A Reconfigurable Arduino Crypto FPGA Shield

Ryan Bornhorst
Gomathy Venkata Krishnan
Meiqi Zhao
Dustin Schnelle

Version 1.2 1/24/18

Table of Contents

Purpose of Product Design Specification

This product design specification fully defines the high-assurance security provided by the reconfigurable FPGA crypto shield designed to be compatible with the Arduino. The document is a detail reference guide for the team members, faculty advisor, and project sponsor associated with the Capstone at Portland State University.

Background

Inexpensive open hardware platforms, such as the popular Arduino[1], have had a huge impact on the embedded systems industry. Especially in the world of the Internet-of-Things (IoT)[4]. Arduino's open-source microcontroller platform has become popular to both hobbyists and educators because of its easy-to-use hardware and software interface. The low cost of the Arduino platform also helps. As Arduino has become more popular in academics, the amount of documentation for the Arduino platform has greatly increased. There are already two course at the University of California that teach Arduino, its programming environment, and interfacing principles for an IoT specialization[1].

With the massive growth of the IoT the world is changing, allowing for the innovation of new designs and embedded devices within the home. This growth of devices depending on a connection to the internet means that we need to focus on the security as a first class component when communicating sensitive data. It's becoming increasingly necessary to ensure the security of embedded systems connected to the network, so adding cryptography[4] to the functionality is important.

One solution to the problem is using an FPGA[5] shield to perform the encryption and add to the functionality of the Arduino. The current Arduino FPGA shields on the market however don't guarantee high-assurance[6] of the software, firmware, and hardware. High-assurance is needed for the embedded system to provide objective evidence of the system's correctness and security.

---

[1] The Arduino is an open-source electronics platform based on easy-to-use hardware and software.

[2] The Internet of Things (IoT) revolves around increased machine-to-machine communication. Its' build on cloud computing and networks of date-gathering sensors. It's going to make everything in our lives from streetlights to seaports smart

[3] Arduino has proved to be a very popular educational platform in embedded engineering. In addition, it can be used to overcome a number of challenges facing embedded systems nowadays.

[4] Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries. It's all about constructing and analyzing protocols that prevent adversaries or the public from reading private messages.

[5] Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnect. They can be reprogrammed to desired application or functionality requirements after manufacturing.

[6] High-assurance guarantee that a system is not susceptible to conventional hacking attempts. The makers of these systems use tools and techniques that produce high assurance software: software that is mathematically proven to function as intended, and only as intended.

Usually this evidence comes from a hand-written test bench, but we are looking to obtain this evidence from formal assurance or what is known as formal methods[7]. Many of systems that are important to national security to undergo rigorous certification and evaluation regimes, but some systems require stronger evaluation and certification to provide guarantees and mathematical evidence that they function exactly as intended at all times.

In the past there have been a few attempts of creating a shield for the Arduino to add the ability for the perform encryption functions on the Arduino platform like the CryptoShield[8] developed by SparkFun. Shields like the CryptoShield use dedicated chips to perform functions like the SHA[9]. It also features a dedicated chip to handle HMAC[10] which may be used to simultaneously verify both the data integrity and the authentication of a message, but the shield doesn't guarantee high-assurance. The fact that it uses dedicated chips also means that the functionality of the encryption features can't be reconfigured like they could with the use of an FPGA. The introduction of FPGA based encryption would allow for flexibility of the algorithms being implemented. The implementation of an FPGA Arduino shield is not the first of its kind as it was once for a Hackaday project[11], but this FPGA shield focused on expanding the overall functional of the Arduino and doesn't focus on cryptography or mention it. Even then the shield doesn't guarantee high-assurance of crypto algorithms, so this means that there is a major need for a reconfigurable FPGA based Arduino shield when it comes to dealing with the ever expanding IoT.

Problem Statement

In current internet market, the embedded system architects rarely pay attention to security as a first class component, which could cause many vitally important information to be leaked. The sponsor's primary goal of the project is to develop a high-assurance crypto FPGA shield for Arduino. This vision system should be able to guarantee performancing high-assurance crypto on inexpensive Arduino devices.

