zkTAP: A Zero-Knowledge Trustless Authentication Protocol

By

Majdi Hassan

Joseph Kuo

Lilan Yang

Final Report for ECE 445, Senior Design, Fall 2019

TA: Evan Widloski

11 December 2019

Project No. 33

Abstract

We present an implementation of a secure RFID tag-reader system utilizing public-key cryptography. In brief, our tag is resistant against 'man-in-the-middle' attacks, replay attacks, and is secure even with a dishonest reader. Because all zero-knowledge proofs require a random number generator, we included a hardware random number generator, which we verified using the NIST SP 800-90B Entropy Estimation Suite. From this we measured that our random number generator has a minimum entropy of 0.88 bits of entropy per bit sampled. Our tag's unit cost is $14.352 and our reader's is $17.188. A Python-backed web application using Django framework is also included as visual aid to simulate the door accessing scenario.

Contents

1 Introduction

1.1 Problem and Solution Overview

On January 19, 2010, Muhammad al-Muhbound was relaxing in his hotel room, when 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 formulate the problem to better understand what happened that night: Alice (the tag) and Bob (the tag reader) want to communicate with each other over an unsecured channel (wireless). Eve (the assassins) is trying to eavesdrop their conversation.

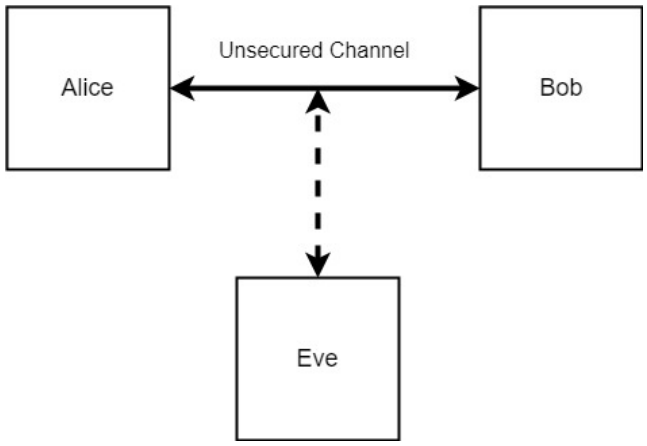


***Figure* 1 : *Illustrative Example* 1**

Unfortunately for Alice, there is nothing actually stopping Eve from eavesdropping.

Current implementations such as Mifare DESFire employ symmetric key cryptography, to deal with this problem. This require that all parties know a secret key. In order to send a message, would require both Alice and Bob to know a shared secret key. Alice would first encrypt her message using the secret key and then transmit it over the unsecured channel. In order for Bob to read her message, he needs to decrypt the encrypted messaged from Alice. While Eve has a copy of the encrypted message, she cannot read the contents of the message without knowledge of the secret key. The advantage is that these tend to be computationally inexpensive, and thus seem to be a good choice for a RFID tag. However, the most obvious drawback is that it is difficult to securely distribute these secret keys! This is since after one member releases the secret key out, now everyone can eavesdrop.

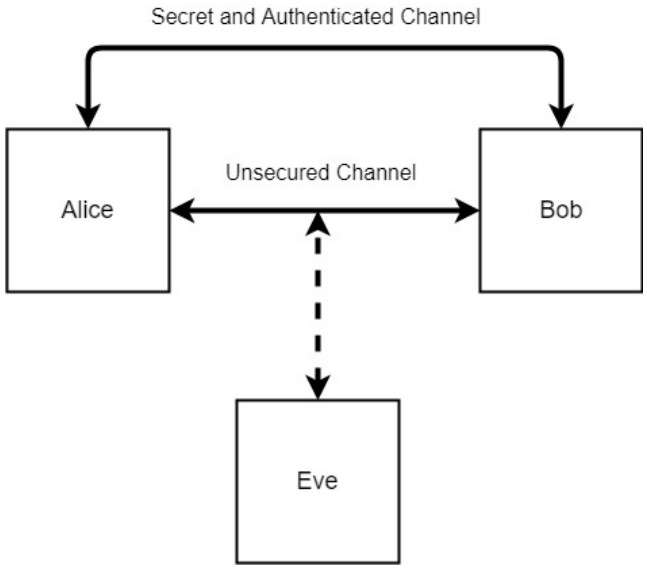


Figure 2 : Illustrative Example 2

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

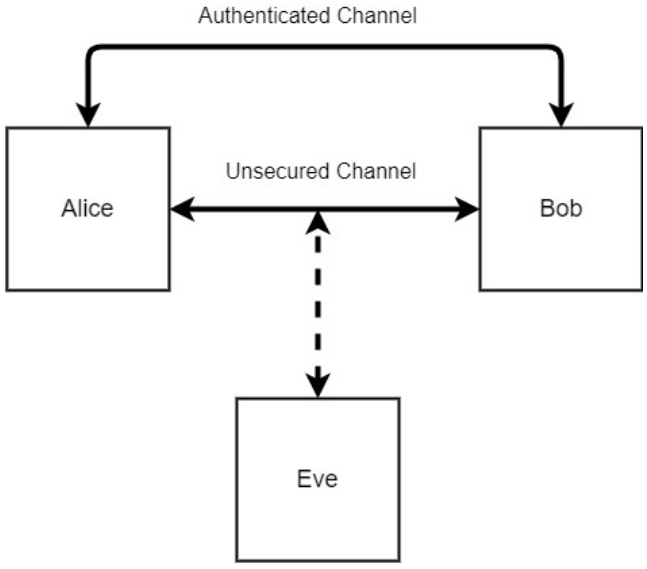


Figure 3 : Illustrative Example 3

In contrast to symmetric-key cryptography, we will be employing a form of public-key cryptography (PKC) instead. One possible application goes as follows, every member now generates a private-public key pair. The public key is freely disseminated. In order for Alice and Bob to communicate, Alice uses her private key and Bob's public key to encrypt a message. Now in order for Bob to decrypt the message, Bob needs to use his private key, along with Alice's public key in order to decrypt the message. Eve cannot read the message as it requires knowledge of either Alice or Bob's private key.

This has several benefits when compared to the symmetric-key cryptography system. In a symmetric-key cryptography system, it becomes more difficult to securely distribute the secret key as the number of users increase. In contrast, PKC only requires that the recipient's public key is authentic. Now suppose that one member of the party decided to reveal their private key. Only communicated to and from that user is comprised. On the other hand, if that same user were to reveal the secret key in a symmetric-key cryptography system, everyone's communication would now be comprised.

We propose to create a RFID Tag/Reader System that uses public-key cryptography. The tag will be an active RFID tag which will use a MSP430 to perform the public-key cryptography and it will feature a hardware random number generator.

