Internship Report: Protocol verification

Romain Soumard

25 mai 2020

Table des matières

1 Introduction

The present document is an internship report redacted during my last undergraduate year at the university of aix-Marseille from april to june 2020.

I already realized another intership in the Calculus division, in the COALA (Constraint, Algorithms and Applications) team, responsible for working, among other things, on CSPs (constraint satisfaction problems). This time, I realized my internship in the MOVE (Modelisattion and verification) team to see a more security focused theme of research.

During the course of my internship I was tasked with learning the basics of cryptograhpic protocols and study a real use case in the belenios voting system. In this document, you will find a presentation of the organisation in which I realized my internship, the LIS (laboratoire d'informatique et systèmes, french for computer science and systems laboratory.), a brief overview of cryptographic protocols, their properties, and the tools used to study and elaborate them.
Finally, I will answer the question I was asked during my internship, which is :

Is belenios suitable to administer a vote about internships at university level ?

2 The Computer Science and System Laboratory (LIS)

2.1 Presentation

The LIS is a Mixt Unit Of research (Unité Mixte de Recherche, UMR in french.), under the administrative supervision of the CNRS (Centre Nationale de Recherche, National Research Center.) which is split on different sites. Two of those sites are located in Marseille, at Luminy and Saint-jérôme, and one is located in Toulon. The LIS is actually linked with the universities of Toulon and Aix-Marseille and a lot of their members are actually teachers at those places. The researches led in the laboratory find several applications in different domains, such as energy, transport and healh, and the laboratory itself has an important contractual activity.

2.2 structure

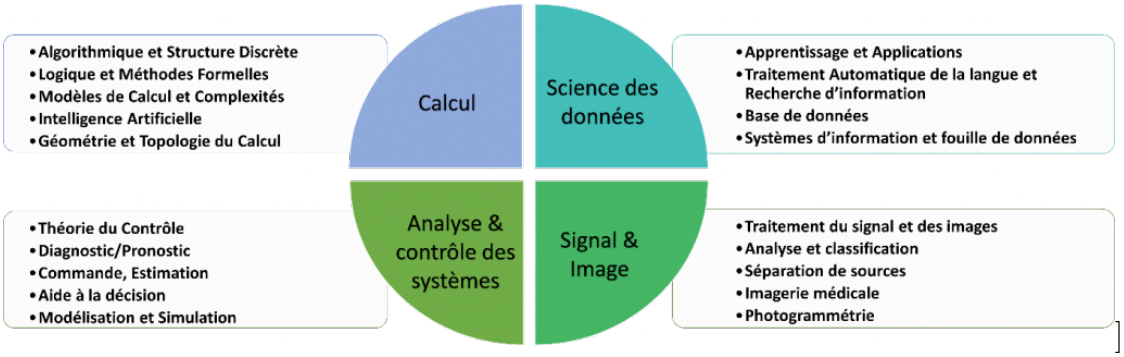

FIGURE 1 – LIS's Structure

The laboratory is divided into 4 divisions :

- Calculus
- Data Science
- Signal and Image
- Systems analysis and control

All of which are further divided in several teams such as COALA and MOVE as mentioned earlier. Among the researches led by those different divisions, there is artificial intelligence, simulation and modelisation, data bases, machine learning, image treatment and signal and so on.

3 Internship subject

As mentioned earlier,my internship subject consisted in studying the cryptographic protocols used in e-voting systems. All electronic voting systems that have been in used achieved varying degrees of security based on the protocol they use and implement. The use of those systems is especially interesting from a political and economical standpoint. In the following sections, I'll present Belenios, an electronic voting system and protocol developed by a team of researcher from the INRIA (Institut National de Recherche en Informatique et en Automatique, Computer Science and Automation National Research Institute.).

4 The study case of Belenios

4.1 Description of Belenios

Part of my intership consisted in studying the case of the Belenios system. Belenios is both the name of the electronic voting system and the name of the protocol used by it. It is mainly developed by Stéphane Glondu since 2012, and was proven by Dr Véronique Cortier her team to respect several security protocols properties.