[7] Formal methods (or formal assurance) refer to class of mathematics used to find and close cyber vulnerabilities.
[8] The SparkFun CryptoShield is a dedicated security peripheral for the Arduino that adds specialized ICs that perform various cryptographic operations. This allows for the addition of a hardware security layer for Arduino projects.
[9] Secure Hash Algorithms (SHA) are a family of cryptographic hash functions published by the National Institute of Standards and Technology.
[10] Keyed-Hashing for Message Authentication (HMAC) is a mechanism for message authentication using cryptographic hash functions and provide a way to check the integrity of information transmitted over the world of open computing and communication.
[11] A low cost Arduino FPGA (Xilinx Spartan 6 LX9) shield including SDRAM and SPI Flash for configuration created by technolomaniac of Hackaday Projects.

Behavioral Requirement

- The system must be affordable and be built for under $100.
- The system's API must interface the FPGA via microcontroller.
- The system must exhibit high assurance to provide system correctness.
- The system must be high performance by being 10x faster than OpenSSL[12] current benchmarks.

Non-Behavioral Requirement

- The system should use model checking as a formal assurance technique.
- The system should use implementation tools to synthesis, validate, and verify cryptographic systems.
- The system should be built on an Arduino R3 format compatible shield.
- The system should be built from scratch.

Objective

- Gain expertise in Arduino programming.
- Gain expertise in FPGA programming especially for a specific FPGA shield.
- Design a custom Arduino FPGA board.
- Learn to use cryptographic libraries to implement high-assurance algorithms.
- Learn how to create APIs.
- The Arduino crypto FPGA shield should be dynamically reconfigurable.
- Learn to use Cryptol[13], SAW[14], and NaCl libraries[15].
- Learn how to create a PCB and interface the Arduino board with an FPGA embedded shield.

---

[12] OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Socket Layer (SSL) protocols. It's also a general purpose cryptographic library.

[13] Cryptol is a domain specific language for specifying cryptographic algorithms. An implementation of an algorithm resembles its mathematical specification more closely than an implementation in a general purpose language.

[14] Software Analysis Workbench (SAW) provides analysts with the ability to extract models for programs, and analyze them using a variety of automated reasoning tools.

[15] NaCl (pronounced "salt") is a new easy-to-use high-speed software library for network communication, encryption, decryption, signatures, and more. The goal is to provide all of the core operations needed to build higher-level cryptographic tools.

Deliverables

Delivered by Capstone Deadline

- *Proposal*: Document describing the background, motivation, and initial design requirements for implementing a Crypto FPGA Shield for the Arduino.
- *Test Plan*: This document will be used to verify that all of our system requirements can be met by identifying required testing instrumentation as well as test results.
- *Code Repository and Wiki*: All project documentation including schematics, source code, and general information will be located here.
- *Crypto FPGA Shield*: System Verilog code will be generated from Cryptol software provided to us from Galois. That code will then be uploaded to an FPGA shield. An Arduino will then be able to interact with the FPGA shield through an API.
- *Wrapper Cryptographic Software Library*: A variant of an AES library, such as Sodium, will be used within the Arduino development environment to configure the FPGA shield.
- *Demonstration*: Benchmarks and test applications will be used to demonstrate the use of the API[16] as well as validate our requirements for "formal assurance."

Time Permitting or Future Deliverables

- *Stretch Goal 1*: Our team would like to contribute our software library to the Arduino website and gain visibility to it by publishing periodic updates and announcements on related forums.
- *Stretch Goal 2*: Design a daughterboard containing an FPGA shield from scratch that can interface with an Arduino, use our software library, and meet all of the design requirements within this document.

[16] Application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. Specifically it's the set of clearly defined methods of communication between various software components.

Marketing Requirements

- The shield should design should be compact.
- The system should be plug and play.
- The system should cost less than $100 while maintaining high functionality.
- There will be an online forum that supports the consumers and gives them constant updates about the device and features.
- The system will cater to the requirements of the cybersecurity market and will have features that will be constantly updated.
- Future software updates will remain compatible with the hardware and the firmware used on the system.
- The system will be durable within its intended operating environment.

Engineering Requirements

Performance

- The system should guarantee “formal assurance.”

Functionality

- The Arduino environment and FPGA will be able to interface with each other.
- The system should have a reset (software or hardware) incase it hangs.
- The system API will be able to use at least 4 crypto algorithms.
- The FPGA should be chosen in a way that it will be dynamically reconfigurable.

Economic

- The total cost and manufacturing cost should not exceed $100.

