Biometric Bit locker

Project Plan

Yousef Al-Absi

Cole Alward

Morgan Anderson

Larisa Andrews

Ammar Khan

Justin Kuhn

Problem Statement

Our problem is the inability to fully encrypt phone data because of the lack of a TPM(Trusted Platform Module) chips on Android phones. We can solve the problem by dynamically generating the key using a PUF(Physical Unclonable Function).

Project Deliverables and Specifications

Goals:

- To develop an open source PUF library.
- To make the PUF work as a lock screen by asking a user to draw shapes to unlock the phone and authenticate then properly.

Deliverables:

- A well tested PUF Java gestures library
  - An open source library that should be released with unit tests and rewritten methods. We need to provide a valid testing framework and an appropriate architecture for the software.
- A PUF based Android app
  - The app should act as a lock screen whenever the phone is closed, it'll authenticate users by asking them to draw a shape and it should only work for the correct user. The app will also automatically start when the phone boots.

Dates for deliverables detailed in timeline section.

Previous Work/Literature Review

For our project, we will be working with existing software and integrating it into our project. The given implementation is named PUF. Presently, the PUF creates a unique key from data received from the user. It receives this data by displaying a pattern on the screen and forcing the user trace the design several times. From this data, a function will find a trace range the user should always fall between. When the user attempts to draw the shape of the design again, it will recognize the user. For more in-depth information about the PUF, research and documentation provided by our client for the current code is on our website under project documentation.

When considering whether similar products are in the market, no comparable implementation has been found. However, there are similar technologies available that offer different functionalities. PUF is not a new way of securing a device. In fact, the most common use of this technology for securing phones has been the fingerprint scanner, which is a property found on most mobile phones. Additionally, the idea of an Android bit locker is not new either. Having a secure way to unlock one's data once it has been encrypted has been a problem that many are trying to solve. Microsoft has a dated bit locker that uses a key pair to encrypt and decrypt data. Their solution utilizes a USB stick, a PIN or both to be able to authenticate the user. More information about this can also be found on Microsoft's website.

Proposed Design/System/Solutions

For our project, some of the groundwork for the solution has already been completed. The current solution is to develop an Android application that uses data generated from pressure readings returned by the screen when the user traces a generated shape like the Android unlock pattern. The data would be generated by a PUF, which has already been implemented. The pattern tracing and user authentication features have also been implemented but need to be updated, reworked and thoroughly tested before we can deem them functional or reliable. Currently, the goal of the solution is to integrate this application into the Android OS so we could have application level encryption. This may look like a series of patterns that you have to trace and pass when you open a certain application. If we can get this functionality in place, we will begin to move the encryption further back into the boot up process. Our end goal is to be encrypting and requiring user authentication at the kernel level before the OS is booted, which is like the bit locker on Windows computers. The feasibility of this solution is currently unknown, and research is being done to see how if we could implement this feature at all.

Assessment of Proposed Solution

The proposed solution has a strength that most other encryption methods contain, and that is PUF implementation. When functional, this allows some of the most effective authentication available, as only the user on their exact phone has access to their application data. Having data saved at an application level allows only specific data to be encrypted, which can be a benefit if there are certain file systems of the device that must stay unencrypted for general functionality to resume, but could also be seen as a weakness as the user's operating system wouldn't be given the same protection, making the device susceptible to attacks still. A trade-off of boot and kernel level encryption would be that several usual background applications and general functionalities of the device would need constant authentication from the user to be unlocked and allowed to operate, which isn't very user friendly.

Validation and Acceptance Test

The largest functional requirement of this product is that a user, and only the user, can unlock their device by tracing the given pattern. To validate this requirement, no one other than the user who created the device profile may unlock or decrypt user data. An acceptance test for this requirement pertains a full system test in which an owner can create a device profile themselves to lock and encrypt their device data and should be able to unlock and decrypt their data with at least a 95% acceptance rate. Any other user should have under a 0.1% success rate unlocking the same device with the owner's device profile. As a related non-functional requirement, the user should be able to unlock or be denied access to the device in under 3 seconds.

Application data should have to be unlocked by our application for the user to access it. Data should not be accessed unless unlocked by the owner. An acceptance test case would require the device to invoke our authentication layout before continuing to allow the user to access application data, or during the device unlock screen, such that data is never accessed without authentication.