2 Design

2.1 Block Diagram

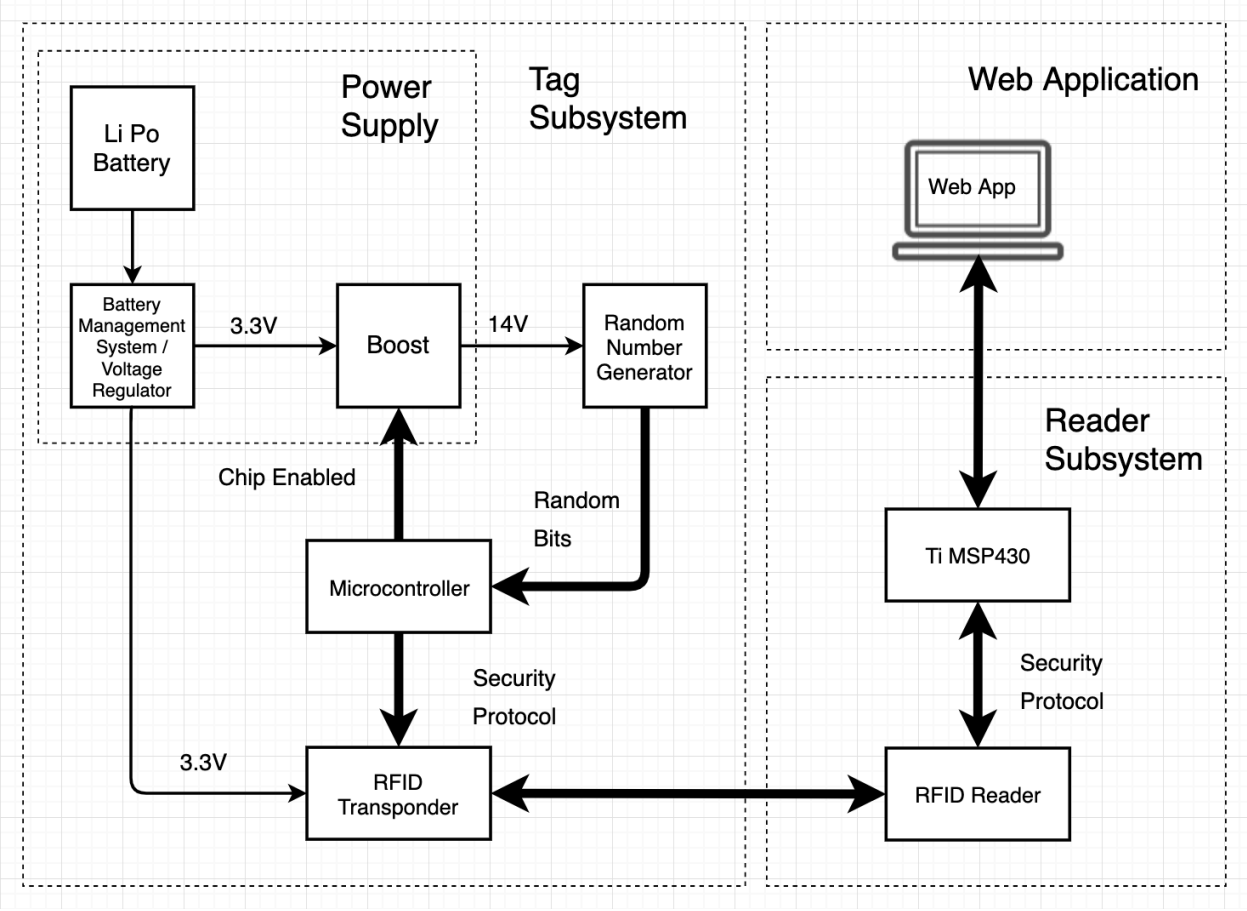


Figure 4 : Block Diagram

This project has been divided into three major subsystems: tag subsystem, reader subsystem and the web application. The tag subsystem also include the power supply that contains battery, BMS and a boost that turn 3.3V into 14V that feeds the hardware number generator. And the hardware number generator will

provide a random bit to microcontroller. Then the tag and reader subsystem communicate through RFID Transponder and Reader. The RFID Reader connecting to MSP430, which was previously a Raspberry Pi, will transmit data with the web application to verify information.

As results, the response given by the correct tag is accepted by the reader with all but negligible probability. A user can be authenticated in around 10 seconds, and the eavesdropper cannot impersonate a user.

Tag Subsystem

For the RFID transponder, we decided to approach using an MSP430 and NRF24L01 chip. Other options would include Arduino or Raspberry Pi for the processing portion of the design and wifi chips for communication. The reason we chose the parts that we did for the RFID transponder is due to a number of reasons. First, we had the MSP430 already in hand, so we did not have to purchase that part. Even though we also have the Raspberry Pi, we reserved it for the RFID reader. Second, the NRF24L01 chip allows for communication within the vicinity whereas with a wifi chip, one could authenticate from anywhere as long as they have access to a wireless network. We did not deem that acceptable for the project we were demoing since we want the RFID tag and reader to communicate in the presence of one another. Also, there is reliable support and tutorials on the internet when working with NRF24L01 chips. Moreover, we decided to power the RFID tag actively, meaning it is battery powered. The other approach is to have the tag be passively powered, meaning it would power on when in range of the reader's energy field. This would allow the tag is be less bulky and not be limited in its lifespan due to the battery; however, the decision to not power the tag passively stemmed from the focus of our project being the random number generator and did not want to promise a feature that we did not know if we would have enough time to implement. However, for future expansions of the project, this is definitely the route to pursue.

Reader Subsystem

For the RFID reader, we decided to utilize a Raspberry Pi and NRF24L01 chip. We chose the Raspberry Pi since we had that on hand and we were going to host the web application on the raspberry pi. In our final cut, we switched out the raspberry Pi for an MSP430 since the script running on the raspberry pi to handle the reader task would constantly freeze and crash, which caused a drop in communication between the reader and tag. On the other hand, this issue was not present when we switched to the MSP430 and we had no trouble communicating between the reader and the tag. We chose the NRF24L01 chip since we wanted to have matching hardware with the communication chip on the tag. The current setup has the reader connected over the serial port through USB to a computer in order to grant or deny access based on tag permissions. Ideally, we would use a wifi chip to allow the reader to send this information to the web application wirelessly to a computer that is hosting it centrally since it is not viable to have a computer attached to every reader.

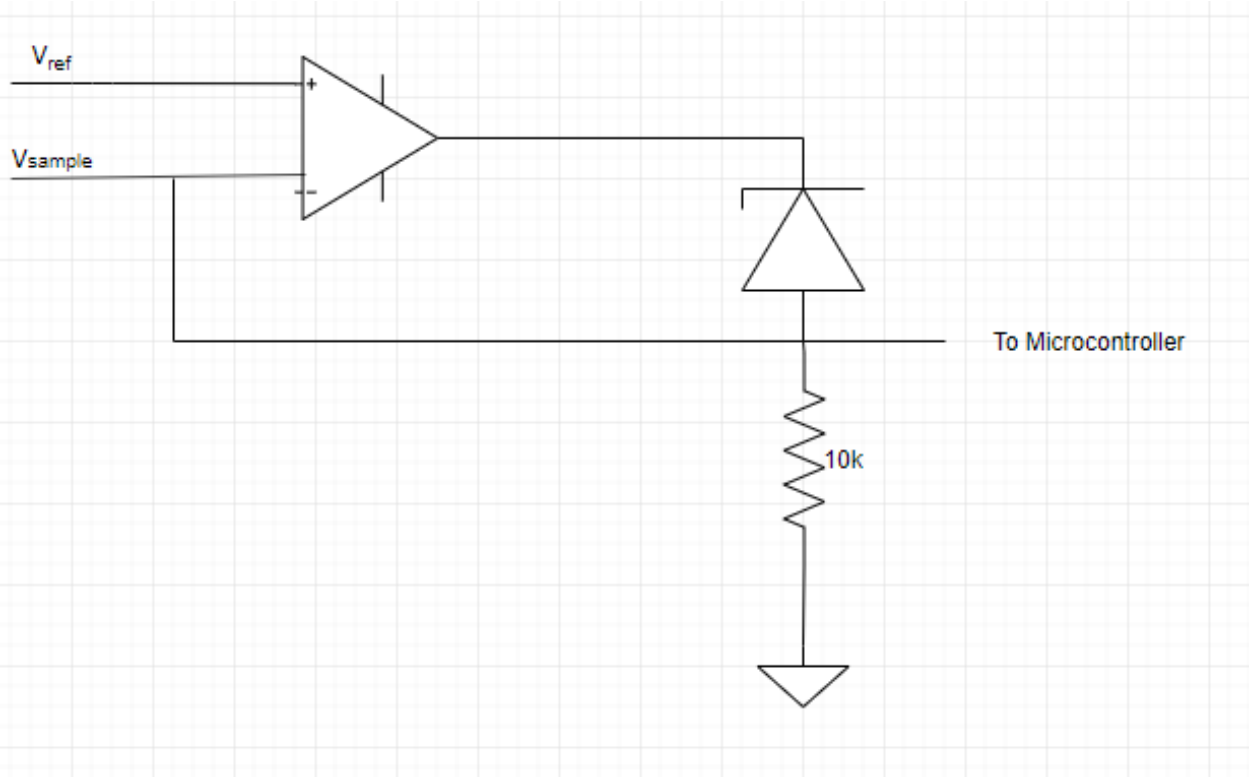
Web Application

There are plenty of alternatives to Django for Python based web development such as Flask and web2py. And the reason that we started off with Django is that our original plan is to program everything in Python, and Django is a quick solution for web development with back-end database support and a deadline considered.

