UniCrypt - A cryptographic library implemented in Java

Project Report

Juerg Ritter (rittj1@bfh.ch)

Bern University of Applied Sciences
Engineering and Information Technology

June 17, 2012

Abstract

The Bern University of Applied Sciences is doing research in the area of E-Voting. One particular field of their research is to evaluate existing approaches by implementing them into real applications. These implementation projects are usually done as student projects. Also, the E-Voting Group of the Bern University of Applied Sciences is currently implementing an E-Voting system which will be used in practice by the University of Bern. E-Voting systems often rely on the same cryptographic building blocks. Until now, these building blocks have been implemented from scratch for every project. The aim of this project is to build a cryptographic library which contains the most important building blocks. This library can then act as a foundation for new projects in the area of E-Voting.

1 Introduction

One of the research fields of the Bern University of Applied Sciences is the area of E-Voting. E-Voting has become a big field of research in the past couple years. Still, there is no generic approach which meets all the criteria such as privacy, transparency, etc. which we want in E-Voting. The E-Voting research group of the Bern University of Applied Sciences [1] tries to improve this situation with the following approaches:

- Develop new approaches and provide them to the community for review
- Take existing approaches and evaluate them in terms of practicability. These approaches are usually available as scientific papers

The evaluation of these approaches is usually done by implementing them into a prototype level application to show that the approach actually works. In the past, a couple of student projects dealt with this kind of implementation. Furthermore, the E-Voting group has been charged with the implementation of an E-Voting system which can be used to conduct the election of the student organization of the University of Bern, Switzerland.

Normally, all the approaches in the area of E-Voting rely on the same so called cryptographic building blocks (see chapter 2 on page 2). Until now, these cryptographic building blocks have been reimplemented in each project, which is time consuming and error prone.

The goal of this project is to implement a cryptographic library which serves as a foundation of new implementations of E-Voting systems. It implements the three most commonly used cryptographic building blocks:

- ElGamal crypto system
- DSA signature system
- Zero Knowledge Proofs

The implementation is done using the Java programming language. With regards to further enhancements of the functionality of the library, the design of the library has to be such that extensions are easy to implement. It has to allow developers to build E-Voting applications sigificantly faster and less error prone. The timeframe of this project is a 9 ECTS project during the MSE studies of the Bern University of Applied Sciences.

2 Background and Related Work

This section roughly explains the theoretical background of the cryptographic building blocks which are implemented during this project. It also reviews similar projects which are already available.

2.1 Cryptographic building blocks

The currently available approaches for building E-Voting systems often rely on the same three cryptographic concepts. We call these concepts cryptographic building blocks. The upcoming sections shortly discuss these building blocks.

2.1.1 ElGamal crypto system

The ElGamal crypto system [7] is the asymmetric cryptosystem which is mostly used in the context of E-Voting. An asymmetric cryptosystem uses two keys to operate, one which is used to encrypt a certain message (the public key) and another to decrypt the message (the secret key). Another well known example of an asymmetric cryptosystem is RSA [8].

2.1.2 RSA crypto system

RSA [8] is a crypto system which appeared in 1978. It has been invented by Ron Rivest, Adi Shamir and Leonard Adleman, hence the name RSA. The RSA algorithm is based on the problem of prime number factorization which is considered as a hard problem with sufficiently large numbers. RSA can be used in two ways, as a crypto system and a signature system. This package contains only the crypto system though. This library contains the signature system as well in a different package.

2.1.3 RSA signature system

The RSA signature system [8] can be used to digitally sign and verify a message. It is based on the same technique as the RSA crypto system described above.

2.1.4 DSA signature system

DSA (Digital Signature Algorithm) [6] is a standard signature schema which is used in several places in the E-Voting area. Signatures are used to guarantee the integrity of a message, meaning that the sender can clearly be identified and the message has not been altered on its way to the receiver. Just like the asymmetric cryptosystem, it uses two different keys to operate, a secret key to generate a signature and a public key to verify a signature.