Project Timeline

The schedule for this project is based on the rolling wave planning technique, which enables the team to discuss and make decisions concerning the project as it progresses. Tasks that should be completed soon are discussed in-depth, whereas tasks scheduled for later dates are discussed in a more abstract, high-level manner. The current team is composed of individuals who have the necessary knowledge of the project parameters, hardware and software used in the project to make educated estimates for the projected times. Once the current tasks are completed and it is time to analyze the subsequent tasks, dates may be modified depending on the coming in-depth discussions and knowledge of the team.

The following diagrams present the process (**Figure 1**) and projected schedule (**Figure 2**) our team will utilize over two semesters, respectively

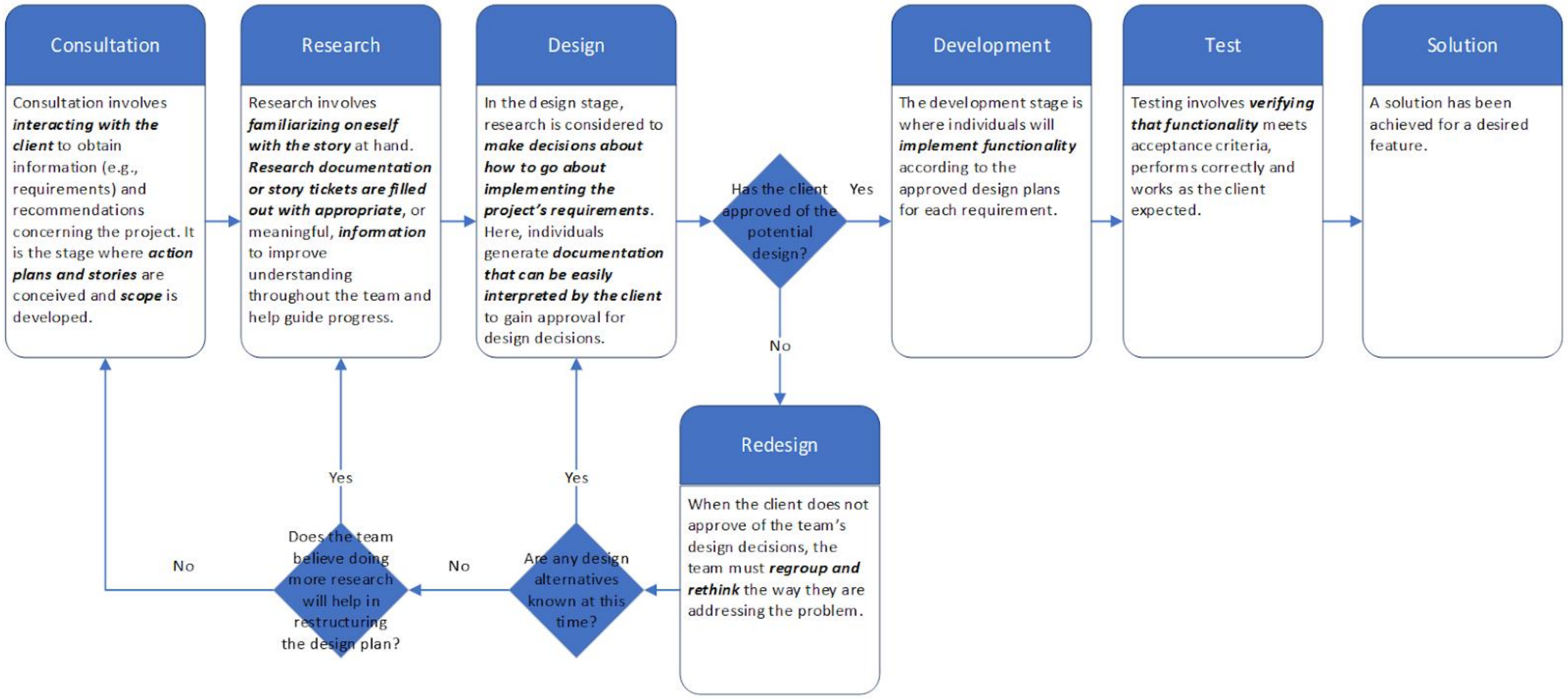
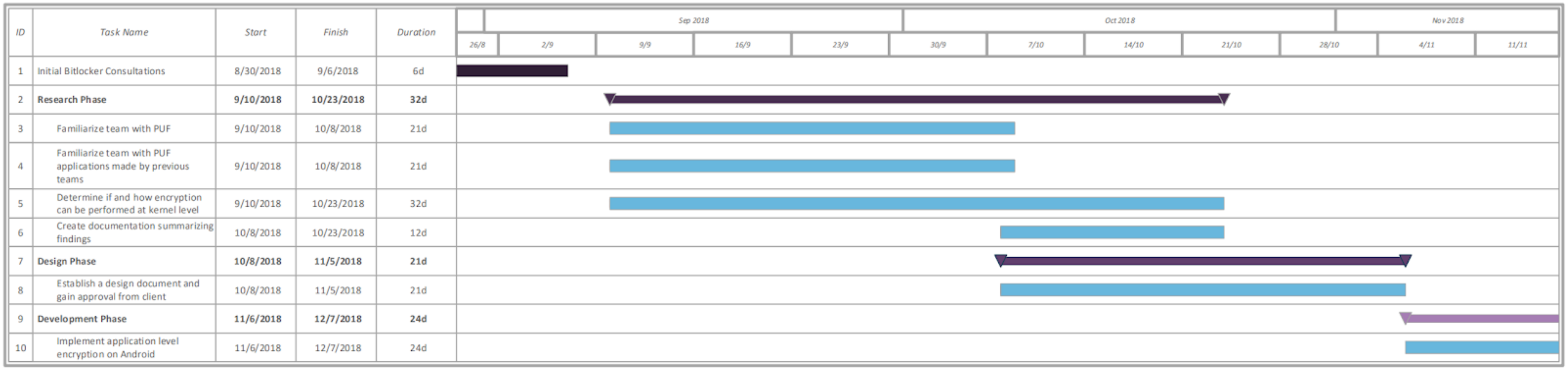

