zkTAP: A Zero-Knowledge Trustless Authentication Protocol

By

Majdi Hassan

Joseph Kuo

Lilan Yang

Design Document for ECE 445, Senior Design, Fall 2019

TA: Evan Widloski

1 October 2019

Project No. 33

1 Introduction

1.1 Problem and Solution Overview

On January 19, 2010, Muhammad al-Muhbound was relaxing in his hotel room, with a group of four individuals opened his hotel room and assassinated him. It turns out that they had reverse engineered the RFID hotel "master key" and they were able to gain entry to his room. This underscores that many RFID authentication systems do not employ adequate security.

We can reformulate the problem: Alice (the tag) and Bob (the tag reader) want to communicate with each other over an unsecured channel (wireless). Eve is trying to eavesdrop their conversation.

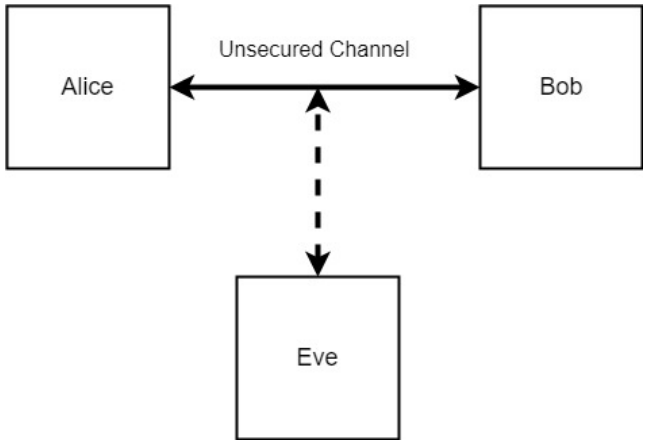


Figure 1 : Illustrative Example 1

Current implementations such as Mifare DESFire employ symmetric key cryptography, to deal with this problem. This require that all parties know a secret key. These tend to be very computationally inexpensive and thus seem to be a good choice for an RFID tag. However, the most obvious drawback is that it is difficult to securely distribute these secret keys and keep these secret keys secret!

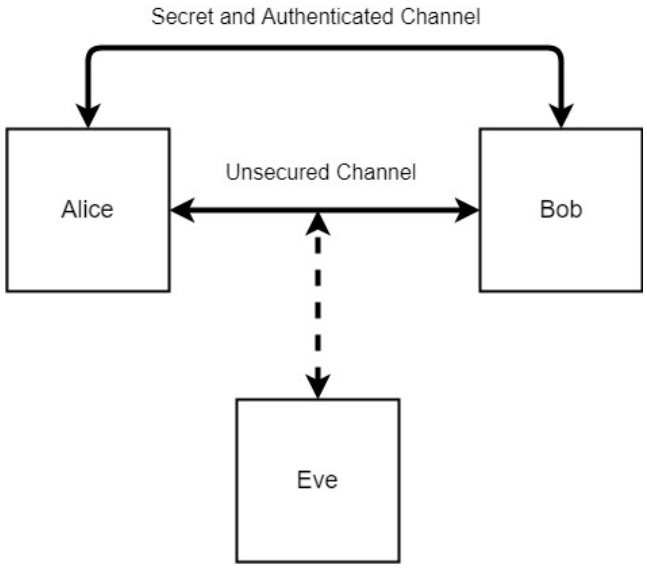


Figure 2 : Illustrative Example 2

Sometimes these keys get leaked, such as in London, where attackers were able to get free rides on their underground system [1].

In contrast to a symmetric-key system, we will employ public-key cryptography, which only require that information being sent is authentic.

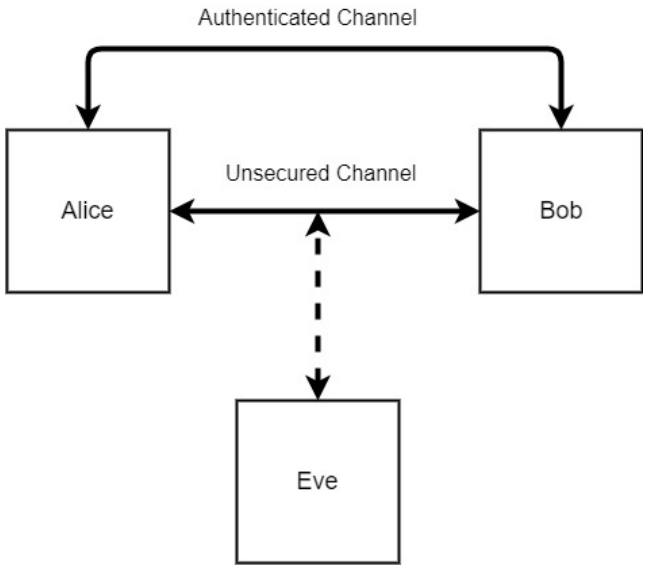


Figure 3 : Illustrative Example 3

Each party selects a public key pair and instead proof-of-knowledge of the private key. Given a public key, it should be computation infeasible to come up with the private key and only the person who know the secret should be able to proof that they know the secret key.

1.2 High-Level Requirements

- The response given by the correct tag must be accepted by the reader with all but negligible probability (where a negligible probability is defined as being somewhere on the magnitude of 2^{-64}.
- A user must be authenticated in under 10 seconds.
- An eavesdropper must not be able to impersonate a user (except with negligible probability).