2.2 Design Details

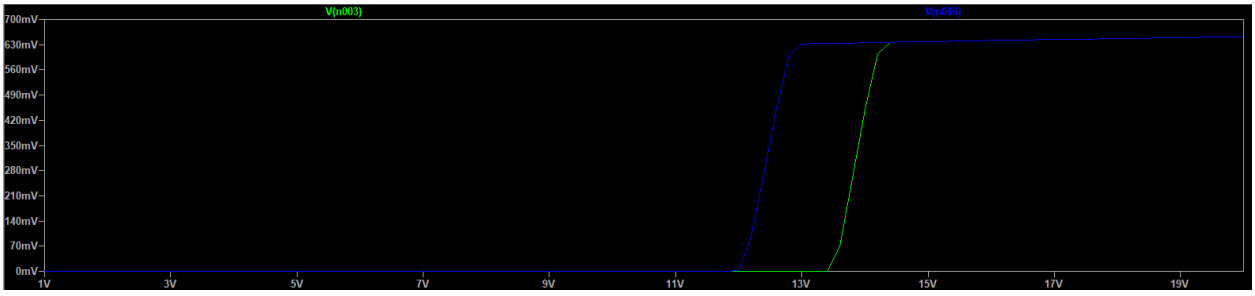
2.2.1 Random Number Generator

In order to design our random number generator, we first start off with the Zener diode noise circuit described in here [2]. Lots of methods exist such as nuclear decay and clock drift, however we chose to use Zener diodes due its low cost.



The addition of the op-amp allows us to bias the output of the Zener diode to be the value of V_{ref}. This is helpful because operating voltages across diodes can vary.

The avalanche effect of a Zener diode occurs at Voltages greater than 6. In addition, when you increase the voltage, you also increase the amount of noise generated. Because of this, V_{ref} will most likely have to come from a boost converter. Most boost converters have some voltage ripple and so the output from this circuit might be affected by ripples in V_{ref}. In order to mitigate this, we followed the example provided in [2] and connected two of those units to same V_{ref} and then fed the two inputs into a comparator.



In this simulation two different Zener diodes were used, $V(n003)$ has a breakdown voltage of around $14V$

and $V(n006)$ has a breakdown voltage of around $12V$. From these results, we can conclude that the output from the noise circuit will be identical so long as both Zener diodes are undergoing avalanche breakdown.

Most manufactures don't provide data on noise characteristics on their Zener diodes unless it is advertised as 'low-noise.' As a result, we picked ones with a Avalanche breakdown voltage of 12 Volts to ensure that adequate noise was available.

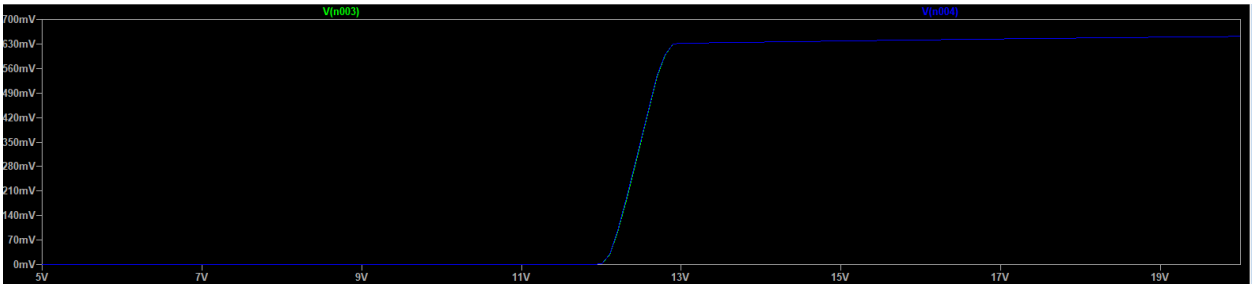


Figure 1: In this scenario, we gave different pull down resistors to each of the diodes. V(n003) has one with $10k\Omega$ and V(n006) has one with $8k\Omega$. This is definitely outside of the manufacturing tolerance for a resistor. In our simulation, we find that has no effect on our output. In all likelihood, having completely different resistances might affect our sampling rate, however LTSpice is not able to simulate the quantum effects of a Zener diode.