Consultation
Consultation involves ***interacting with the client*** to obtain information (e.g., requirements) and recommendations concerning the project. It is the stage where ***action plans and stories*** are conceived and ***scope*** is developed.
Research
Research involves ***familiarizing oneself with the story*** at hand. ***Research documentation or story tickets are filled out with appropriate***, or meaningful, ***information*** to improve understanding throughout the team and help guide progress.
Design
In the design stage, research is considered to ***make decisions about how to go about implementing the project's requirements***. Here, individuals generate ***documentation that can be easily interpreted by the client*** to gain approval for design decisions.
Has the client approved of the potential design?
Yes
No
Development
The development stage is where individuals will ***implement functionality*** according to the approved design plans for each requirement.
Test
Testing involves ***verifying that functionality*** meets acceptance criteria, performs correctly and works as the client expected.
Solution
A solution has been achieved for a desired feature.
Redesign
When the client does not approve of the team's design decisions, the team must ***regroup and rethink*** the way they are addressing the problem.
Are any design alternatives known at this time?
Yes
No
Does the team believe doing more research will help in restructuring the design plan?
Yes
No

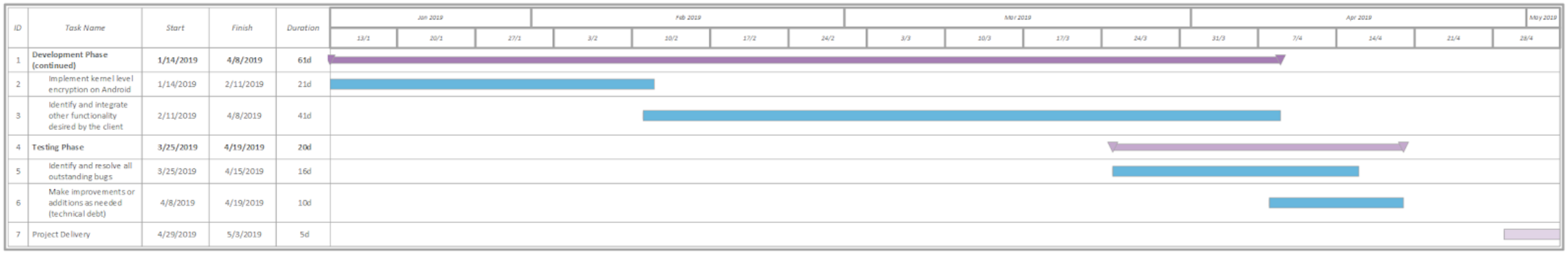
Figure 1: Process workflow

Project Schedule Fall 2018



ID	Task Name	Start	Finish	Duration
1	Initial Bitlocker Consultations	8/30/2018	9/6/2018	6d
2	**Research Phase**	**9/10/2018**	**10/23/2018**	**32d**
3	Familiarize team with PUF	9/10/2018	10/8/2018	21d
4	Familiarize team with PUF applications made by previous teams	9/10/2018	10/8/2018	21d
5	Determine if and how encryption can be performed at kernel level	9/10/2018	10/23/2018	32d
6	Create documentation summarizing findings	10/8/2018	10/23/2018	12d
7	**Design Phase**	**10/8/2018**	**11/5/2018**	**21d**
8	Establish a design document and gain approval from client	10/8/2018	11/5/2018	21d
9	**Development Phase**	**11/6/2018**	**12/7/2018**	**24d**
10	Implement application level encryption on Android	11/6/2018	12/7/2018	24d

Project Schedule Spring 2019



ID	Task Name	Start	Finish	Duration
1	**Development Phase (continued)**	**1/14/2019**	**4/8/2019**	**61d**
2	Implement kernel level encryption on Android	1/14/2019	2/11/2019	21d
3	Identify and integrate other functionality desired by the client	2/11/2019	4/8/2019	41d
4	**Testing Phase**	**3/25/2019**	**4/19/2019**	**20d**
5	Identify and resolve all outstanding bugs	3/25/2019	4/15/2019	16d
6	Make improvements or additions as needed (technical debt)	4/8/2019	4/19/2019	10d
7	Project Delivery	4/29/2019	5/3/2019	5d

Figure 2: Projected Schedule

Challenges: Risks/Feasibility Assessment, Cost Considerations

Presently, the feasibility of our project is undetermined, for it is unknown whether encryption can be performed at the kernel level. Android does not use encryption at the kernel level in its own releases, so it may be the case that it is impossible. There are other risks that are related to PUF itself, such as the ability to update and integrate it into our solution. However, given our previous work with each other and progress already completed, we do not believe the project to be outright infeasible. Most of our project will be working with Android, which we all have experience