2 Design

2.1 Block Diagram

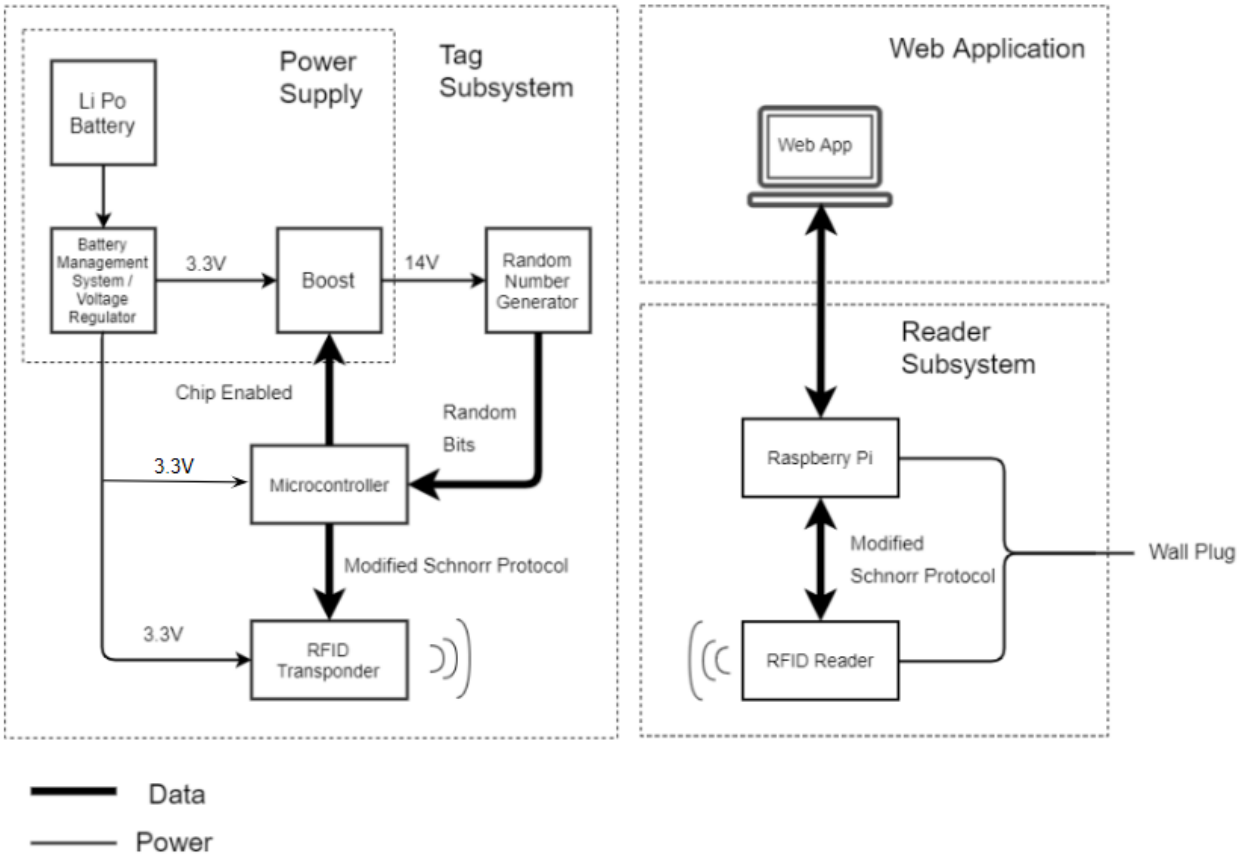


Figure 4 : Block Diagram

We assert that our protocol satisfies our first requirement. A short and straightforward proof is attached in Appendix A.

For requirement 2, we will make the assumption that the processing power of the reader is at least a magnitude greater than the microcontroller. Thus any computational costs of the reader are therefore negligible. Thus we can reduce our problem into: generation of a 255 bit random number, 2 Elliptic Curve Multiplications, 1 addition (mod p). An elliptic curve multiplication on Curve22515 takes $12.34 * 10^6$ clock cycles on MSP430 microcontroller [2]. A MSP430 micrcontroller can run between 8 MHz-24 MHz depending on the generator of the microcontroller [3]. Lampert circuits can be safely sampled at a rate of around $10^4 \sim 10^6$ Hz [4]. Thus we can expect that the random number generation portion takes roughly 0.03 seconds. In addition, we will assume that addition take negligible time. We will assume the worst case scenario which would imply that our protocol would take about 3-4 seconds. We could likely get this down to 1 Elliptic Curve multiplication if we allow for precomputing. There are two ways to further speed this up, we could a 32 microcontroller or use a FPGA + microcontroller combination.

Requirement 3 can only be realized if we have a random number generator on the tag. Consider the dishonest

reader example. The tag's response is $s = d + a + tc$. If a reader were able to predict the value of t, then that reader would be able to extract the secret key a, as s, d, t and c are all known to the verifier. By introducing a random number generator, we can maintain privacy of the tag.

In the case of an honest verifier, an eavesdropper actually can't figure out whether or not authentication was successful, as the value d or d' can only be calculated by the tag and by the reader.

2.2 Overview of Cryptography Protocol

Before introducing our cryptosystem, we would like to introduce the following things. Let p be a prime number, then the finite field F_p represents the set of all integers from 0 to p-1. In addition, we define addition to be integer addition (mod p) and mutliplication to be integer multiplication (mod p) (where multiplication does not contain 0). Let G be a finite, cyclic, and multiplicative group. Also, G contains a generator g, where the elements of it are $g^i : 0 \leq i \leq q - 1$ where q is the order of G. We will be using elliptic curve cryptography, where G represents a point on an elliptic curve over a finite field. We will be using Curve25519, which has the following parameters.

$$p = 2^{55} - 19$$

$$y^2 = x^3 + 486662x^2 + x$$

$$q = 2^{256} - 1$$

G has an order of 2^{252} plus a big number. We will not be including this big curve as it has not be including this big curve curve as it has no elegant form of representation. Note that the following operator [A].x represent the x-coordinate of the curve point.