2.1.5 Schnorr signature system

The Schnorr signature [9] is a digital signature system developed by Claus-Peter Schnorr. Like ElGamal, the signature security relies on the hard problem of calculating logarithms in cyclic groups.

2.1.6 Zero Knowledge Proofs

Zero Knowledge Proofs [10] are used in several areas of E-Voting. The involved parties are generating such proofs to demonstrate that they are following the specified protocol. In many cases, Zero Knowledge Proofs need to be combined with logical operators in order to demonstrate that an entire combination of values are valid.

2.2 Existing community projects

In the cryptographic community a few projects have evolved which cover more or less the need that we have for a cryptographic library for E-Voting. All of them address a particular topic, but none of them implements all the components which are needed for this project. The following section should give an overview what exists nowadays and how it could be included or used in this project.

2.2.1 Qilin

Qilin [4] is supposed to be an experimental framework which allows to build cryptographic protocols. It is developed by Tal Moran, a cryptographer at the Harward University. Qilin is an open source project and has been released under the conditions of the MIT/X11 licence. It contains implementations such as the ElGamal Cryptosystem or Zero Knowledge Proofs. In the background it makes use of the Bouncy Castle library [2], for example for digests (hashfunctions) and Elliptic Curve Group parameters. Qilin is designed to be an experimental library. The developer intends to build a simple framework without focus on high security or performance optimization.

Advantages

- It focuses on cryptographic protocols which is exactly what we need
- Almost all bricks are implemented in two ways, with Elliptic Curve Groups (ECC) and Multiplicative Modular Groups
- Qilin is implemented in Java
- The ElGamal implementation is a textbook implementation. Bouncy Castle in contrast uses mechanisms such as padding which makes it impossible to do calculations with the ciphertext. This is used for example in relation with Zero Knowledge Proofs.

Drawbacks

- Qilin doesn't implement any kind of signature mechanisms
- Generalisation is driven to the extreme which makes it rather difficult to understand
- At some points the design of the library could be better
- Qilin is designed as a prototyping library and is apparently not heavily tested in critical production environments

2.2.2 Bouncy Castle

Bouncy Castle [2] is an open source implementation of the JCA (Java Crypto API) specification and has therefore an interface to the Java Standard Library. It has been released under the conditions of the MIT/X11 licence. It enhances the capabilities which are defined in the Java Crypto API specification with additional cryptographic algorithms. It also implements the ElGamal Cryptosystem which is heavily used in the area of E-Voting.

Advantages

- Bouncy Castle is a stable library which is used in practice
- It is based on Java
- It integrates tight into Java using the Java Cryptographic API (JCA)
- Bouncy Castle can be used as a security provider of JCA which can be useful for dealing with digests. Digests (also known as hashes) are required to generate Zero Knowledge Proofs and digital signatures.

Drawbacks

- The ElGamal implementation is not designed to use in a numerical way
- The focus of the Bouncy Castle library is on more generic cryptographic applications, so many implementations are on a too high level

2.2.3 Verificatum

Verificatum [11] is an open source implementation of a secure Mix Net. Mix Net are used for example to anonymize ballots or keys by shuffling them in a certain way. Verificatum has been released under the terms of the GNU Lesser General Public Licence (LGPL). It is developed as a part of a research project by Douglas Wikstroem at the Royal Institute of Technology of Sweden.

Advantages

- Mix Nets are strongly related to E-Voting
- Main parts are implemented in Java
- Makes use of Zero Knowledge proofs

Drawbacks

None

2.2.4 CACE NaCL

CACE stands for Computer Aided Cryptography Engineering and is an European project whose goal is to implement a secure cryptographic toolbox which allows to build secure applications. NaCL (Networking and Cryptographic library) [3] is a subproject of the CACE Project and is a library of cryptographic functions. It is implemented in C and there are language bindings available for C++ and Python.