Energy

- The system will operate for at least 5V or 3.3V logic.
- The system will function using the embedded system power input.
- The system will operate indefinitely while plugged in.

Legal

- The system will be 100% open source.

Maintainability

- Future software updates will remain compatible with the hardware and the firmware used on the system.

Manufacturability

- The custom board should be able to easily interface with the arduino headers.
- The custom board should be manufactured with readily available parts.

Reliability and Availability

- The product will have frequent software updates to ensure that it works as expected.

- The product will be operational 100% of the time.
- Future hardware or firmware updates will be communicated to the consumers months in advance.

Social and Culture

- The development team should contribute a library on the Arduino website.
- The team will publish announcements on related blogs and forums to gain visibility for the software library.

Usability

- Users of the system should be able to learn 80% of its functionality within 2 hours.
- The system API will be well documented.
- The API will be user friendly and can be subject to refactoring to enhance functionality at any time.
- Documentation for the hardware, software, and firmware will be provided.

System Component Block Diagram

The AtMega microcontroller will be powered from a PC source using a USB connector. This system will use the Arduino IDE to configure the AtMega MCU on the board. For the initial project design, a development board containing both the MCU and the FPGA will be used. The boards used will be the Mojo v3 and the Papilio One 250k. Both of these boards contain a variation of the AtMega controller as well as the Spartan 6 and Spartan 3E FPGAs respectively.

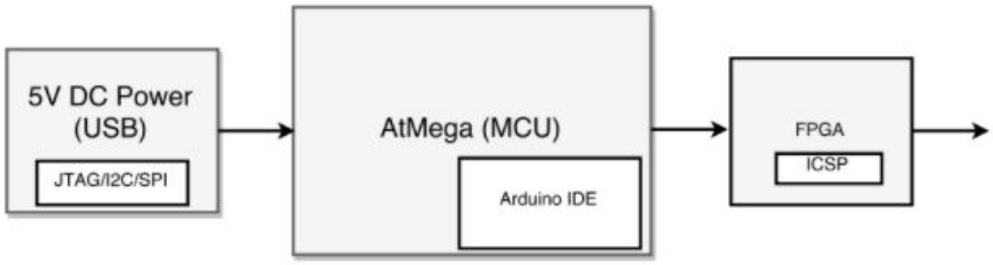

Figure 1. System Component Block Diagram

Software Component Block Diagram

AES Wrapper Library - Wrapper libraries take in a current software library's interface and turns it into a more compatible one. This is usually done to make the original library less complicated or to make the data between the library and the interface you are calling it from more compatible. The wrapper library that we are implementing will use the pre-existing cryptography library Sodium and make it compatible with the Arduino interface as well as the synthesis and formal assurance tools we will be using.

Cryptol - Cryptol's synthesis tools perform extensive and often very complicated transformations to turn Cryptol programs into hardware primitives available on target FPGA platforms. The formal verification framework of Cryptol allows equivalence checking between Cryptol and netlist representations that are generated by various parts of the compiler [17].

SAW - SAW is another tool designed by Galois that is used to formally verify properties of code written in C, Java, or Cryptol. SAW is closely connected with Cryptol and is commonly used to prove equivalence between a Cryptol specification of an algorithm and one that was written in C or Java [14].

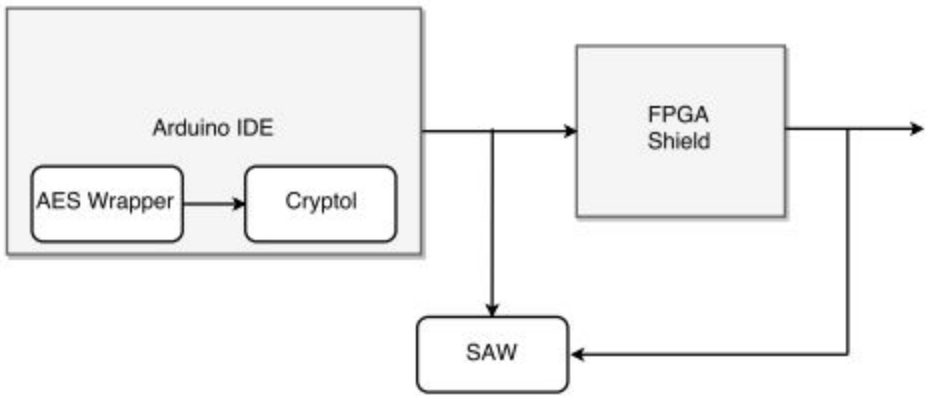

Figure 2. Software Component Block Diagram

Testing/Debugging

This part of the document is expected to change dynamically during testing as constraints and new measures to check for assurance are identified.