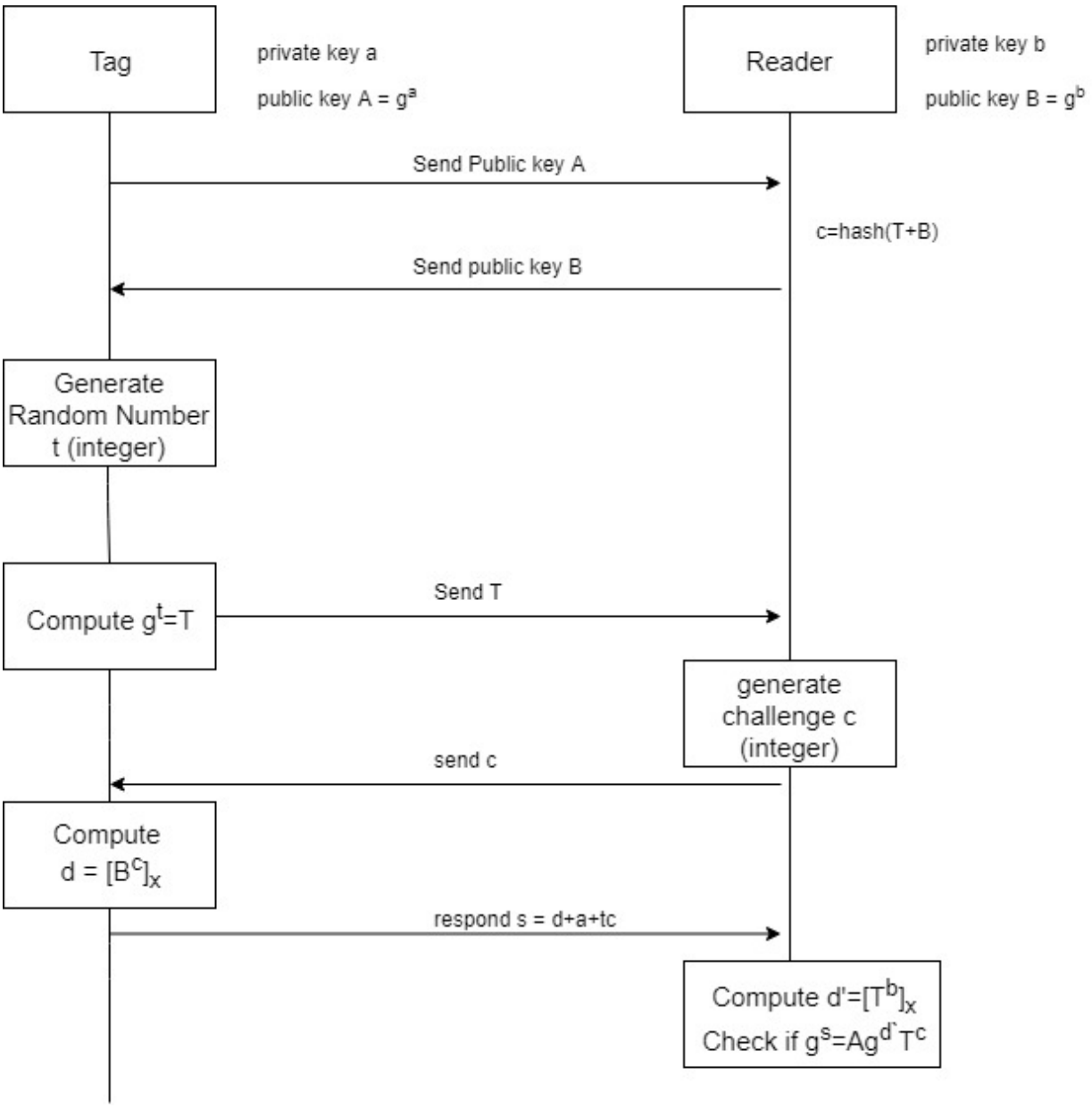


Figure* 5 : *Protocol

This follows several main steps. Our protocol is based off of the Schnorr protocol outline here [5].

1. Exchange public keys.
2. Tag generates t, a blinding factor and sends a commitment to the reader.
3. The reader generates a challenge.
4. Tag responses to the challenge.
5. Reader verifies that the challenge is correct.

The diagram shows the interaction between the RFID reader system and the RFID tag system. The interaction between the two systems is executed in a series of challenges and answering those challenges. The reader will send out a challenge to the tag which only the tag can solve because it holds the key information needed for the challenge. If the tag responds correctly, the reader will authenticate the tag

2.3 Physical Design

2.3.1 Tag System

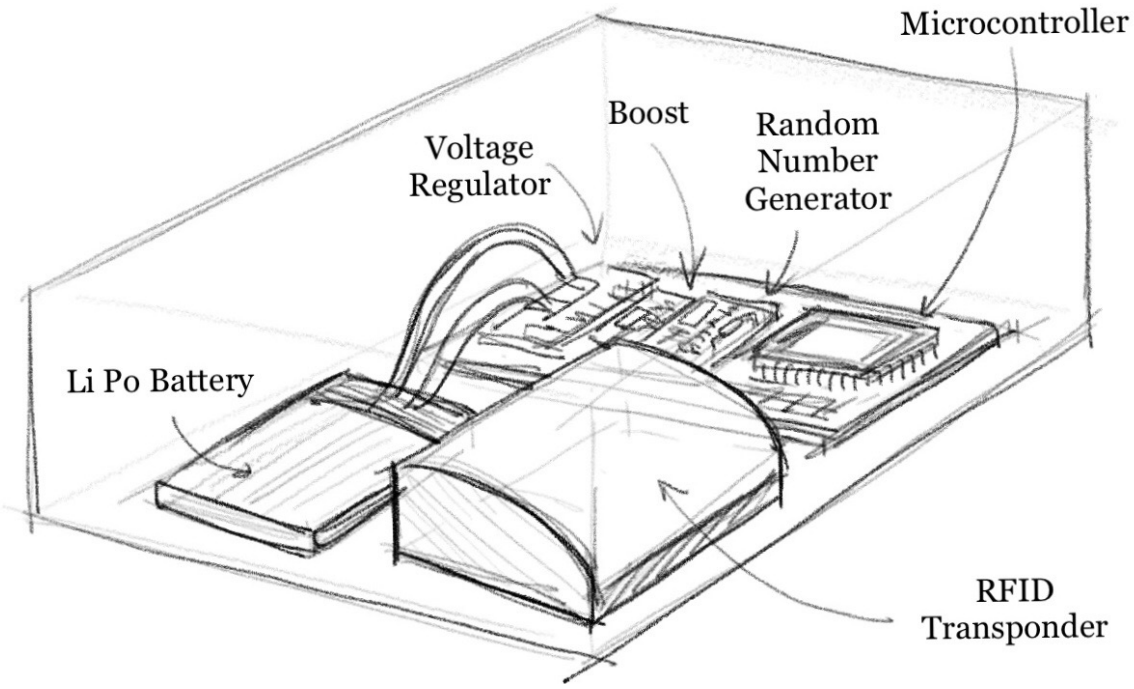


Figure* 6 : *Physical Tag Design

In order to group all the components for the tag system, our plan is to put them inside a box that is large enough to hold Li Po Battery, PCB and RFID Transponder. The desired Li Po Battery would approximately have the size of a card and RFID transponder would be similar to I-PASS transponder mounted on the windshield of the car. As for the PCB, there are microcontroller, hardware random number generator, voltage regulator and boost as parts of power supply. Altogether, we have a tag box that gives us access.