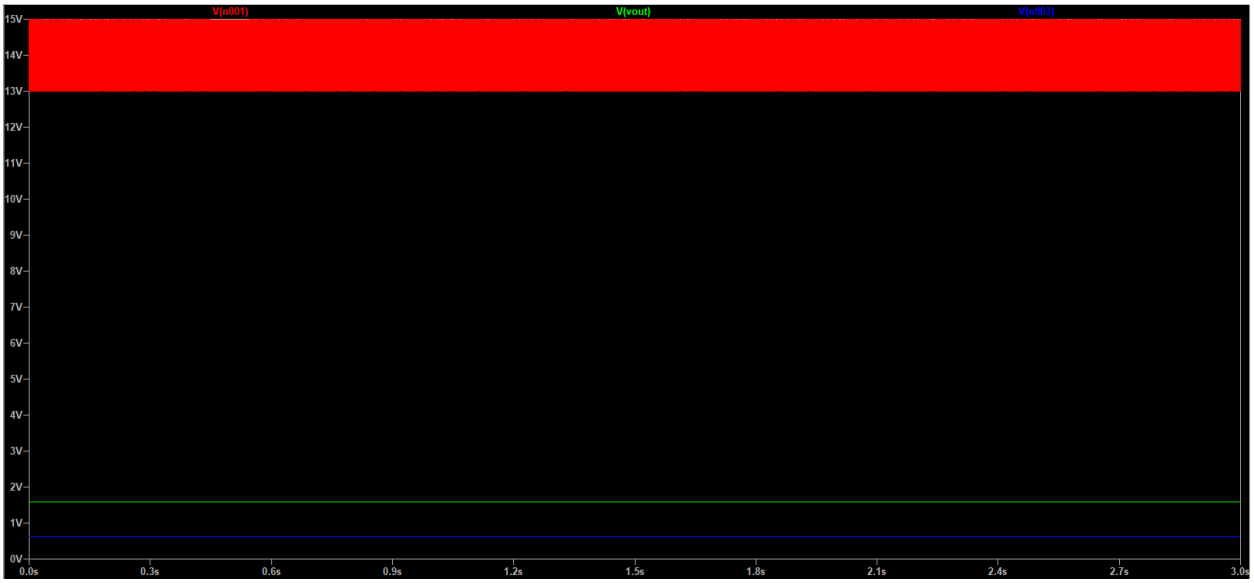


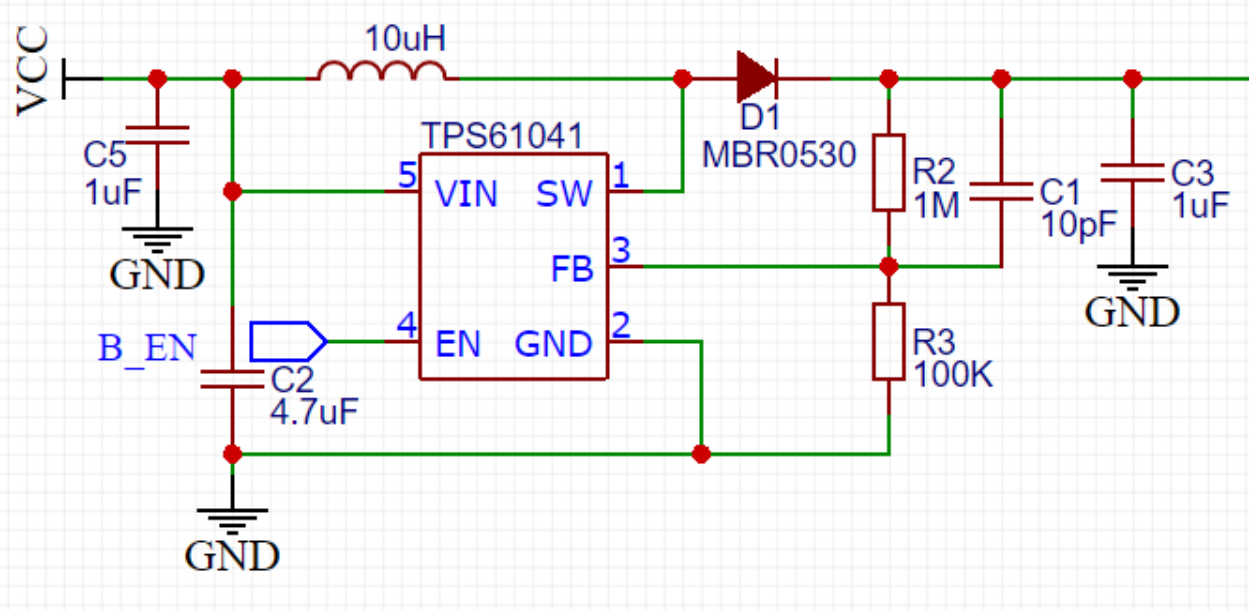
Figure 2: In this test, we simulation an extremely noisy power supply with a mean of 14V and an amplitude of 1V and a frequency of 20kHz. V(n001) is our power supply. V(n002) is the output from the top diode and V(n001) is the reference voltage being fed into our diode circuit. As we see, as long as we are in avalanche breakdown, there is no effect on our output.

2.2.2 Power System

From the discussion on the Random Number Generator, we picked a Zener diode with an avalanche breakdown voltage of 12 Volts and from our simulation results, we found that it is extremely important to make sure that the diodes are in avalanche breakdown. From the datasheet, we gathered that the maximum avalanche breakdown voltage for a 12V Zener diode is 12.5 Volts, therefore we selected that our boost

converter should have a target Voltage of around 14V.

We decided to use the TPS61041 boost converter as it has a maximum output of 28V and was the best selling boost converter from our supplier.



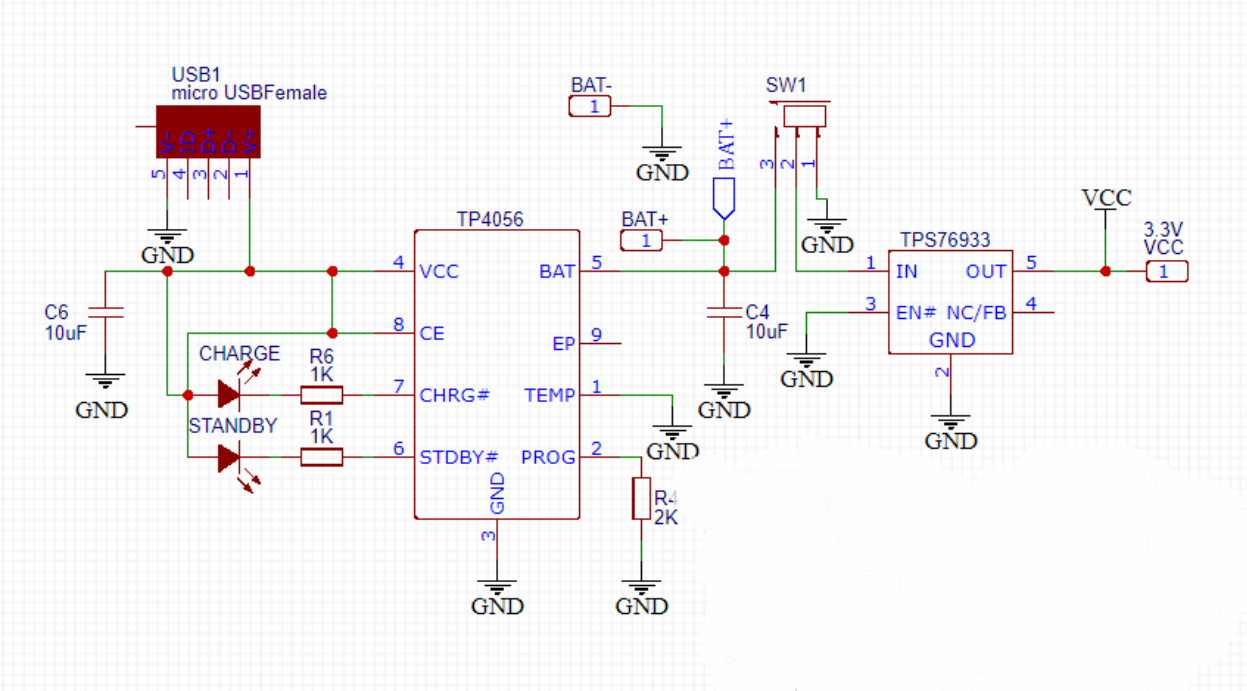
The inductance value and the value for C1 were recommended values from the datasheet. In order to compute values R2 and R3 we used the following formula from the datasheet [3].

$$V_{out} = 1.233V \times (1 + \frac{R1}{R2})$$

By selected R1 to be 1MΩ and R2 to be 100kΩ, we get an output of 13.57V. Although this is less than 14V, it still leaves us plenty of room for the diodes to be in avalanche breakdown, thus it satisfies our design goals.

The input B_{EN} is comes from the microcontroller and it controls whether or not the boost converter should be turned on or not.

In order to make our tag portable, we used a 18650 Lithium Ion battery, as such we need to use a BMS IC to prevent the battery from overcharging and to provide a suitable charging voltage for our battery. To provide power for our microcontroller, we used a low dropout linear regulator.



2.2.3 Microcontroller

In order to meet our power requirements, we included two power states, one when the tag is in range of the reader and one when it is idling. Specifically we switched the tag's power state to 'LPM2' which reduced the micrcontroller's power usage to 23 μW. While we are in this phase, we also turn off the boost converter. In the phase the microcontroller is running at 32 kHz. In order to get out of this phase, the NRF24L01 will send a interrupt whenever it sees something interesting. Once this happens the tag will use the 16 MHz again.

2.3 Overview of Cryptography Protocol

Lots of different public key encryption algorithms exist. The two most popular ones are RSA and Elliptic Curves. RSA's security relies on the assumption that it is difficult to factor a large semiprime integer, thus it roughly scales based off of the smaller of the two numbers used to construct the semiprime integer. On the other hand the security of elliptic curve cryptography relies upon the discrete logarithm assumption. The current best method to solve the discrete logarithm problem, is Pollard's Rho algorithm, which isn't significantly better than brute forcing values. Thus we can expect that the key sizes to achieve a relatively similar amount security should be lower for Elliptic Curve Cryptography and indeed this is case, with an Elliptic Curve key size of 256 bits being roughly equivalent to an RSA keysize of 3072 bits [4].

Choosing parameters for an elliptic curve that are both secure and computationally easy to compute is difficult. To make matters more difficult, we chose to implement this on a microcontroller. Due to this, we were left with two possible implementations, FourQ and Curve25519. As Curve25519 has been much more thoroughly vetted, we chose to use Curve25519. As we continued along in our project, we found that this elliptic curve was defined over a non-prime field (with a cofactor of 8). Thus in order to prevent the small subgroup attack, many elliptic curve operations were not provided in the reference implementation.

In order to get around this, we employed the Ristretto point compression method, which in essence allows us to define points on a prime order group [5].

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.