4.2 Structure of the Belenios system

To run an election with Belenios, one can use the online voting platform at disposal on belenios.loria.fr. However, for the sake of studying Belenios, I installed it from the sources and delved into the research papers to learn more about the underlying structure.
An election relying on Belenios can have several configurations. The one that is mostly represented in the specification and research papers relies on 4 distinct entities :

- A registrar authority, tasked with providing the signing keys. Once the signing key is generated, it sends it to the proper voter and to the election server.
- The voting server, which is in charge of maintaining the bulletin board, and, by default also generate the credentials itself, in which case, the security properties guaranteed by Belenios are weaker, though organizing the election gets more simple.

- The voters, and, by extension, their voting device. The voters select their vote, and their device encrypt it, sign it for them, and send it to the voting server.
- The decryption trustees. Among a set of m trustees, $t + 1$ of them are need to decrypt the result of the election. The multiplication of the trustees allow for more neutrality during the tally in particular.

Depending on the configuration, the structure as well as the properties guaranteed by Belenios change. Most of the proofs presented in the research papers relies on the above structure with four distinct entities, and the idea that the voting server and registrar are not both dishonnest at the same time.

4.3 Properties of Belenios

Belenios is built upon another existing protocol named Helios. Hence, it offers the same guarantees. Its main difference is that it prevents ballot stuffing.

The main security properties guaranteed by Helios are :

- **Privacy :** In the context of voting systems, privacy refers to the inability of someone to know how you voted.
- **Strong verifiability** : Strong verifiability can be divided into two sub-properties :
  - **Individual verifiability :** A voter can check that his vote has been properly counted
  - **Universal verifiability :** Everyone can check that the results correspond to the ballots on the public board.

in order to insure those properties, Belenios uses several cryptographic primitive to build its protocol, which we are going to see in the next section.

4.4 Cryptographic primitives

Cryptographic primitives, as their name suggests, offer the basic functions with which are built cryptographic protocols.
Belenios make mostly use of four of them :

- **Encryption :** Belenios uses the El Gamal encryption system, which is an assymetric key encryption algorithm relying, as a lot of encryption algorithm, on group theory.

- **Hash function :** An hash function used to generate an hash from a message. Belenios uses a tag system to make the hash context-dependent and avoir hash collision.
- **Signature :** In Belenios, voters use a Schnorr signature to sign their encrypted ballot before sending it to the voting server. This mecanism helps to prevent ballot stuffing. Indeed, the voting server knows the signing keys, which can be used to authentify an honest voter from a dishonest one.
- **Zero-knowledge proof :** Belenios uses the Fiat-Shamir technique to provide non-interactive zero-knowledge proofs several times. One of them is to prove that voters encrypted a valid vote (what valid means is context dependant here.). Another one is to prove the decryption trustees correctly decrypted the result of the election.

4.5 Proof methods

In order to prove the strong verifiability and the privacy of Belenios, the research team used an interactive theorem prover called EasyCrypt, which supports the writing of cryptographic proofs. This allowed them to yield the first machine-checked analysis of ballot privacy and strong vverifiability on a deployed electronic voting protocol (To their knowledge at the least.). For all of the properties proved, **the team had to create a formal definition by creating several formal security experiments.** Since my time was limited, I could not try EasyCrypt myself however.
To establish privacy, the research team made the distinction between three aspects that impact the privacy of individual votes :

- Ballot privacy
- Strong correctness
- Strong consistency

Ballot privacy guarantees that ballot themselves do not reveal any information on the vote cast. Like most proofs I found in the papers, the idea behind the formal definition relies on the simulation of an adversary who is going to try to corrupt a part of the users. In the security experiment, the user gets the credentials of a part of the users. The proof makes use of an oracle for the vote. In the security experiment scenario created by the team, an adversary forces an honest id to challenge the oracle. To procude its proof, this oracle has to cast a vote. At this moment, the adversary corrupts the id and cast the vote on behalf of the honest voter.