Test Writer:					
Test Name:	Software Compatibility Test				
Description:	This test will be used to test if the Cryptol and SAW software is able to compile and function properly. (generating expected outputs)				
Tester Information:					
Name:		Date:		Time:	
Setup:	Load Cryptol and SAW software onto the development boards.				
Test	Program Compiles	Test Hash	Test Key Encryption	Test Stream Cipher	
1 (Cryptol: Papilio One)					
2 (Cryptol: Mojo v3)					
3 (SAW: Papilio One)					
4 (SAW: Mojo v3)					

Test Writer:						
Test Name:	AES Library Functionality Test					
Description:	This test will be used to test if the Sodium library is able to compile as well as check if the benchmark tests produce valid results. (expected function output)					
Tester Information:						
Name:			Date:		Time:	
Setup:	Load Sodium library onto the development boards and test benchmark files if they exist.					
Test	Program Compiles	Test Hash Benchmark	Test Key Encryption Benchmark	Test Stream Cipher Benchmark		
1 (Papilio One)						
2 (Mojo v3)						

Test Writer:						
Test Name:	Model Checker					
Description:	This test will be used to test all possible inputs for our crypto algorithms as well as test that the program does not stall during testing. (exceed 30 min program time out)					
Tester Information:						
Name:			Date:		Time:	
Setup:	Use SAW to exhaustively test all inputs into our generated crypto functions.					
Test	Program Compiles	Model Check Hash (all inputs valid)	Model Check Key Encryption (all inputs valid)	Model Check Stream Cipher (all inputs valid)		
1 (Papilio One)						
2 (Mojo v3)						

Test Writer:						
Test Name:	Benchmark Comparison Test					
Description:	This test will be used to test speed benchmarks for x amount of inputs using original AES library and comparing it to the Cryptol generated library after loading it onto the FPGA. Test will determine % speed increase using an FPGA shield loaded with Cryptol generated HDL.					
Tester Information:						
Name:			Date:		Time:	
Setup:	Load Cryptol benchmark files and test them against the original Sodium library benchmarks.					
Test (Mojo v3 and Papilio One)	Program Compiles	Test Hash Benchmark	Test Key Encryption Benchmark	Test Stream Cipher Benchmark		
1 (10%)						
2 (20%)						
3 (30%)						
4 (40%)						
5 (50%)						
6 (60%)						
7 (70%)						
8 (80%)						
9 (90%)						
10 (100%)						

Licensing Information

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

This Project will follow all GPL licensing guidelines and processes to keep this open source and available for the overall community.

Proposal Approval

The undersigned acknowledge they have read/examined the project proposal document and agree with the material contained within. Any changes to this document require the discussion and approval of the undersigned.

_____	_____
Dustin Schnelle - Team Member	Date
_____	_____
Gomathy Ventaka Krishnan - Team Member	Date
_____	_____
Ryan Bornhorst - Team Member	Date
_____	_____
Meiqi Zhao - Team Member	Date
_____	_____
Dr. Christof Teuscher - Faculty Advisor	Date
_____	_____
Joe - Project Sponsor	Date
_____	_____
Dan - Project Sponsor	Date

Revision History

Version Number	Implemented By	Reversion Date	Reason
1.0	Team	1/29/2018	Initial PDS didn't have enough information
1.1	Team		Revised per Advisor and sponsors comments

Appendix A - Bill of Materials

Sno	**Product**	**Quantity**	**Cost/Piece**	**Total Cost**
1	Papilio One 250k	2	$37.99	$75.98
2	Mojo v3	2	$74.99	$149.98
3	PulseRain M10	2	$85.00	$170.00
4	Spartan 3E	1	$13.86	$13.86
5	Spartan 6	1	$23.73	$23.73
6	Intel MAX10	1	$7.10	$7.10
6	Arduino Mega	2	$45.95	$91.90
7	PCB	2	$10	$20.00
8	Soldering tools: Liquid solder, solder wick, reflux	1	$13	$13.00
9	M/M, F/M, F/F header pins or wires	1	$8	$8.00
10	Heat sink	2	$3	$6.00
Total:				$579.55