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

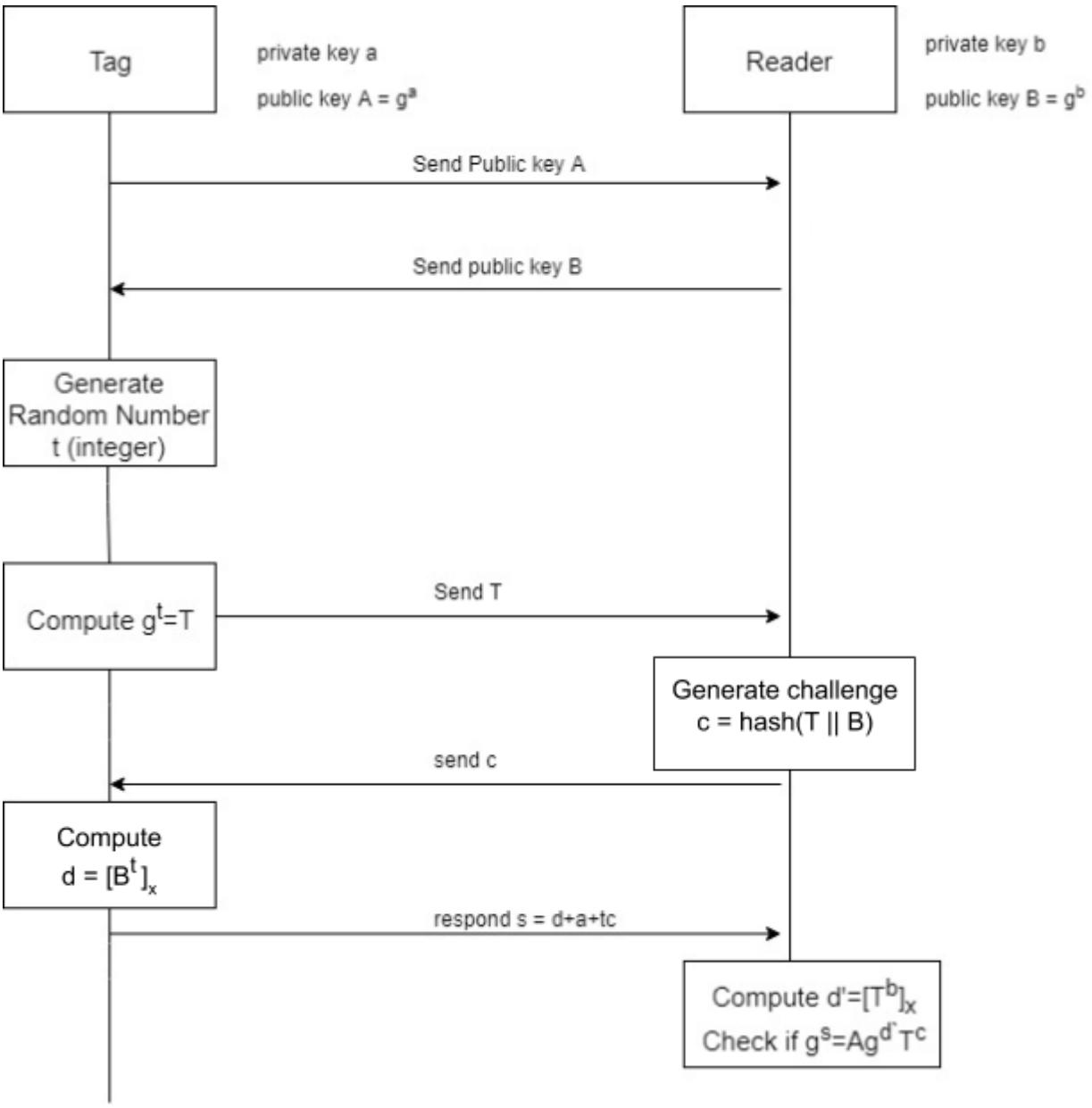
$$y^2 = x^3 + 486662x^2 + x$$

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

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.

Many different zero-knowledge 'Proof-of-Identity,' with the most basic one being the Schnorr protocol [6]. However the Schnorr protocol is not 'honest-verifier' zero-knowledge. Because we plan this to be used on an RFID tag system, it is reasonable to assume that a verifier (in this case a tag) might want to try to steal your public key. As such we decided to use a modification proposed by Peeters [**?**], which include a Diffie-Hellman like key-exchange portion in order to maintain a security under the dishonest verifier case.

We made a small modification to the challenge generation phase. The challenge is suppose to be generated by a 'random-oracle' which is a black box that responds to a unique query with a random response and replies with the same response given the same query. Thus we substituted this with a hash function that takes both the commitment to the blinding factor, T and a random number c (who is sent by the reader). This way it only requires either the reader or the tag to be honest (The case where both the tag and reader are dishonest is trivial).



This follows several main steps. Our protocol is based off of the Schnorr protocol outline here [7].

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 responds 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.

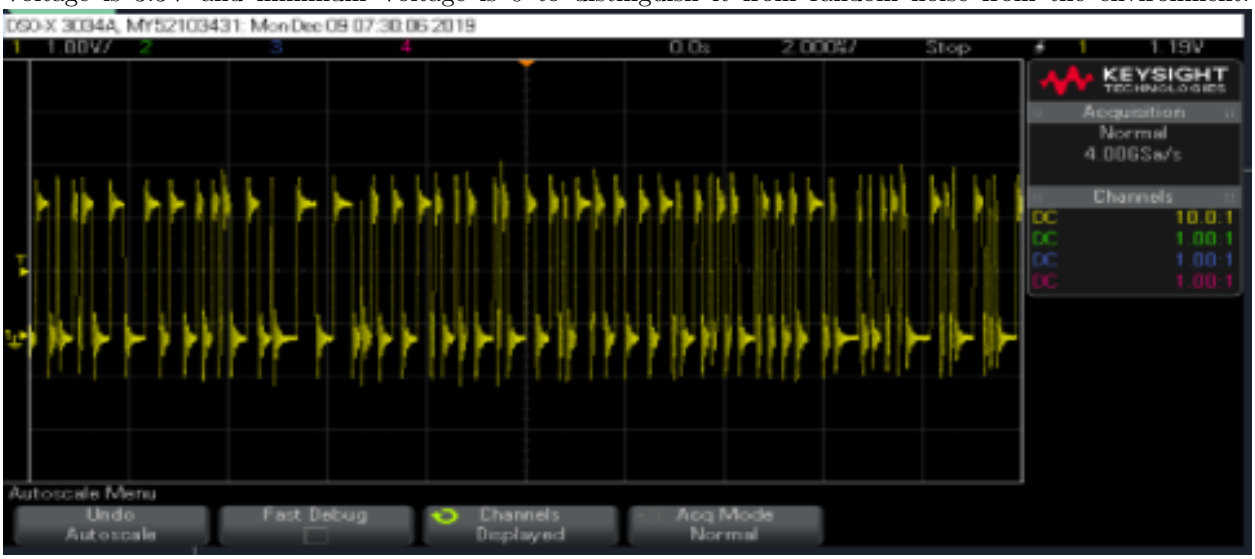
2.4 Verification

2.4.1 Hardware

Because there are many parts in our project and some parts (especially those that come in a QFN package) are difficult to solder, we have developed the following testing procedures.

1. To check USB port, we plug it in and check to see that either the 'Standby' or 'Charge' LED flash.
2. To verify the BMS IC, we measure the voltage across the BAT+ and BAT- terminals and make sure it reads 4.2V. In addition, we short the two terminals to test the short circuit protection of the IC and ensures that the chip survives the process.
3. To check our 3.3V linear regulator works, we simply measure the voltage across its terminals.
4. In order to verify that our microcontroller was soldered on properly, we hook up the programming headers on to a TI Launchpad and program it to set the signal to turn the boost converter to high.
5. Next we check that the enable signal to the boost converter is high, and then check the output of the boost converter. It should read a value of $13.5V \pm 0.3V$.
6. Sample the output from the random number generator to make sure that it fluctuates.
7. Reprogram the microcontroller to send a 'Hello World' message over the NRF24L01 to a known working chip.

Furthermore to verify that our random number generator was correct, we first hooked it up to an oscilloscope. The output should look something like this. We want to extra certain that the maximum Voltage is 3.3V and minimum Voltage is 0 to distinguish it from random noise from the environment.