Advantages

None

Drawbacks

- Currently there are no language bindings for Java available
- Does not focus on E-Voting related topics
- The licencing is not clear

2.2.5 VIFF

VIFF [5] is a Python based, lightweight framework which allows to do secure multi-party computations. For this kind of computations, Zero Knowledge Proofs are required. VIFF is open source software and released under the conditions of the GNU Lesser General Public Licence (LGPL).

Advantages

- Focuses on cryptographic protocols
- Implementation is object oriented

Drawbacks

- The project is implemented in Python

2.2.6 Student projects

Lead by the E-Voting group [1] of the Bern University of Applied Sciences, a couple of E-Voting protocols have been implemented to check how they behave in practice. Protocols in the area of E-Voting often rely on the same building blocks such as the ElGamal cryptosystem, Zero Knowledge Proofs, etc. This means that all the required building blocks have been implemented already in practice. These implementations were made for this specific problem only, and therefore the code is not designed to be reusable.

Advantages

- All the required building block are implemented
- Implemented in Java
- In-house knowledge

Drawbacks

- The implementations did not focus on reusabilitiy

3 Results

The result of this project is a cryptographic library called **UniCrypt**. The name UniCrypt is derived from the name of the UniVote project of the E-Voting group. UniCrypt is implemented in Java which implements cryptographic building blocks which are often used in the context of E-Voting. It implements the following cryptographic building blocks:

- ElGamal crypto system
- RSA crypto system
- RSA signature system
- DSA signature system
- Schnorr signature system

The implementation of all these building blocks is based on the mathematical concept of groups and their elements. The UniCrypt library reflects this concept with the implementation of the whole mathematical part, including different kinds of mathematical groups.

A big advantage of UniCrypt is certainly that it does not rely on third party libraries such as Bouncy Castle. The only thing required to use UniCrypt is a Java Standard Edition Runtime environment. One of the advantages of this concept could be a deployment on mobile devices, for example smartphones or tablet computers.

3.1 Design

Since a library acts as a foundation for projects, the codebase should fairly stable to make life for the users as easy as possible. For this reason, it is really important to have a proper design. Another important aspect to consider is the addition of new features after this project. Of course, there are more building blocks which are candidates to be added to this library. From a design perspective, it should be fearly easy to introduce such extensions. For this reason, the first stage of this project was done on design level, trying to fit all the components we need into a class diagram. Periodical reviews of the design have been done in order to steer the project in a good direction. After that first stage of designing, the mode has been switched to an iterative approach. Further improvements were made based on findings during the implementation of the initial design. Many iterations were necessary to find a proper design to rely on. The final design of the library can be found in the documentation of this project.

In order to verify the extensibility and flexibility of the design, the RSA crypto- and signature algorithms have been implemented. The implementation of these components

helped to gain a different perspective and improve the design so that it also works for RSA (and hopefully all the other components which will be implemented in the future).

3.2 Testing

During implementation, several testcases have been implemented. The testcases have been designed such that the little compontents can be tested individually. The framework which has been chosen to implement and run the tests is JUnit 4. The big advantage of this framework is the high maturity, the popularity in the Java development community and the good integration in several Integrated Development Environments (IDEs). By the end of this project, 84 testcases have been implemented. All these testcases passed sucessfully.

3.3 Documentation

A library needs to have a good documentation which shows users how to use it properly and efficiently. The user documentation of UniCrypt is included in the Javadoc documentation. Javadoc allows to include documentation directly in the source code and generates a set of HTML pages which can be displayed in a browser. Javadoc is a very common technique to document Java code and therefore most Java developers know how to use it.

The codebase of the project also contains executable code examples, one per implemented building block. The code can be found in the "examples" package.

3.4 Problems

As in most projects, some problems occured during the project phase. From a timing perspective, the design of the library turned out to be more time consuming than anticipated, a lot of iterations and refactoring had to be done before the final design was in place.

Also, the first concept on how mathematical functions should be implemented turned out to bring in too much complexity. As a consequence, the implementation of the Zero Knowledge Proofs had to be abandoned due to time- and complexity constraints. The work which has been done in this area has not been deleted, but moved into a different tree. The components of this concept are not a part of the library by the end of this project, but it leaves the option to reinclude these components in the future.