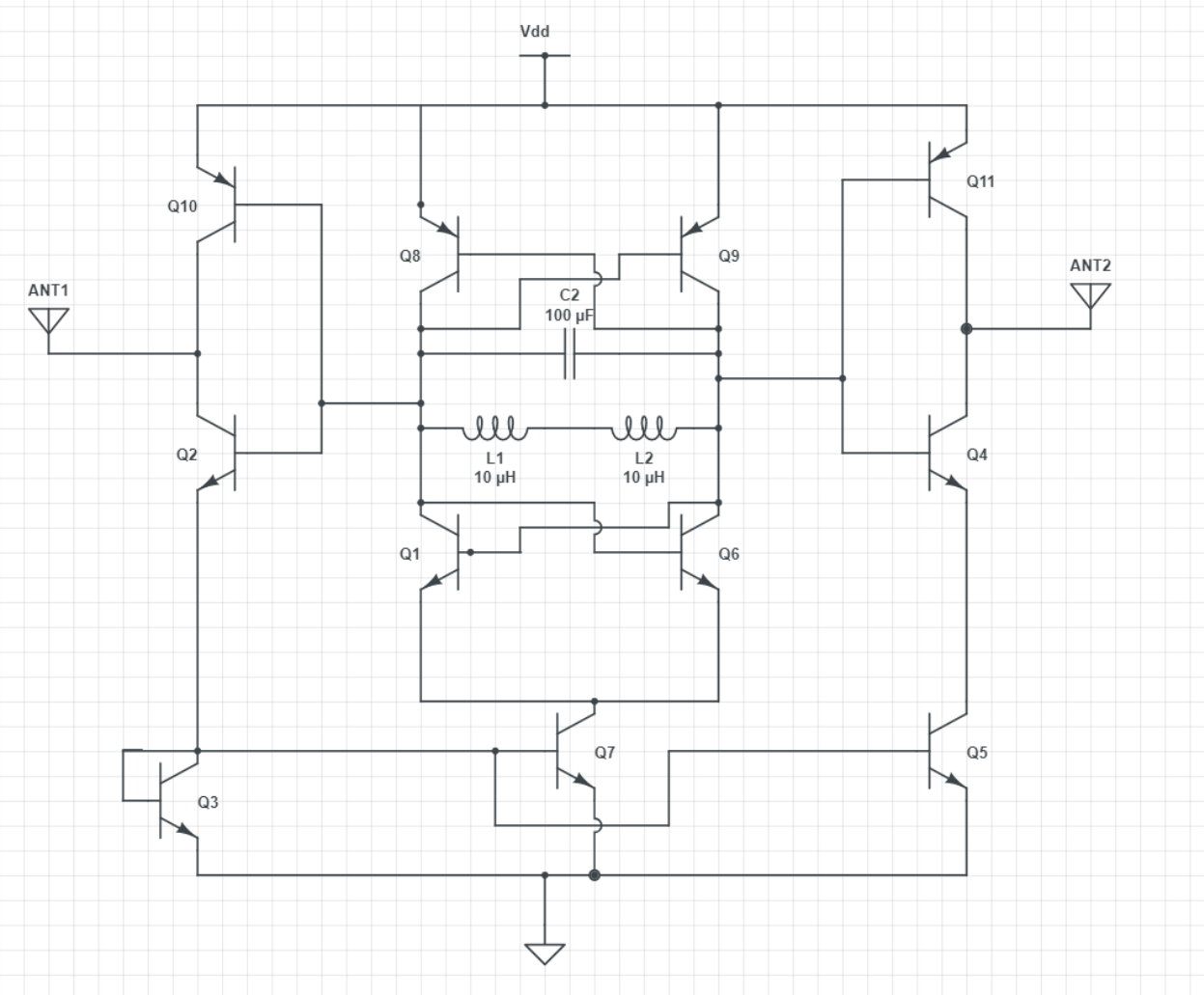


Figure 7 : RFID Transponder Circuit Design[6]

2.3.2 Reader System

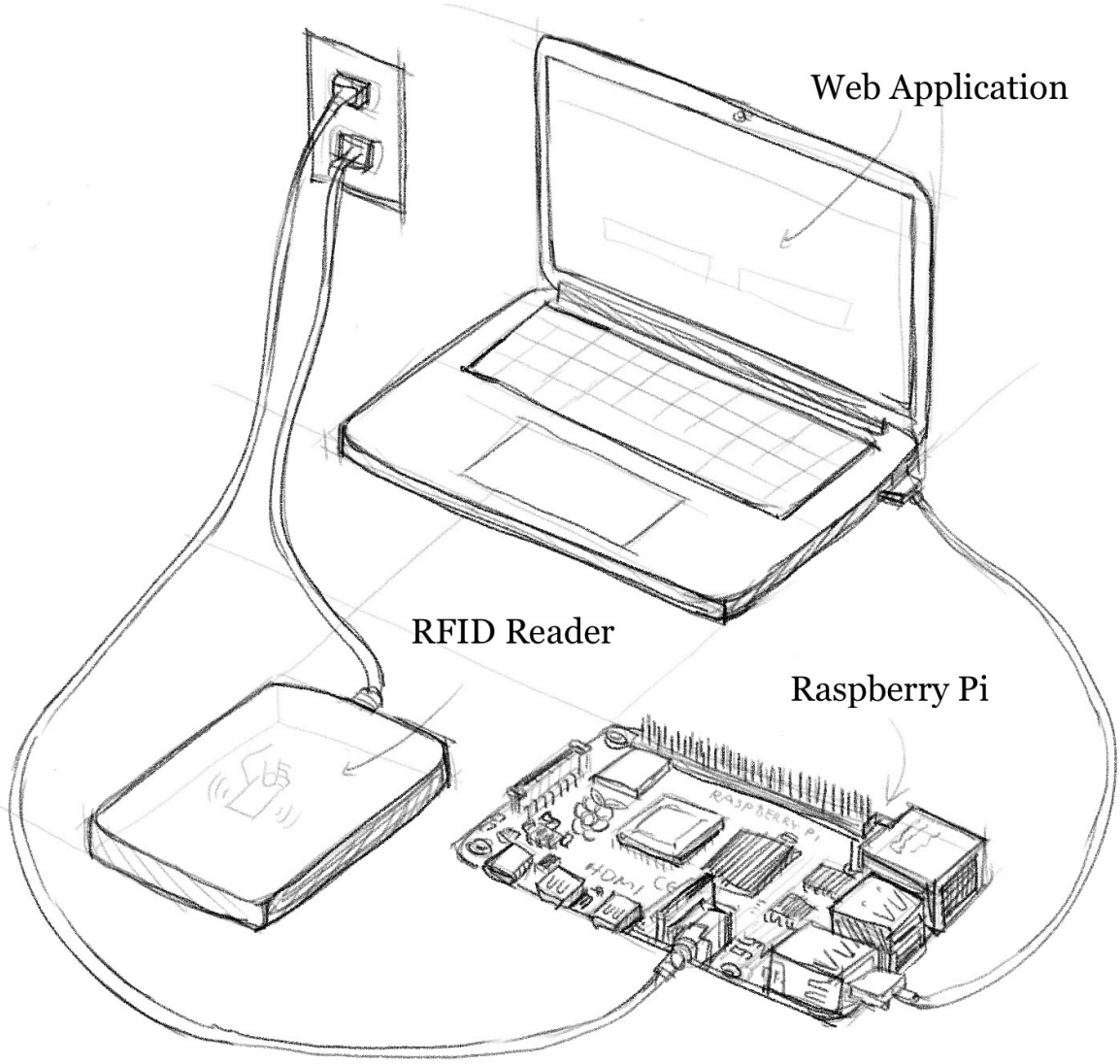


Figure* 8 : *Physical Reader Design

Another critical part of this project is reader's side, which is composed of a commonly found RFID reader, a Rasperry Pi, both powered by the wall, as well as web application that visually present the functionalities. RFID Reader will read our tag and the data will be transmitted to our web application.