After this step, we plugged this into the serial port and ran the NIST SP-800-22 test suite [4]. We were originally planning to run the Diehard test suite, however the source code for the test suite was hosted on

a server in Hong Kong and we were not able to access the server. As a result we decided to run the NIST SP-800-22 as it is another reputable test suite.

From the results from the test suite, the one that we scored the lowest one was in the bitstring compression test, which we scored around 0.88 bits of entropy per bit sampled. Theoretically a random string shouldn't be compressible. However the numbers that we produce are somewhat compressible. As a reference point, the random number generator from the linux kernel returns a minimum entropy value of around 0.93 bits of entropy per bit sampled. Keep in mind that the linux kernel performs some cryptographic whitening and performs lots of preprocessing. The full results are included in the appendix.

2.4.2 Software

As discussed earlier we had to perform the Ristretto construction on Curve 25519. Given that the paper for this came out roughly 7 months ago, no implementation for the MSP430 microcontroller exists. In order to test that our implementation was correct, we generated the follow test vectors using the cryptography library, libsodium [8].

1. Let a be $0xECE445$, then g^a should yield us $0x90b1090256df6455fc24217994199d0ee9b4b331b72562f83edf4a79118be26$.
2. Let a be $0xECE445$, $c = 0xCAFE$, $t = 0xDEADBEEF$, then $a \times c + t$ should yield $0x167878046$. Let $A = g^a$ and $T = g^t$, then $A^c + T$ should give us $0xBE14D8Ef7E7B07678EDE4DFDEB1BFB4AA6A3CA9B1C2D9A9778070F76617C844$

3 Cost

3.1 Parts

Tag Parts	Cost
TLV3201AIDBVR Analog Comparator [9]	$0.496
PCBs (JLCPCB)[10]	$0.04
Microcontroller (MSP430) [11]	$2 * \$1.68 = \$3.36$
NRF24L01 [9]	$2 * \$1.198 = \$2.396$
10pF 0603 Capacitor [9]	$0.002704
1KΩ 0603 Resistor [9]	$0.00378
Battery Samsung 25R [9]	$2.99
TPS61041 Boost Converter [9]	$0.207576
6 pF 0603 Capacitor 0603 [9]	$0.003071
15-21/GHC-YR1S2/2T Green LED [9]	$0.029304
Female Micro USB Port [9]	$0.033495
LM358ADR Dual Op-Amp [9]	$0.062
10kΩ 0603 Resistor [9]	$5 * \$0.000437 = \$0.002185$
10μH 1210 Inductor [9]	$0.046288
X322516MLB4SI 16 MHz Crystal Oscillator [9]	$0.057927
10 μF Capacitor [9]	$2 * \$0.018053 = \$0.036106$
MBR0530 Schotty Diode [9]	$2 * \$0.020097 = \$0.040194$
TPS76933 Low Dropout Linear Regulator [9]	$0.369697
100kΩ Resistor [9]	$2 * \$0.003242 = \$0.006484$
MMSZ12VCW Zener Diode [9]	$2 * \$0.009196 = \$0.018392$
1MΩ Resistor [9]	$0.004755
2nF Capacitor [9]	$0.007028
MSP430FR5969 Microcontroller	$2.25
470pF Capacitor [9]	$0.003606
2kΩ Resistor [9]	$0.003804
4.7 μ F Capacitor [9]	$0.015139
TP4056 BMS [9]	$0.043009
47kΩ Resistor [9]	$0.000541
15-21/R6C-FQ1R1B/2T Red LED [9]	$0.018992
Total	$14.352

Reader Parts	Cost
TI Msp430FR5969 Launchpad [11]	$15.99
NRF24L01 [9]	$1.198
Total	$17.188

3.2 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:

$$\frac{\$96,518}{1yr} * \frac{1yr}{2080\ hrs} = \$46.40/hr$$
$$\frac{\$46.40}{1hr} * 3 * 2.5 * 80\ hrs = \$27,840$$

For a combined cost of $27,871.54.

4 Conclusion

4.1 Accomplishments

We were able to complete the project successfully. The main hardware component which is the Random Number Generator generates noise that we are able to sample. The sample is uniformly distributed and passes the NIST's Entropy Test. The test runs a series of smaller tests such as chi square, length of longest repeated substring test, and the IID permutation tests. The chi square test checks if the sample is more random than a distribution of a sum of squares of k independent random variables [12]. The IID permutation tests are a resampling technique that checks if it should reject a sample based off the mean comparisons between different samples. Furthermore, we were able to apply our Schnorr-Peeters security protocol between the tag and reader system and we verified its functionality through the web application. We were able to either grant or deny access to certain users depending on their tag through the web application and the reader system. We demonstrated that even if an eavesdropper assumes the tag's identity, without information that is exclusive to the tag and never transmitted over the air, the eavesdropper would get denied by the reader when the reader issues it challenges according to the protocol. All in all, we were able to demonstrate the improvements of our security measure when compared to current technologies.

4.2 Uncertainties

Lots of side channel attacks exist, such power analysis attacks. While we did do our best to prevent side channel attacks, given the wide scope and nature of these, there probably exists some side channel for our tag/reader system [1]. In order to help mitigate this, we will be releasing our implementation of Ristretto25519 on the MSP430 microcontroller as an open source project. We rely on the assumption that discrete logs are difficult to compute. While no quantum computer that is power enough currently exists, a quantum computer could theoretically solve for a user's private key given its public in polynomial time.

4.3 Ethical considerations

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' [13]. 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 [14]. 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.

In order to avoid any potential FCC violations and potential licensing issues, our RFID transmitter and

receiver will operation in the 2.4 GHz, which is part of the ISM band [15].

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 [13]. While we do not know of any potential side channel attacks, side channel attacks on our system may exist. These attacks may allow a malicious user to bypass our system or impersonate other users. As a result, we recommend that users combine this with other security systems such as electronic surveillance and security personnel.

4.4 Future work

There are many ways to improve this project. At the current state of our design, we power the tag through a battery. The benefits this offers is better range for the tag since it receives more power; however, this is not necessarily a good thing in our case since we only want to transmit information when in close proximity to the reader. Other disadvantages is the tag will have limited lifespan, more expensive than the solution we have, and it will be bulky, which is not ideal for everyday use when someone wants to fit the tag in their wallet. Thus, the solution to this is to have the tag be passively powered. This means having the tag power itself from the energy fields that the reader emits within a few feet. A passive tag is paper thin so they are light, cheap to make, and they will only power on in close proximity to the reader which solves the issue of transmitting information across large distances.

Moreover, we have the reader hooked up through USB to a laptop which is not ideal to have a computer attached to every reader. As for the solution, we need have a wifi chip that the reader can transmit information to the web application which would be hosted somewhere within the building. On this computer, only admins will have access to specific privileges like granting or denying access to a user and they would be able to do this through wireless means.

References

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

[2] 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.

[3] "MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers," Texas Instruments, Dallas, TX, 2008.

[4] H. M. T. S. Langley, A., "Elliptic curves for security," Internet Research Task Force, January 2016.

[5] M. Hamburg, "Decaf: Eliminating cofactors through point compression," in *Advances in Cryptology – CRYPTO 2015*, R. Gennaro and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 705–723.

[6] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77–94, Jun 1988. [Online]. Available: https://doi.org/10.1007/BF02351717

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

[8] F. Denis, "The sodium cryptography library," Jun 2013. [Online]. Available: https://download.libsodium.org/doc/

[9] "LCSC electronics." [Online]. Available: https://lcsc.com/

[10] "Pcb prototype pcb fabrication manufacturer." [Online]. Available: https://jlcpcb.com/

[11] "Mouser Electronics." [Online]. Available: https://www.mouser.com/

[12] "Chi-square statistic: How to calculate it / distribution." [Online]. Available: https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/chi-square/

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

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

[15] T. Mooring, "FCC Allocation History File," Federal Communications Commission Office of Engineering and Technology Policy and Rules Division, May 2019.

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.

