zkTap: A Zero-Knowledge Trustless Authentication Protocol

ECE 445 Senior Design Fall 2019; Team 33: Joseph Kuo, Majdi Hassan, Lilan Yang; TA: Evan Widloski

Department of Electrical and Computer Engineering, Grainger College of Engineering, University of Illinois at Urbana-Champaign

PROBLEM STATEMENT

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.

PROPOSED SOLUTION

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

TOP LEVEL BLOCK DIAGRAM

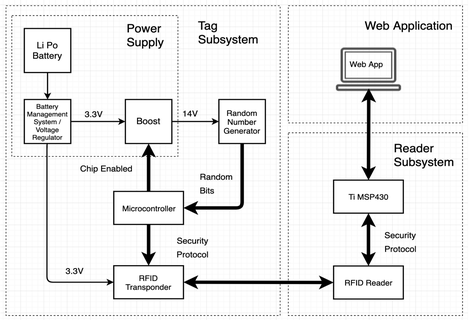

Figure 3: Block Diagram

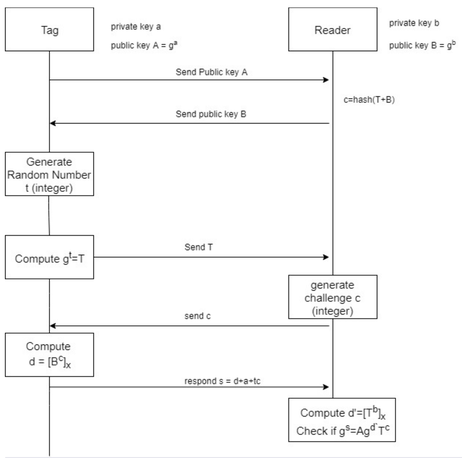

Figure 4: Protocol

SUBSYSTEMS

Tag System

The subsystem consists of power supply and management system, random number generator, a microcontroller, and an NRF24l01 chip. The microcontroller utilizes the output of the random number generator for calculations that help determine that the tag is who it claims to be when communicating with the reader. The NRF24l01 chip allows for this communication to take place between the tag and reader systems.

Reader System

The subsystem consists of msp430 and an NRF24l01 chip. The reader system interacts with the tag subsystem and web application subsystem. The reader communicates with the tag via NRF24l01 chips and verifies if the tag is who it claims to be. Then, it grants or denies access by pulling information from the web application and determining if the tag has access rights or not.

Elliptic Curve Cryptography

Originally we chose to use the curve, Curve25519, as it is less computationally expensive to implement than most other elliptic curves. Rather than using a large prime order, we use an elliptic curve of size 8p. While this allows faster computations, we have to clear the last three significant bits in order to prevent small subgroup attacks. Because of this clamping, we cannot implement our protocol since the verification step isn't guaranteed to hold true.

Instead we chose to use the curve, Ristretto25519, which builds off of the curve, Curve25519 and allows us to cheaply provide a prime order group.

Random Number Generator

Often times computers have a hard time generating random numbers as most processes are deterministic. To make it even harder for us, we do not have access to that many sources of entropy on a microcontroller. Instead we create our own source of entropy.

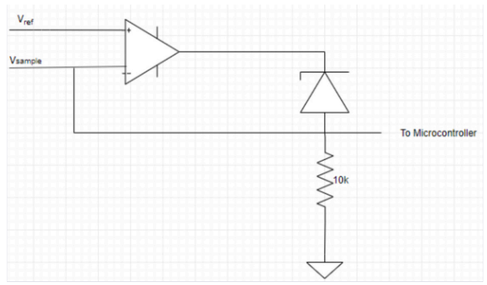

Figure 3: Zener Diode noise cell

In order to harvest entropy, we used avalanche noise from a Zener diode and set it up the following cell configuration.