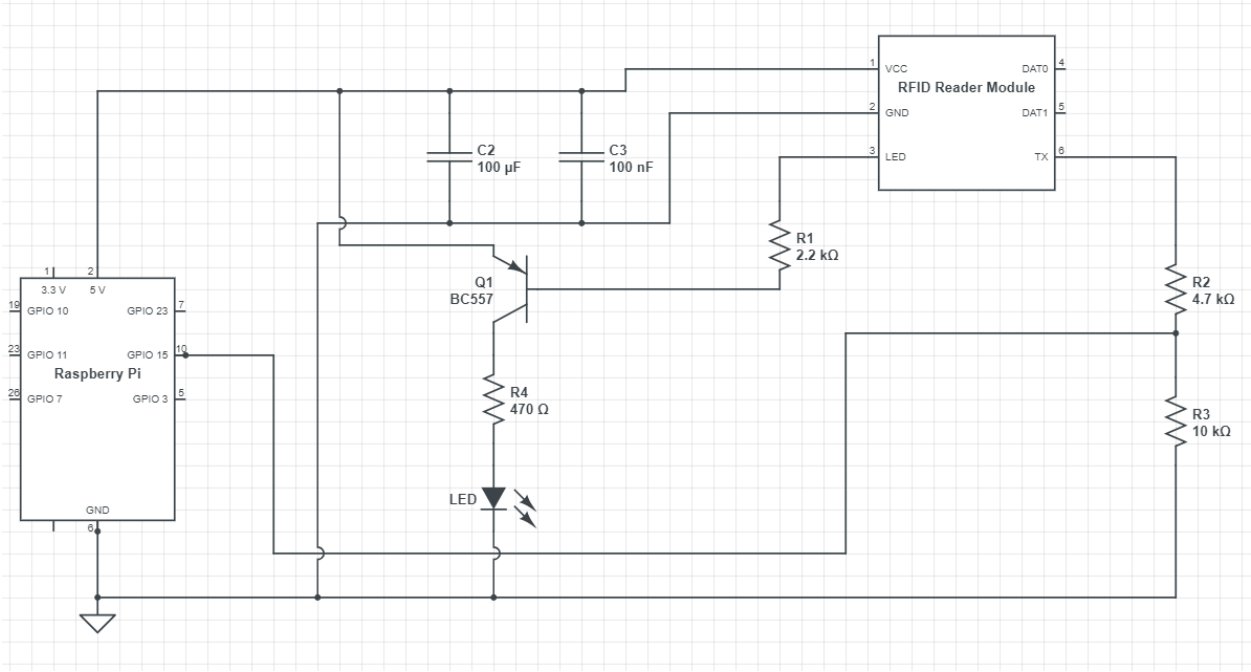
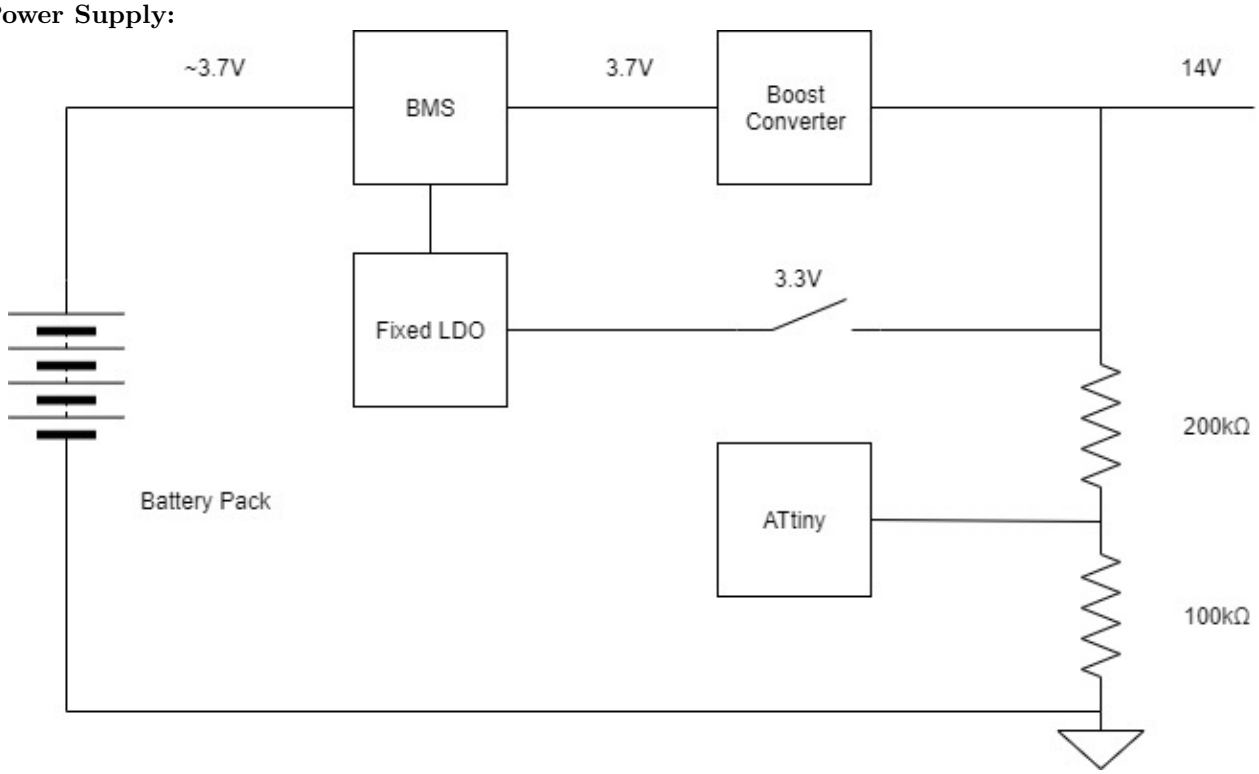


Figure 9 : RFID Reader Circuit Design

2.4 Subsystem Overview

2.4.1 Tag System

Power Supply:



The power supply has to supply around 12.5-14V and 3.3 V. In order to save powerm it will receive a signal

from the main microcontroller on whether or not is should power the random number generator. It will also contain a battery as well

Requirement	Verification
1. When the battery cannot supply sufficient power to power the random number generator, it must shut off.	1. Set chip-enable signal to high and add various resistive loads to the 3.3V and 14V terminals and monitor the voltage across the terminal and make sure it either shuts off when it dips to the 12.5V level

Random Number Generator

Our random number generator provides random numbers to be used to create blinding factors. We intend on using the randomness of breakdown current in a Zener diode. This will fed into a 1 bit Analog to Digital Converter and sampled by our microcontroller.