working together on. We created a user application, which is like our initial goal for this project. The project also heavily deals with operating system and kernel levels of programming, and most of us have taken or are taking “Introduction to Operating Systems” or “Linux Operating Essentials,” which provide a good foundation for the programming skills and knowledge required. The project we will be completing has also been in progress for a few years now, so there is already lot of research and development to use as resources, and the problem space has been thoroughly explored. There are several academic articles written about PUF, justifying its practicality.

Process Details

Our process will consist of establishing functionality at the lowest level of security and later implementing the solution at the highest level. Our initial version will establish success at the user application level, at which point we will advance to explore a solution utilizing higher security. This may or not be the kernel level. We will use the information learned from completing the initial version to assess what the next step should be. The goal is to encrypt a key at the kernel level.

Standards

We are not working in a traditional lab environment for this project and most of our work is completed outside of a collaborative environment. Most of the work is going to be software development and we have decided to follow the agile model for development, so all our standards and protocols, such as sprints, sprint planning and meeting practices are adapted from the agile model. As a result, our standards and practices should all be approved by IEEE and none of them should considered unethical by any organizations.

Test Plan

As code is created, developers must create unit tests simultaneously to test their components and ensure the expected behavior is operational and the number of bugs introduced to the repository is minimal. The longer a bug or malfunction is present, the more difficult the bug is to fix since code may build on top of it. Unit tests will be in a parallel directory with similar path and test file names as the actual code. While no minimum or maximum amount of unit tests are required, all core functionality and common edge cases should be unit tests.

As core components are combined, integration tests should be designed to show the implementation of other targeted components delivers results as expected. These tests may be created by collaborating developers or by the test engineer. These tests should focus on core functionality and interactions the connecting components share. They are often still separate from fully operational actions and focus on specific functionalities.

System tests should demonstrate fully operational tests between multiple components using actual data. These should demonstrate intended functionality, and, for our purposes, some may even be end-to-end tests. These tests will be developed by either collaborating developers in need of data from a live environment or by the test engineer to show operational system behaviors.

Conclusion

Our goal for this project to integrate existing software that details a PUF into an android based OS application. We will do this by research, design, development, testing and conferring with our client. We think this is a worthwhile project because the use of a pressure PUF as a bit locker for a phone has not been done and will add needed security. We hope to complete the whole application in entirety by the end of spring 2019.