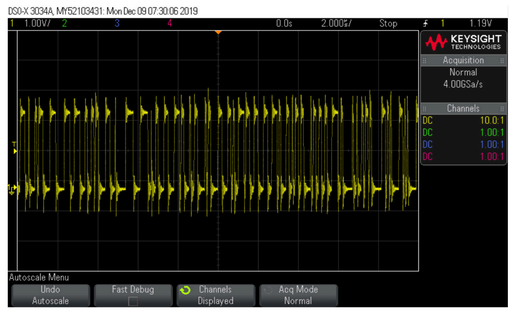

Figure 4: Waveform of Random Number Generator

```
[joseph@null cpp]$ ./ea_non_iid ../bin/rng2.bin

Running non-IID tests...

Running Most Common Value Estimate...

Running Entropic Statistic Estimates (bit strings only)..

Running Tuple Estimates...

Running Predictor Estimates...

H_original: 7.353758
H_bitstring: 0.885569

min(H_original, 8 X H_bitstring): 7.084548


[joseph@null cpp]$ ./ea_iid ../bin/rng2.bin
Calculating baseline statistics...
H_original: 7.868209
H_bitstring: 0.995580
min(H_original, 8 X H_bitstring): 7.868209

** Passed chi square tests

** Passed length of longest repeated substring test

Beginning initial tests...
Beginning permutation tests... these may take some time
** Passed IID permutation tests
```

Figure 5: Results from NIST SP800-90B Entropy Assessment Test

Protocol

The interaction between the tag and reader systems follows a certain protocol. First, the tag and reader exchange public keys. Then, the tag will utilize the random number generator and compute its point on the elliptic curve and transmits this information to the reader. The reader will generate a challenge from the information it just received and transmits the challenge to the tag. The tag will compute an answer according to the challenge and the reader will verify if this message is correct.

Web Application

Written in Python and Django, the web application functions as a visual aid for the authentication system to create a door accessing scenarios where students can request access to doors in ECEB. Users can login as admin and students where students can request access to all doors available in the SQLite database while admin can either accept or reject these requests. It also shows alert whether if there's a failed or successful attempt and log these attempts.

RESULTS

Misc Results: Random Number Generator >200 kbits/s the software UART doesn't allow for very fast transfer rates and so we were not able to properly benchmark the throughput of our Random Number Generator.

Overall circuit consumes roughly 15mA~20mA of power @ 4.2V, however the majority of it is consumed by the NRF24L01 chip and so if we were to get rid of it we could significantly cut down on power usage.

Random Number Generator consumes ~5mW of power, however this can be further reduced if we use Zener diodes with a lower avalanche breakdown voltage.

Tag handshake takes around ~12 seconds, however can be accelerated by implementing Ristretto25519 in assembly. transmits the challenge to the tag. The tag will compute an answer according to the challenge and the reader will verify if this message is correct.

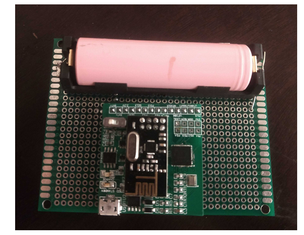

Figure 6: Tag

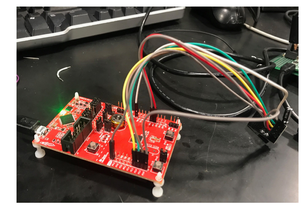

Figure 7: RFID reader connecting to TI MSP430

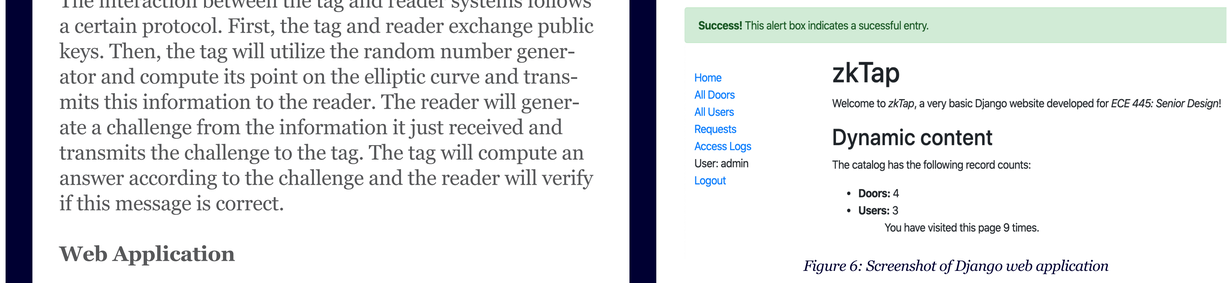

Figure 6: Screenshot of Django web application

ILLINOIS