture was completed as explained in "System Architecture" section. This includes defining the server and network architecture in order to configure a secure and highly reliable system. On the other hand, the deployment of the system is incomplete at this time given the delay on the equipment purchase. As it was stated before, our funding just arrived late in August 2017, so we are still clearing some formal requirements in order to buy the equipment (servers and dedicated hardware). Once the equipment arrives (estimated by December 2017) we are certain we can deploy in a very short period thanks to the modular design of the system (see Sect. 3).

7.2 Architecture design, implementation, deployment and fine-tuning

The design as well as implementation of the software are completed at this point (first version), as described in previous sections. Each of the different modules of architecture, including all the necessary cryptographic steps in order to generate a secure random output are completed. Given the delay mentioned before, the deployment, however, is still delayed until the equipment arrives.

7.3 Design, specification and implementation of relevant data formats

The system considers several data formats which need to be designed, specified, and implemented (their parsers). This process has been completed, as it was explained in Sections 3.4 and 3.5.

7.4 Source-verifiable randomness generation study, implementation and deployment

The entire study, design and implementation of all the software architecture dedicated to the source-verifiable randomness generation is already done (as explained in Sections 3.2 and 3.3). We still need to wait, however, for some dedicated software (HSM module and TRNG hardware) to arrive to include their functionality into our system. We are confident that the development of the corresponding modules can be done in a few weeks after the arrival of the equipment.

7.5 Design, implementation, deployment and evaluation of the cryptographic tools and systems to achieve public verifiability

The design of the public verifiability tools is also completed. Unfortunately, we are somewhat behind on the implementation and deployment of those tools, simply because some aspects of the graphic design of the system are still being tested. Usability is a critical concern in the project: if we do not provide a simple user interface to verify the randomness generated, the possibilities of replicating the auditing process to more applications drops drastically.

7.6 Testing and quality assurance of the random beacon system

We are constantly testing the quality of our system, mainly in three areas: security (as described in Sect. 4), integrity (cryptographically correct generation of randomness using external sources of entropy), and availability (in case of a high demand for requesting our services). We still working on the graphic design of the entire system, given the importance of usability on the long-term usage of our system.

7.7 Publication and discussion of the results in academic conferences and public venues

Even though we are working on a technical reports that describe and analyze the design and characteristics of our system (Gómez et al. 2017), we have not shared our ideas on any academic conference yet.

8 Conclusions

In this short progress report we describe the current status of the CLCERT Random Beacon project. We thanks the generous support by The Information Technology Laboratory (ITL) Grant Program, 2016-NIST-MSE-01, Measurement Science and Engineering (MSE) Research Grant Programs, from the National Institute of Standards and Technology, Dept. of Commerce, USA.

Bibliography

Gómez, Camilo; Hevia, Alejandro; Miranda, Sergio; Rojas, Cristián. *A publicly verifiable random beacon, and applications*. In preparation. October 2017.

McGraw, Gary. *Software Security: Building Security in*. Vol. 1. Addison-Wesley Professional. 2006.

Perrin, Chad. 2008. "The CIA Triad." http://www.techrepublic.com/blog/security/the-cia-triad/488. 2008. Last accessed, Sept. 2017.

Shostack, Adam. 2014. *Threat Modeling: Designing for Security*. John Wiley & Sons.