CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 26, 2017

PRIVACY PRESERVING CLOUD, EMAIL, AND PASSWORD SYSTEMS

PREPARED FOR

OSU

ATTILA YAVUZ

PREPARED BY

GROUP 38

THE SECRET BUNNY TEAM

SCOTT MERRILL

Abstract

This document will analyze various pieces of the Privacy Preserving Cloud, Email, and Password Systems project that will be needed to complete the project. Within these pieces we will look at multiple options and decide on the best fit option to meet the goals and requirements.

CONTENTS

1 Programming Language

1.1 Overview

Programming languages come in all shapes and sizes. This is great as it allows for a lot of flexibility when deciding to start a project. Unfortunately, it can also be daunting as one can find it hard to differentiate what makes ones language better than another.

1.2 Criteria

The selection of which programming language to use, is essential before our group can truly begin implementation and should also be considered before we begin design as well. When looking at different programming languages, we will consider these various factors: technical characteristics, ease of learning, ease of understanding, speed of development, supported platform environments, fit for purpose, project groups predisposition, project requirement needs, and languages used in projects used for comparison. These points will help our group to decide on a language that will work best for all group members, allows us flexibility within development, and meets the needs of client within time restraints.

1.3 Potential choices

We will look at three potential choices for languages and determine which choice is best for this project.

1.3.1 C++

C++ is a general-purpose language that has access to imperative and object-oriented programming features. It is a low-level language means that most of the data structures we use will need to be built from data structure primitives. This is useful because our project relies on the ability to construct data structures to enable the ability to search across encrypted data. C++ has also been around for a long time, meaning there is plenty of resources available. One resource our group will be relying on is the similar project that our client has provided for us. An implementation of a different DSSE scheme that we have been advised to use as a template when creating our own. This other DSSE scheme will also be used in comparison when we analyze the success of our project.

1.3.2 Java

Java is also a general-purpose language like C++ and has access to many features that C++ does. Java is a high-level, garbage collected language meaning it can be an easier language to work with. This is an important point to consider when finding a language that all group members can use. Java has many libraries at its disposal as well this is essential to the project as we will need to use different crypto and data structure libraries when implementing our DSSE Scheme. A simplistic implementation of the DSSE implementation that we are focused on has already be accomplished in Java and will be used as a comparison in our analysis at the end of the project.

1.3.3 Python

Python is also a general-purpose language like C++ and Java sharing features such as being object-oriented and imperative. Java is a high-level language and is interpreted rather than compiled. Being an interpreted language allows for more readability and allows for syntax that can express concepts in fewer lines. These are also many platforms and frameworks available to use python on which can be useful when deciding on a cloud API to pair with the project.

Unfortunately, because it is an interpreted language it needs an interpreter which does not allow for low-level work which may provide some challenges when trying to implement the DSSE.

1.4 Discussion

When comparing the different languages we need to look at what the group members are most comfortable in, or which language can be learned by the group the easiest. C++ is a language that all three members know well and is a solid choice. Python would also be a good choice as it is perhaps the easiest language to learn and some members in the group already have extensive experience working with Python.

1.5 Conclusion

We chose C++ as our language to use when implementing the DSSE. The justification for this is that it allows for all the features we need as well as a variety of choices when deciding on cryptology primitives. In addition the DSSE scheme that our client provided is written in C++ and can be used as a template when implementing our code.

2 Cryptology Library

This section will look at three potential cryptology libraries we might use in our project

2.1 Overview

Deciding on a crypto-library is an important decision because it will be what the group uses to encrypt data that will be used in our final DSSE scheme.

2.2 Criteria

The critera we will look at when deciding on a crypto-library will consist of:

- Types of cryptology algorithms available
- Validation of the library
- Ease of use

2.3 Potential choices

2.3.1 Crytpo++

Crypto++ is a free and open source library of cryptographic algorithms written in C++. It is widely used in academia and student projects as well as businesses. the library fully supports 32-bit and 64-bit architectures for many major operating systems and platforms, including Android, Apple and Linux. The project also supports compilation using C++03, C++11 and C++17 runtime libraries. Crypto++ 5.5.2 was the top performing library under two block ciphers, and did not rank below the average library performance under the remaining block ciphers.