Requirement	Verification
1. Must produce statistically random numbers 2. sample it at 100kHz	1. Have the random number generator pass the Duke Universities Diehard Test, which is a trusted source to test if random number generators are truly random. 2. start sampling the random number generator at a very slow rate, run statistical test suite and speed up sample rate until we fail

Microcontroller:

Our microcontroller is responsible for sampling bits from the Random Number Generator, performing Elliptic Curve operations, encoding our responses to the reader and sending information to the transponder to be sent to the reader. We plan on using a Reed-Solomon error correcting code as we can share some primatives with our Elliptic Curve portion. We will not be implementing our own Elliptic Curve as such implementations are tricky and subject to numerous side channel attacks.

Requirement	Verification

1. We would like our circuit to consume as little power as possible, especially when the tag is at idle. In order to do this, we need it to shut off non essential components, when the tag isn't in use. As a goal we would like to get power consumption to be under 10 μW	1. To verify this, we will allow the power to idle and measure the power consumption from an external power supply.

RFID Transponder:

Given that the software side of our protocol already satisfies 'the response given by the tag must be accepted by the reader with all but negligible probability,' our transponder must maintain this and transmit our data free of corruption. Our microcontroller will likely provide an error correcting code.

Requirement	Verification
1. We would like our protocol to operate as quickly as possible. As a result, we will require that our RFID tag should transmit data at a rate of at least 10 kb/s.	1. To test this, we will setup a benchmark for the tag to transmit as much data as possible and test the data rates.

2.4.2 Reader System

RFID Reader:

We would like to use a third party RFID Reader to transmit and receive the proofs from the tag. It enables contactless communications with the tag.

Requirement	Verification

1. It reads to the RFID tag. 2. It has to support Error Correction Codes.	1. (a) Upon Tag being scanned, the reader authenticates user and sends information to web app. If web app displays information, then authentication is successful (b) The LED hooked up to the LED output lights up, which indicates that the reader scanned a tag

Raspberry Pi:
It is responsible for interacting with the transmitter and has to host our web application.

2.4.3 Web Application

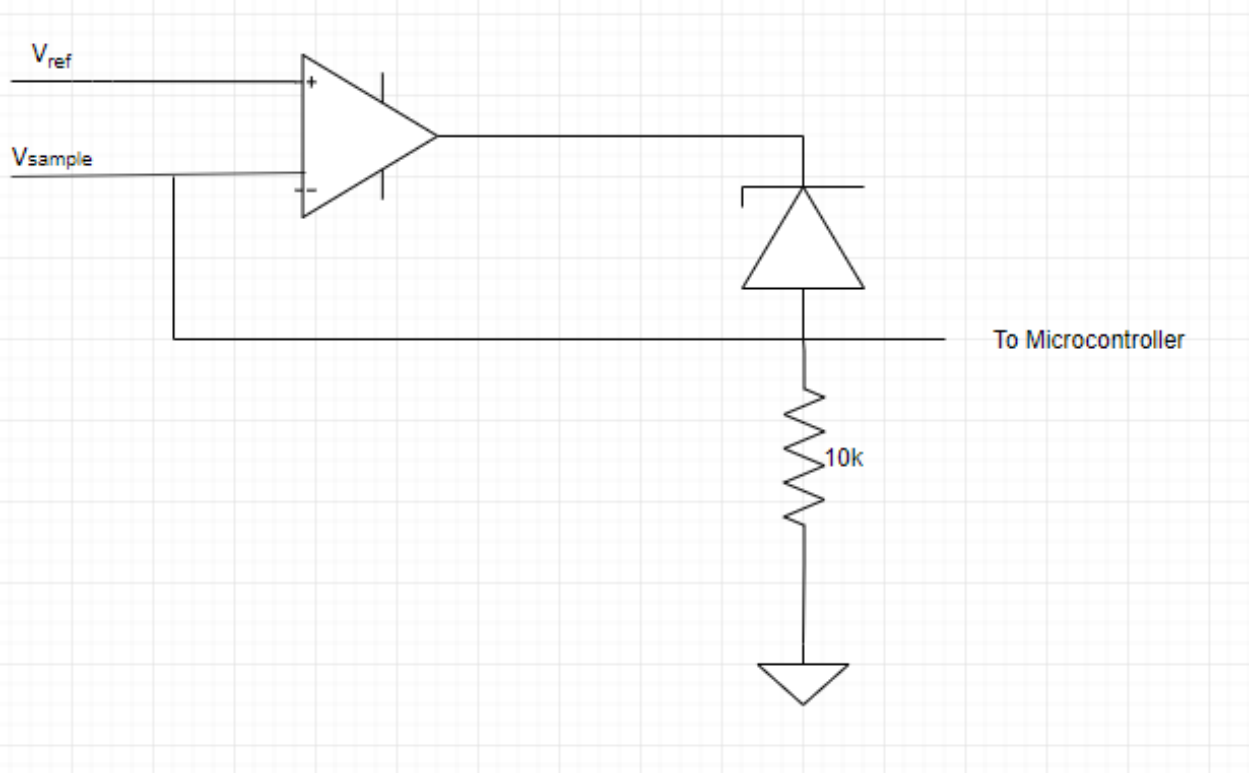
We would also create a simple web application as an user interface. Our goal is to better visualize all the functionalities of the tag.