Strong correctness guarantee that an honestly generated ballot will not be rejected by a ballot validation algorithm. Since the original definition of the notion is too strong to hold in a system with a registrat authority, the team decided to use a weaker version of it. To win, an adversary must, after having seen the list of credentials, choose an id, a vote and outputs a bulletin board. The adversary wins if for such a bord the honest vote for id wins.

Strong consistency requires that for any adversially produced bulletin board for which each individual ballot is valid, the value provided by the alorithm responsible of the tally is correct. To prove strong consistency, the team requires the definition of several algorithm which I am not going to detail here for simplicity sake. The idea behind the proof here is that the adversary is allowed to register a set of identities and then, with honestly generated keys for the election, generates a bulletin board that fakes the result of the tally if it tallied indivdually each vote.

To establish **Verifiability**, the team used the notion of strong verifiability already introduced in Helios-C. This notion takes into account the universal, individual and eligibility verifiability properties. They used tow different security experiments and scenarios to define verifiability :

- **Verifiability against a dishonnest ballot box**
- **Verifiability against a dishonnest registrar**

For the first scenario, the adversary takes entire control of the ballot box and can manipulate the votes in the box. It can require a voter to vote and retire his vote from the ballot box afterwards. The adversary can then manipulate the bulletin board to make the voter believe his vote was taken into consideration. It is important to note that ballot stuffing is not possible in this scenario or the next.

For the second scenario, the security experiment is quite similar, except that the adversary does not have the control of the ballot box but can generate credentials the way he see fit.

4.6 The organisation of an election

Now that we defined the entities and the cryptographic primitives used in Belenios, we're going to describe how an election is organized using it with the default configuration.

This example relies on the presence of 3 entities : the registrar, the voting server and the voters. We'll discuss other potential configurations and their influence on the system security and simplicity later.

Credential generation

During the first step, the voting server generates and send credentials to the voters, while the registrar generates signing keys and sends them privately to the voters and to the voting server. This allows the voters to sign their vote. With a zero-knowledge proof (see cryptographic primitives section), and the signature key, the voting server is then able to know whether a vote is honest.

Voting phase

During the second step, the voters use their credentials to connect to the web interface maintained by the election server. Then they select their vote which is encrypted by their own computer, sign it and send it to the server.

Tally phase

During the third step, and in the default configuration, the election server keeps the decryption key. To insure the privacy and the universal verifiability of the votes, while still being able to announce the result of the election, Belenios uses a very clever mathematical trick : It uses a property called homomorphism. This allows any person to compute the **encrypted** result of the the election from the encrypted ballots. Then, the decryption key can be used to decrypt the results.

4.7 Belenios source code

The source code of Belenios is written in Ocaml, a language that was mostly developed by the teams of the INRIA . Since I never studied it, I alas, could not understand much of it.

To create a clean install of Belenios I first tried to install it on a virtual machine. I encountered a difficulty. Indeed, it seems the 1.10 release of Belenios I was trying to install could not resolve certain dependencies.

Finally, I installed the gitlab version of Belenios on a Debian 10 stable Jessie by cloning it and was finally able to try it. To test the system, I used the scripts and Makefile contained in the software.

5 More on cryptographic protocols

Before studying Belenios, I had to learn some concepts and a few tools necessary to understand, at least at a basic level, the use and the functionning of cryptographic protocols.

5.1 Modelling

To model cryptographic protocols we use a wide variety of formal tools. Cryptographic primitives, modeled by an equational theory are used to represent most of the operations such as encryption, decryption, signature and so on.
On the top of that, we also use applied Pi calculus, which is a process algebra, allowing us to describe the protocol we want as a concurrent system in which several entities interact. An interesting thing is that pi calculus is very similar to the way we represent things in the Promela language for model-checking.