2.3.2 OpenSSL

OpenSSL is a software library for applications that require secure communications over computer networks. OpenSSL contains open-source implementation of SSL and TLS protocols with a core library written in C. The library implements basic cryptographic functions and provides a variety of utility functions. Versions are available for most Unix and Unix-like operating systems.

2.3.3 Tomcrypt

"LibTomCrypt is a fairly comprehensive, modular and portable cryptographic toolkit that provides developers with a vast array of well known published block ciphers, one-way hash functions, chaining modes, pseudo-random number generators, public key cryptography and a plethora of other routines."

2.4 Discussion

when it comes to cryptographic libraries all three of these will work well with our selected languge (C++). All are written in either C or C++ which aligns with the programming language we have selected.

2.5 Conclusion

Since all three libraries: Tomcrypt, OpenSSL, and Crypto++ meet our criteria, we have decided to go with Tomcrypt which has been used in our Clients implementation. This will be useful for when we look at speed comparisons between the two as there will be less variability.

3 Message Authentication Code

3.1 Overview

David Cash's DSSE algorithm requires a variable length PRF(pseudorandom function) as part of the encryption scheme. Essentially that this means is that we will MAC (Message Authentication Code) as a way to confirm our transmitted message.

3.2 Criteria

For this section we will look at three types of criteria

- Is the MAC proven to be secure
- The MAC is sufficiently fast to not slow down the system
- Availability of the MAC

3.3 Potential choices

Include Potential choices

3.3.1 HMAC

HMAC, or hash-based message authentication code, was created in 1997 and was one of the first MAC schemes to be used. This makes it one of the simpler or primitive types of MAC and has many possible implementations available. HMAC also has been tested and proven to be a secure option as a MAC.

3.3.2 CMAC

CMAC or Cipher-based Authentication code, is made using a block cipher along with a secret key. CMAC can be used similarly to HMAC in creating a authentication code to pass along with a message. According to a study [4] CMAC performs at an average speed of 27.75 (picoseconds) per packet compared to HMAC at 28.03 (picoseconds) per packet.

3.3.3 PMAC

PMAC, or for Paralleliz-able MAC, is a message authentication code algorithm created by Phillip Rogaway. PMAC is a method of taking a block cipher and creating an efficient message authentication code that is reducible in security to the underlying block cipher. PMAC is similar in functionality to the CMAC algorithm. PMAC solves one thing that CMAC struggles with and that is you can't compute C[i] until you've finished computing C[i-1]. This "bottleneck" can be a problem if you want to MAC at very high speeds. [5]

3.4 Discussion

For this option all three meet the speed and security that we are looking for in our criteria. There are implementations of all three that are available in the Tomcrypt library. PMAC seems to be a straight upgrade to CMAC if we are looking for scalability. HMAC is by far the simplest one to choose.

3.5 Conclusion

we chose HMAC because our project does not call for any measure of scalability since it is just a proof-of-concept. HMAC has the simplest of implementations and does everything we are looking for and therefore make it the best chioce allowing us to concern ourselves with more complicated matters.

REFERENCES

[1] Attila Yavuz. "Privacy-Preserving Cloud, Email and Password Systems." *CS Senior Capstone* http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?project=334

[2] Attila Yavuz and Jorge Guajardo. "Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Updates on Commodity Hardware" *ACM SAC 2015* http://web.engr.oregonstate.edu/ yavuza/Yavuz_DSSE_SAC2015.pdf

[3] David Cash et al. "Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation" *Cryptology ePrint Archive, Report 2014/853* https://eprint.iacr.org/2014/853

[4] Performance Comparison of Message Authentication Code (MAC) Algorithms for the Internet Protocol Security (IPSEC). *Janaka Deepakumara, Howard M. Heys and R. Venkatesan* http://citeseerx.ist.psu.edu/viewdoc/

[5] PMAC: Background *http://web.cs.ucdavis.edu/ rogaway/ocb/pmac-bak.htm*

[6] Choosing a Programming Language *Cris Britton* https://msdn.microsoft.com/en-us/library/cc168615.aspx

[7] What are the pros and cons vs. C++, Python and Java? *Cody Jackson* https://www.quora.com/What-are-the-pros-and-cons-vs-C++-Python-and-Java

[8] LibTom *2017 Team libtom* http://www.libtom.net/LibTomCrypt/