Requirement	Verification
1. The web application would be able to indicate if a user has been successfully authenticated. 2. Additionally the interface would allow to add new users and revoke access.	1. (a) Have the web application receive data from any sensor and display it until the RFID reader and tag system are setup (b) Have the web app receive data from the RFID reader system 2. (a) Test if Tag gets rejected after revoking access (b) Test if Tag gets accepted after adding new user

3 Tolerence Analysis

Our random number generator poses some risk to our project, namely that it can be difficult to pass 3rd party tests. In addition, it is possible that outside disturbances such as temperature, can affect the quality of the bits that our random number generator outputs. Our protocol requires 255 bit random numbers, however for simplicity, we will round this up to 256 bits. According to the NIST [4], the number of bits that we will need to generate are $\frac{256}{entropy}$. If we take a conservative estimate and assume that each bit produces 0.7 bits of entropy, then we will need to sample 366 bits and then feed it into a cryptographically secure cipher such as salsa20.

Another possible point of contention is that it might seem that our random number generator is sensitive to hardware variances. To analyze our circuit, we will first simplify it as such.



Upon further inspection, we realize that this circuit resembles a closed-loop amplifier. The input impedance on a typical op-amp is on the order of 1MΩ 10TΩ, which is many orders of magnitudes greater than than our $10k\Omega$ resistor. Thus it is a valid assumption to use an ideal op-amp model. With this $V_{ref} = V_{sample}$ and so it isn't hard to see that the average of value of V_{sample} would be V_{ref}. In order to this, we ran LTSpice simulations across a number of possible scenarios.

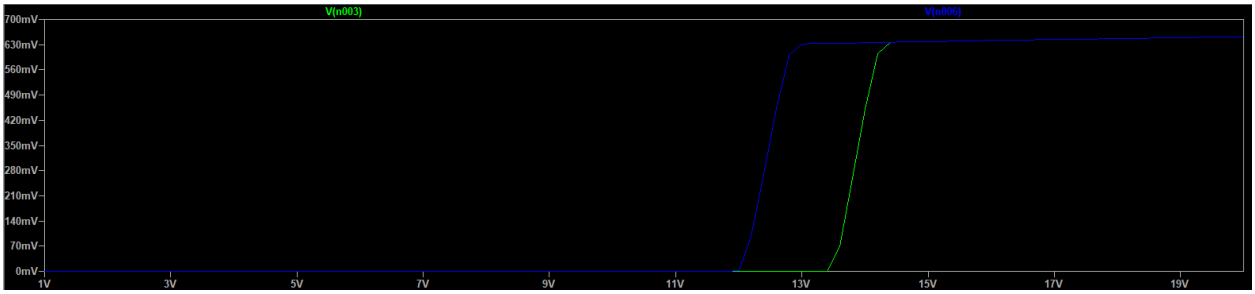


Figure 1: In this scenario, we ran our circuit with two completely different manufactures and two different Avalanche voltages. What we found was that so long as both diodes are in Avalanche breakdown, our output is unbiased. In this image V(n003) (the output from the top diode) has a Avalanche breakdown voltage of around 12V and V(n006) (the output from the bottom diode) has one of 13V.

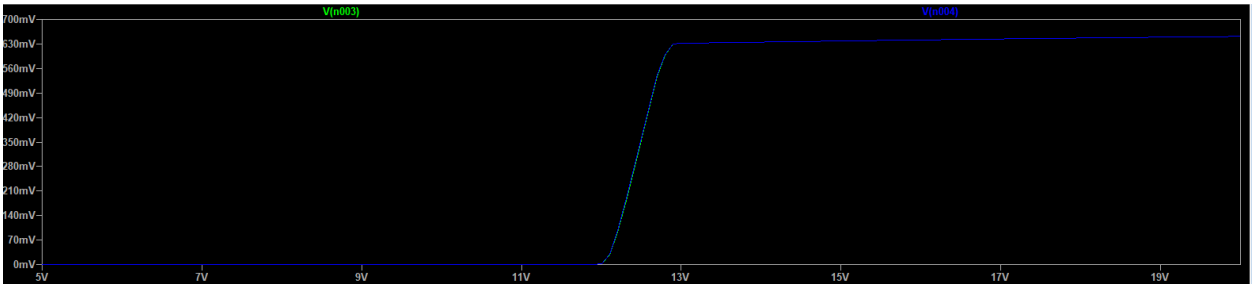


Figure 2: In this scenario, we gave different pull down resistors to each of the diodes. V(n003) has one with $10k\Omega$ and V(n006) has one with $8k\Omega$. This is definitely outside of the manufacturing tolerance for a resistor. In our simulation, we find that has no effect on our output. In all likelihood, having completely different resistances might affect our sampling rate, however LTSpice is not able to simulate the quantum effects of a Zener diode.

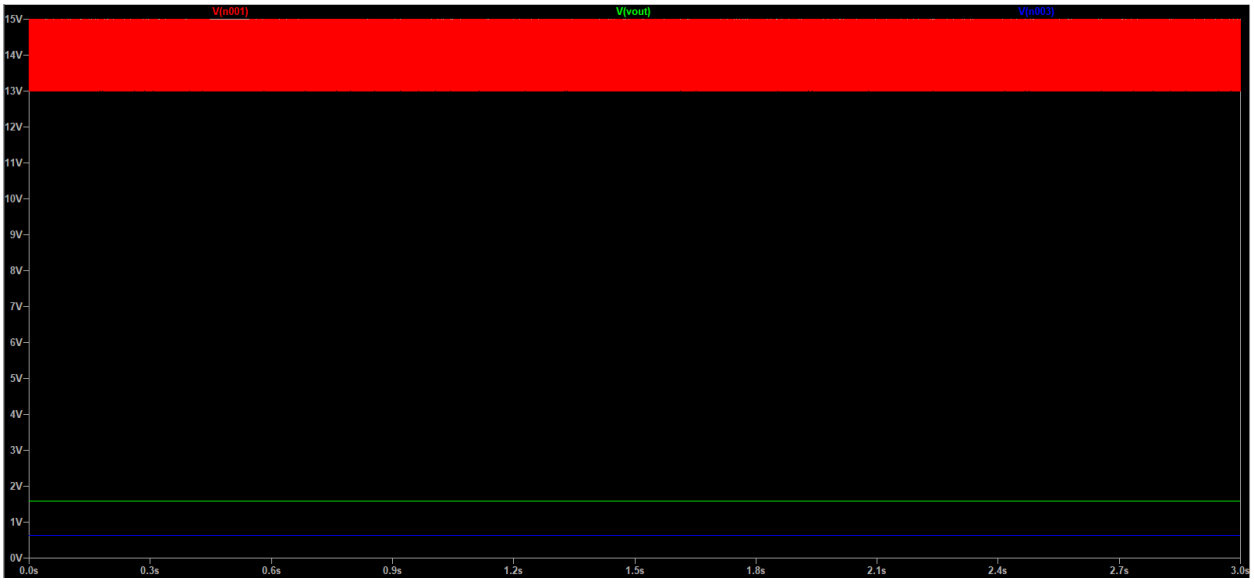


Figure 3: In this test, we simulation an extremely noisy power supply with a mean of 14V and an amplitude of 1V and a frequency of 20kHz. V(n001) is our power supply. V(n002) is the output from the top diode and V(n001) is the reference voltage being fed into our diode circuit. As we see, as long as we are in Avalanche breakdown, there is no effect on our output.