5.2 A few more properties

Besides the properties we already saw in Belenios, I had the occasion to learn a few more, namely :

- **Identification :** The act of indicating one's identity.
- **Authentification :** That is the act of proving an assertion. In Belenios, we saw it when the voter needed to use his credentials to prove he was who he was.
- **Secrecy :** This property is a bit harder to explain since I found several definitions of the term depending on the context. In general, secrecy refers to the practice of hiding information to non-authorized recipients. However, during my researches, I also learnt about forward secrecy (see below.)
- **Forward secrecy :** Forward secrecy refers to key agreement protocols (that is protocol where several entities agree on cryptographic keys to communicate.) An encryption system ensures forward secrecy if, during the key agreement phase, the examination of plain-text data exchange does not reveal the key that was used to encrypt the remainder of the session.

5.3 Belenios and the administration of a university election

To conclude on Belenios and to answer the question I was asked over the course of this internship, I would say that considering the stakes of a university level election

or vote, Belenios would be suited for the task. The system has already been used in such occasions successfully.

In the case of a vote from teachers The guarantee of privacy and strong verifiability met by Belenios would allow for example, the teacher to vote while reducing the risks of corruption and complicity since only the result of the vote could be decrypted and no one would be able to know how they voted (Keep in mind however that Belenios does not prevent coercion.). The verifiability property of the system would also allow the member of the university to check the result of the election all the while keeping the content of the vote secret. The same would be achieved in the case of an election.

However, I would strongly advise against using Belenios for large stake elections or vote **or** to administer a vote or an election including tech-savyy students (and potentially teachers). The same is advised by the research team, since no electronic voting system is currently able to achieve the same level of security as a traditionnal paper one.

6 Difficulties encountered

Over the course of the internship, I encoutered a few difficulties, namely :

- The installation of belenios which took me several days because of an unsolvable dependency problem in the 1.10 release.
- The lack of mathematical prerequisites which made my study of the research paper much harder.
- The use of oCaml basically prevented me to study or debug the source code of Belenios. The only thing I could actually understand were the Makefile and the shell scripts.
- The wide number of research papers took me a while to go through to know exactly which part was interesting to review and which part were not.

7 Conclusion

In conclusion, I learrnt many things over the course of this intership, despite the many problems encountered and the short time I had left to work. I have mostly learnt the existence, and partly the use (but not the mastery) of several mathematical tools such as process algebras, like the pi calculus, the equational theories,

several cryptographic properties such as privacy and verifiability, and the first steps of a research work. I also had the occasion to get acquainted with the ocaml language and the tools surrounding it, especially opam, the package manager, which I found very interesting and efficient in its syntax. I also learnt more about voting protocols, their functionning, their use, and the stakes behind their development. Considering my intention to pursue a researcher career in the field of security and / or Artificial Intelligence, this internship was a good first introduction to the problems related to research

8 Addendum : Problems related to the internship organisation

During the realization of the internship, a lot of problems arised that prevented me to produce what I consider a proper work. The confinement, first, caused by the Covid-19 crisis forced us to stay at home while working remotely, making the support for the work harder. The work environment was far from ideal as well, considering my fammily situation.

The upholding of the exams in the middle of the internship made things even worse considering the chaos in which it was organised. From the April 30th to May 19th, we had to work on the internship while also reviewing for the exams (5 TD for each subject more the rendering of the network program with the redaction of the report.).

The administration tasks also slowed us down, since it required some time to procude CV, letters and the papers necessary to candidate to this university or others.
Out of the five weeks first intended for the internship, I would say only two of them could be properly exploited.

Considering all these problems, I had to think about a few ways to improve this work. Hence, with more time, this work could have used a bit more polish to eliminate innacuracies. I could have also proposed a real research work by attempting to reproduce the proof system created by the team of Belenios in EasyCrypt. I could have also widen my studies to the use of cryptographic protocols in general. If Dr.Lugiez agrees, I intend to work a bit longer on this topic as an introduction to research.

9 bibliography

Coming...