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

□

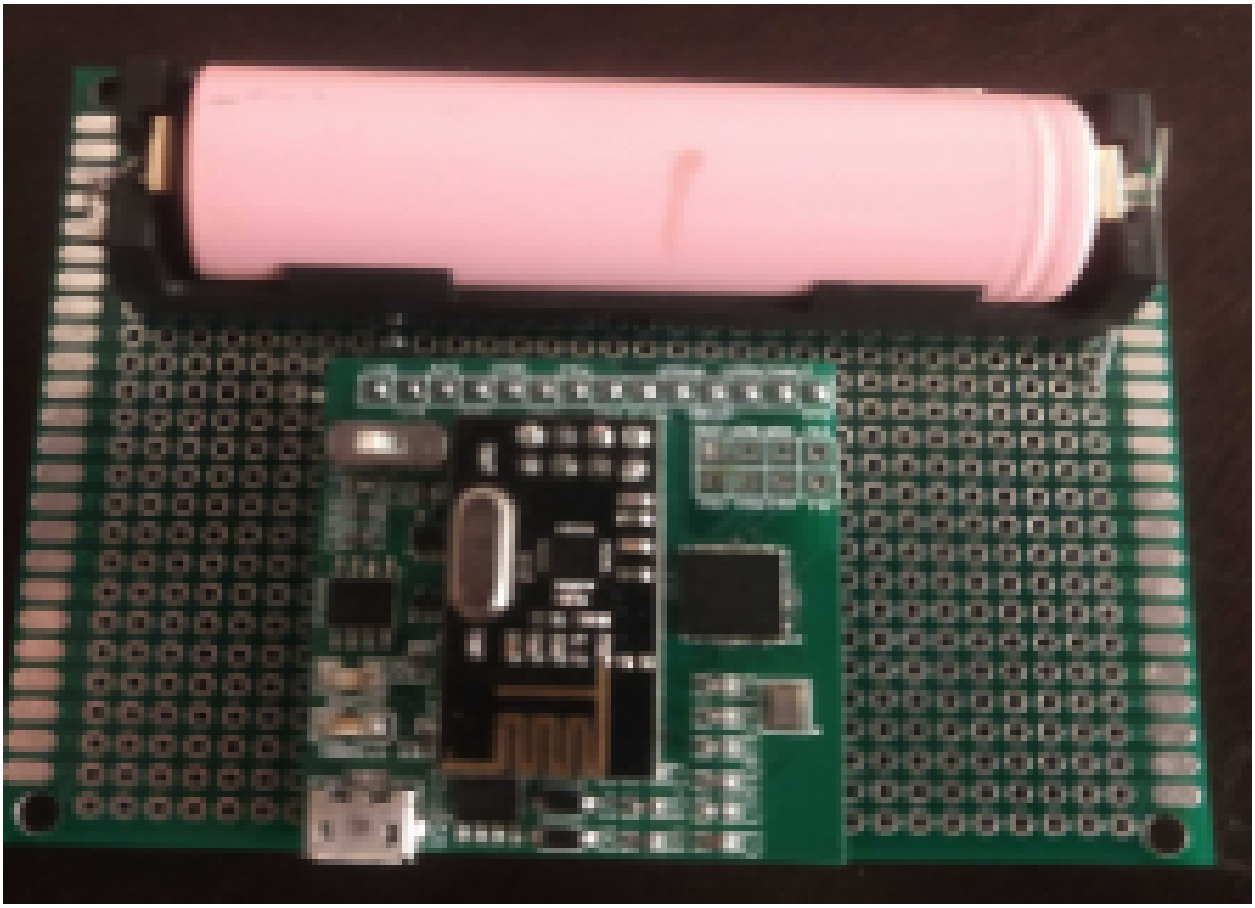
Appendix B Full Results of NIST SP 800-90B

```
Literal MCV Estimate: mode = 4115, p-hat = 0.0041149999999999997, p_u = 0.0042798947437199921
Bitstring MCV Estimate: mode = 4008631, p-hat = 0.50107887500000003, p_u = 0.50153422056036678
H_original: 7.868209
H_bitstring: 0.995580
min(H_original, 8 X H_bitstring): 7.868209

Chi square independence
        score = 64998.311797
        degrees of freedom = 65280
        p-value = 0.781970

Chi square goodness of fit
        score = 2317.713543
        degrees of freedom = 2295
        p-value = 0.365427

** Passed chi square tests

LiteralLongest Repeated Substring results
        P_col: 0.00390792
        Length of LRS: 4
        Pr(X >= 1): 1
** Passed length of longest repeated substring test

Beginning initial tests...

Initial test results
              excursion: 62436.2
     numDirectionalRuns: 666576
     lenDirectionalRuns: 8
  numIncreasesDecreases: 501888
          numRunsMedian: 499965
          lenRunsMedian: 22
           avgCollision: 19.6962
           maxCollision: 73
         periodicity(1): 3962
         periodicity(2): 3965
         periodicity(8): 3860
        periodicity(16): 3777
        periodicity(32): 3872
          covariance(1): 1.62942e+10
          covariance(2): 1.62914e+10
          covariance(8): 1.62917e+10
         covariance(16): 1.62921e+10
         covariance(32): 1.62942e+10
            compression: 1.06758e+06
```