4 Ethics

In regards to safety, the issue that may arise pertains with our use of a lithium battery. If not properly understood when it comes to its implementation, fires and other accidents become likely risks to occur and would pose danger to 'the safety, health, and welfare of the public' [7]. In order to mitigate these risks, proper practices must take place. First, the specifications of the battery must be thoroughly looked over and understood, so when it comes to implementing it in conjunction with the rest of the hardware, there is no overload or improper design such as short circuiting or placing near an area of high temperature that could cause the battery to catch fire and result in a hazardous situation. According to the University of Washington, certain practices such as proper procurement, storage, charging practice, and disposal should be taken into account [8]. For example, acquire the battery from a trusted source, place them away from flammable materials, disconnect batteries that exhibit any unusual behavior such as heating up or releasing some sort of smell, and dispose of batteries safely by taking them to designated facilities.

We plan on using 120∼150 kHz band which is unregulated so we won't have to worry about infringing on any regulations.

Curve22519 is under public domain and so we don't have to worry about patent infringement.

We understand the risks and 'societal implications of conventional and emerging technologies,' which is why we are creating this product. As a result, our top priority is to protect the user's private information [7].

5 Cost

Make sure that any tables of costs are numbered, given titles, and cited directly in the text.

5.1 Parts

Part	Cost
Voltage Regulator	$\$0.33 * (10) = \$3.3$
PCBs (PCBWay)	\$5
Microcontroller (MSP430)	\$2.53
RFID Transponder	$\$0(Chips\ from\ lab)$
RFID Reader (Mifare RC522)	\$6.99
Raspberry Pi 3 Kit	\$49.99
Assorted RLC	\$10.00
Battery Samsung 25R	\$2.99
BMS	\$1.19
T1561041 Boost	\$1.36
ATtiny	\$1.23
Total	\$91.53

5.2 Labor

The average starting salary of a ECE graduate from University of Illinois at Urbana-Champaign is \$96,518 as of 2016-2017. We have three people working on this project and we estimate that we will work about 10

hours per week for 16 weeks. Under these assumptions, the total cost of labor is shown in Equation 3.2.2:

$$\frac{\$96,518}{1yr} * \frac{1yr}{2080\ hrs} = \$46.40/hr$$
$$\frac{\$46.40}{1hr} * 3 * 160\ hrs = \$22,280.15$$

We expect to build 3 prototypes just in case of some failure in one, causing it to be unusable. This will elevate our cost to a total of $ 22,280.15

6 Schedule

Week	Joseph	Lilan	Majdi
9/30	Design Document	Design Document	Design Document
10/7	Order Parts	Begin design of Web App	Tutorials on RF circuits
10/14	Begin PCB	Programming and data transmission research	Check PCB & RF transmission research
10/21	Solder PCB & build power circuit	Have a prototype ready for web app	Build RF tag and validate
10/28	Program ECC onto Microcontroller and validate library	Have web app accept generic input from sensor	Build RF reader and validate
11/4	Integrate Microcontroller plus random number generator with rest of project	Have web app accept transmission from RF reader	Validate Tag and Reader system work and transmit to web app
11/11	Preparing for Mock demo and presentation	Preparing for Mock demo and presentation	Preparing for Mock demo and presentation
11/18	Review of project and handling of defects/improvements	Review of project and handling of defects/improvements	Review of project and handling of defects/improvements
11/25	Testing and Validation	Debugging and Testing Web App	Testing and Validation
12/2	Demo Prep	Demo Prep	Demo Prep
12/9	Final Paper	Final Paper	Final Paper

References

[1] N. Courtois, "Card-only attacks on mifare classic," University College London.

[2] Z. Liu, J. Großschädl, L. Li, and Q. Xu, "Energy-efficient elliptic curve cryptography for msp430-based wireless sensor nodes," in *Information Security and Privacy*, J. K. Liu and R. Steinfeld, Eds. Cham: Springer International Publishing, 2016, pp. 94–112.

[3] "Msp430f552x, msp430f551x mixed-signal microcontrollers," Texas Instruments, Dallas, TX, 2008.

[4] B. Lampert, R. S. Wahby, S. Leonard, and P. Levis, "Robust, low-cost, auditable random number generation for embedded system security," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, ser. SenSys '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2994551.2994568 pp. 16–27.

[5] J. Hermans, A. Pashalidis, F. Vercauteren, and B. Preneel, "A new rfid privacy model," in *ESORICS*, 2011.

[6] L. Fragomeni, F. Zito, F. G. D. Corte, F. Aquilino, and M. Merenda, "Cmos fully integrated 2.5ghz active rfid tag with on-chip antenna," *Melecon 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference*, pp. 922–926, 2010.

[7] "IEEE IEEE Code of Ethics," 2019. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[8] "Stay safe when using lithium batteries," April 2018. [Online]. Available: https://www.ehs.washington.edu/about/latest-news/stay-safe-when-using-lithium-batteries

Appendix A Proof of Requirement 1

Requirement 1 can be rewritten as, $g^s = Ag^{d'}T^c$. Now we need to show that this shows true for all users.

Lemma A.1. *The value d and d' are equivalent so long so as either the value t (blinding factor generated by the tag) or the value y (secret key of the reader) is known.*

Proof. Recall that $d = [B^t]_x$ and $d' = [T^b]_x$, where $[]_x$ is the x coordinate of the elliptic curve point represented by the value in brackets. Substituting values in, we get $d = [g^{bt}]_x$ and $d' = [g^{tb}]_x$ and thus we see that the values are equivalent. As we can see knowledge of either t or b is required to construct d. □

Proof. Let s represent the responsive given by the tag. We need to show $g^s = Ag^{d'}T^c$ hold true.

$$
\begin{aligned}
g^s &= Ag^{d'}T^c \\
&= (g^a)g^{d'}g^{tc} && \text{(substituting variables)} \\
&= (g^a)g^{d}g^{tc} && \text{(from lemma a.1)} \\
&= g^{a+d+tc} && \text{(the tag's response is s = a + d + t c)} \\
&= g^s
\end{aligned}
$$

□