DSA was the third problem area in this project. As it turns out, DSA cannot strictly be implemented with group operations as it was done in the implementation of the other cryptographic building blocks. This problem could be solved by implementing operations on groups which are strictly speaking no group operations. The DSA algorithm itself works, but it violates some design guidelines which have been defined. The Schnorr signature algorithm has been implemented as an alternative signature algorithm which relies on the discrete logarithm problem. Schnorr can be implemented without violating the guidelines.

4 Future Work

The development of UniCrypt is certainly not finished. The E-Voting group of the Bern University of Applied Sciences will continue to integrate features into the library. One thing which needs to be implemented for sure are Zero Knowledge Proofs. Other possible features include elliptic curve groups, threshold crypto systems or symmetric crypto systems.

After UniCrypt has reached a certain maturity, there is the possibility to release the code of UniCrypt as open source and make the code accessible to the community. This step could help to improve the quality further in terms of functionality and security.

5 Conclusion

The UniCrypt Library can be used as a foundation for applications in the area of E-Voting and other applications which rely on secure protocols. At this stage, UniCrypt implements the following building blocks:

- ElGamal crypto system
- RSA crypto system
- RSA signature system
- DSA signature system
- Schnorr signature system

It also implements all the mathematical concepts which are used in these building blocks. A lot of effort went into a proper object oriented design. This allows to introduce new features fairly easy. The idea of introducing the Schnorr signature algorithm into the library came quite late in the project, and it could be implemented very efficiently without spending much time. This shows nicely that the design of the library can't be all bad.

UniCrypt can certainly help to speed up the development in upcoming projects in the area of E-Voting and secure protocols. Since not all concepts have to be implemented from scratch, the time budget of the project can be spent more efficiently, focusing on the problem which the project tries to solve. UniCrypt can also help students to dive into the world of E-Voting. From own experience, getting started is not always easy. Experimenting and looking at the code often helps to understand concepts faster.

Acknowledgments

I would like to thank all the members of the E-Voting group of the Bern University of Applied Sciences who contributed a great deal by giving valuable input in constructive discussions. A special thanks goes to my advisor Prof. Dr. Rolf Haenni, with whom I spent many hours in front of the whiteboard, developing and discussing concepts and techniques on how

to implement this piece of software. These discussions also helped to increase the deeper understanding of the cryptographic concepts which are behind E-Voting.

References

[1] BFH E-Voting Group, *E-Voting Group*. http://e-voting.bfh.ch/, 2012. [Online; accessed 09-June-2012].

[2] Bouncy Castle Project, *Bouncy Castle*. http://bouncycastle.org, 2012. [Online; accessed 09-June-2012].

[3] CACE Project, *NaCl: Networking and Cryptography library*. http://nacl.cace-project.eu/, 2012. [Online; accessed 09-June-2012].

[4] T. Moran, *The Qilin Crypto SDK*. http://qilin.seas.harvard.edu/, 2012. [Online; accessed 09-June-2012].

[5] VIFF Project, *VIFF, the Virtual Ideal Functionality Framework*. http://viff.dk/, 2012. [Online; accessed 09-June-2012].

[6] Wikipedia, *Digital Signature Algorithm — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/Digital_Signature_Algorithm, 2012. [Online; accessed 09-June-2012].

[7] ——, *ElGamal encryption — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/ElGamal_encryption, 2012. [Online; accessed 09-June-2012].

[8] ——, *RSA (algorithm) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/RSA_(algorithm), 2012. [Online; accessed 09-June-2012].

[9] ——, *Schnorr signature — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/Schnorr_signature, 2012. [Online; accessed 09-June-2012].

[10] ——, *Zero-knowledge proof — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/Zero-knowledge_proof, 2012. [Online; accessed 09-June-2012].

[11] D. Wikstroem, *Verificatum*. http://www.verificatum.org/, 2012. [Online; accessed 09-June-2012].