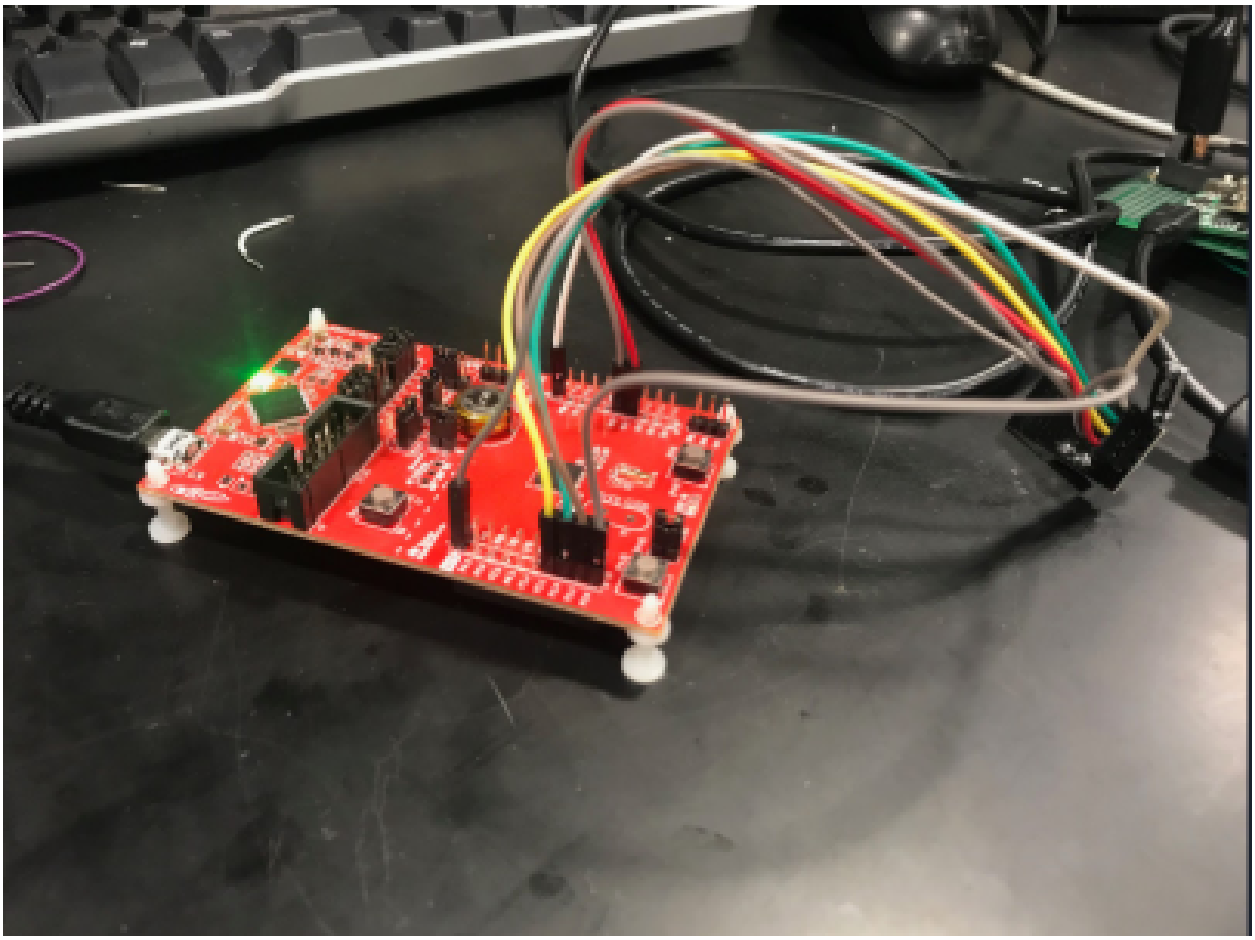
statistic	C[i][0]	C[i][1]	C[i][2]
excursion	6	0	10
numDirectionalRuns	10	0	6
lenDirectionalRuns	3	6	0
numIncreasesDecreases	9	0	6
numRunsMedian	7	0	6

```
Loaded 1000000 samples of 256 distinct 8-bit-wide symbols
Number of Binary Symbols: 8000000

Running non-IID tests...

Running Most Common Value Estimate...
Bitstring MCV Estimate: mode = 4008631, p-hat = 0.50107887500000003, p_u = 0.50153422056036678
        Most Common Value Estimate (bit string) = 0.995580 / 1 bit(s)
Literal MCV Estimate: mode = 4115, p-hat = 0.00411499999999997, p_u = 0.0042789474371999921
        Most Common Value Estimate = 7.868209 / 8 bit(s)

Running Entropic Statistic Estimates (bit strings only)...
Bitstring Collision Estimate: X-bar = 2.4995880365952234, sigma-hat = 0.49999990839825148, p = 0.52378939021014048
        Collision Test Estimate (bit string) = 0.932941 / 1 bit(s)
Bitstring Markov Estimate: P_0 = 0.40892112500000002, P_1 = 0.50107887499999992, P_0,0 = 0.49906172520298803, P_0,1 = 0.50093827479701192, P_1,0 = 0.49878125474756846, P_1,1 = 0.50121874525243149, p_max = 4.0121191586070611e-39
        Markov Test Estimate (bit string) = 0.996491 / 1 bit(s)
Bitstring Compression Estimate: X-bar = 5.2165720934294484, sigma-hat = 1.0163743548300217, p = 0.025148822483072241
        Compression Test Estimate (bit string) = 0.885569 / 1 bit(s)

Running Tuple Estimates...
Bitstring t-Tuple Estimate: t = 19, p-hat_max = 0.52233155719513469, p_u = 0.52278644942750307
Bitstring LRS Estimate: u = 20, v = 58, p-hat = 0.53768543346566977, p_u = 0.53813948488484464
        T-Tuple Test Estimate (bit string) = 0.935706 / 1 bit(s)
Literal t-Tuple Estimate: t = 2, p-hat_max = 0.005916082741141257, p_u = 0.0061136185978303758
Literal LRS Estimate: u = 3, v = 4, p-hat = 0.0039603588267866858, p_u = 0.004122138098759472
        T-Tuple Test Estimate = 7.353758 / 8 bit(s)
        LRS Test Estimate (bit string) = 0.893948 / 1 bit(s)
        LRS Test Estimate = 7.922391 / 8 bit(s)

Running Predictor Estimates...
Bitstring MultiMCW Prediction Estimate: N = 7999937, Pglobal' = 0.50064891240352249 (C = 4001517) Plocal can't affect result (r = 25)
        Multi Most Common in Window (MultiMCW) Prediction Test Estimate (bit string) = 0.998129 / 1 bit(s)
Literal MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.0041128478548804123 (C = 3951) Plocal can't affect result (r = 3)
        Multi Most Common in Window (MultiMCW) Prediction Test Estimate = 7.925647 / 8 bit(s)
Bitstring Lag Prediction Estimate: N = 7999999, Pglobal' = 0.50078265909219633 (C = 4002618) Plocal can't affect result (r = 24)
        Lag Prediction Test Estimate (bit string) = 0.997743 / 1 bit(s)
Literal Lag Prediction Estimate: N = 999999, Pglobal' = 0.0039962536141429058 (C = 3837) Plocal can't affect result (r = 3)
        Lag Prediction Test Estimate = 7.967136 / 8 bit(s)
Bitstring MultiMMC Prediction Estimate: N = 7999998, Pglobal' = 0.50138034612934412 (C = 4007399) Plocal can't affect result (r = 21)
        Multi Markov Model with Counting (MultiMMC) Prediction Test Estimate (bit string) = 0.996023 / 1 bit(s)
Literal MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.0039768643119714592 (C = 3818) Plocal can't affect result (r = 3)
        Multi Markov Model with Counting (MultiMMC) Prediction Test Estimate = 7.974153 / 8 bit(s)
Bitstring LZ78Y Prediction Estimate: N = 7999983, Pglobal' = 0.50147866082508374 (C = 4008178) Plocal can't affect result (r = 21)
        LZ78Y Prediction Test Estimate (bit string) = 0.995740 / 1 bit(s)
Literal LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.0039748824734387282 (C = 3816) Plocal can't affect result (r = 3)
        LZ78Y Prediction Test Estimate = 7.974872 / 8 bit(s)

H_original: 7.353758
H_bitstring: 0.885569

min(H_original, 8 X H_bitstring): 7.084548
```

Appendix C Physical Design of Tag



Appendix D Physical Design of Reader



Appendix E Requirement and Verification Table

Table 1: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Tag System (a) When the battery cannot supply sufficient power to power the random number generator, it must shutoff.	1. (a) First model the battery with an constant voltage power supply, which is initially set a 3.7V. Attach various resistive loads across the 3.3V and 14V terminals. Monitor the voltage across the 14V terminal and measure it shuts off when it dips to the 12.5V. (b) Next perform a similar test, except with the battery.	1. (a) Y (b) Y
2. (a) Must produce statistically random numbers. (b) Must have a data rate of 100 kbps.	2. (a) Have the random number generator pass the Duke Universities Diehard Test, which is a trusted source to test if random number generators are statistically random. We plan on running the test on a Raspberry Pi. The Diehard Test expects random numbers in the form of 32 bits and so we will write a program that returns 32 bits sampled from our random number generator. (b) Start sampling the random number generator at 100 Hz and measure serial correlation of the output. Gradually increase the sampling frequency until the magnitude of the serial correlation exceeds 0.05.	1. (a) Y (b) Y
2. (a) 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 shutoff non essential components, when the tag is not in use. As a goal we would like to get power consumption to be under 10 mW.	2. (a) To verify this, we will allow the power to idle and measure the power consumption from an external power supply.	1. (a) Y

Continued on next page

Table 1 – continued from previous page

Requirement	Verification	Verification status (Y or N)
2. (a) We would like to transmit our protocol as quickly as possible and so it should be transmit data at a rate of at least a net data rate 1 Mbps(data sent - ECC). (b) The data sent should resistant toward corruption.	2. (a) To test this, we will set up the transponder and receive so that it will send an ACK message whenever it receives a packet and won't send another packet until it receives an ACK message back. We will hold the transponder roughly 5cm away from the reader and continuously send messages of "Hello World!". (b) While testing for the ACK, we will collect the sampled "Hello World!". This will be considered sufficient if we are able to receive 1000 messages in a row without any corruptions.	1. (a) Y (b) Y
2. (a) It reads the RFID tag. (b) It has to support Error Correction Codes.	2. (a) Upon Tag being scanned, the reader authenticates user 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.	1. (a) Y (b) Y
2. Web Application (a) The web application would be able to indicate if a user has been successfully authenticated. (b) Additionally the interface would allow users to request for access whiles admin to add new users and revoke access.	2. (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. (c) Test if Tag gets rejected after revoking access. (d) Test if Tag gets accepted after adding new user.	1. (a) Y (b) Y (c) Y (d) Y