Appendix B - Gantt Chart

Business/Documentation Task Timeline

Task Name	Duration	Start	Finish	Task Lead
Portland State Capstone	**110 days**	**Mon 1/8/18**	**Fri 6/8/18**	
Initial Research and Project Proposal	**33 days**	**Mon 1/8/18**	**Wed 2/21/18**	
Overall Project Proposal	**33 days**	**Mon 1/8/18**	**Wed 2/21/18**	
Create initial product design specification	**16 days**	**Mon 1/8/18**	**Mon 1/29/18**	
Project background	5 days	Mon 1/8/18	Fri 1/12/18	Dustin
Define overall project objective	5 days	Mon 1/8/18	Fri 1/12/18	Gomathy, Meiqi
Marketing requirements	6 days	Fri 1/12/18	Fri 1/19/18	Whole Team
Engineering requirements	6 days	Fri 1/12/18	Fri 1/19/18	Whole Team
Linking marketing requirements to engineering requirements	6 days	Fri 1/19/18	Fri 1/26/18	Ryan
Finalize the project design specification	2 days	Fri 1/26/18	Mon 1/29/18	Whole Team
Project proposal	**19 days**	**Fri 1/26/18**	**Wed 2/21/18**	
Create initial project proposal template	3 days	Wed 1/31/18	Fri 2/2/18	Dustin
Create the initial hardware block diagram	4 days	Fri 2/9/18	Wed 2/14/18	Ryan

Finalize the hardware block diagram	2 days	Thu 2/15/18	Fri 2/16/18	Ryan, Gomathy
Create the initial software flow diagram	4 days	Mon 2/5/18	Thu 2/8/18	Ryan, Gomathy
Finalize the software flow diagram	2 days	Fri 2/9/18	Mon 2/12/18	Ryan, Gomathy
Create BOM for project prototyping	2 days	Fri 1/26/18	Sat 1/27/18	Gomathy
Create initial overall BOM	6 days	Mon 1/29/18	Mon 2/5/18	Gomathy
Finalize the overall BOM	5 days	Tue 2/6/18	Mon 2/12/18	Gomathy
Create initial test/debugging plan	4 days	Fri 2/9/18	Wed 2/14/18	Ryan, Gomathy
Finalize the test/debugging plan	2 days	Thu 2/15/18	Fri 2/16/18	Ryan, Gomathy
Finalize the project proposal (pending approval from Teuscher and sponsors)	4 days	Fri 2/16/18	Wed 2/21/18	Whole Team
Initial Gantt chart development	11 days	Fri 2/2/18	Fri 2/16/18	Dustin
Project Wiki	**19 days**	**Sun 2/18/18**	**Thu 3/15/18**	
Add information about the development boards being tested	4 days	Sun 2/18/18	Wed 2/21/18	Dustin
Add documentation of software packages	3 days	Wed 2/28/18	Fri 3/2/18	Ryan
Add information about chosen AES algorithms	3 days	Wed 2/28/18	Fri 3/2/18	Gomathy
Bibliography of project research	2 days	Wed 3/14/18	Thu 3/15/18	Meiqi
Marketing and Promotion	**5 days**	**Mon 5/28/18**	**Fri 6/1/18**	

Contribute libraries on the Arduino website	3 days	Mon 5/28/18	Wed 5/30/18	Whole Team
Publish appropriate announcements to related blogs and forums	2 days	Thu 5/31/18	Fri 6/1/18	Whole Team
Project Documentation	**16 days**	**Fri 5/4/18**	**Fri 5/25/18**	
Summarize test plan results	4 days	Fri 5/4/18	Wed 5/9/18	Meiqi
Detailed summary of progress with development board	4 days	Thu 5/10/18	Tue 5/15/18	Gomathy
Arduino forum and blog references (part of the stretch goal)	4 days	Mon 5/21/18	Thu 5/24/18	Whole Team
Final report	16 days	Fri 5/4/18	Fri 5/25/18	Whole Team
Capstone Poster	**10 days**	**Wed 5/16/18**	**Tue 5/29/18**	
Initial capstone poster layout	4 days	Wed 5/16/18	Sun 5/20/18	Whole Team
Create rough draft of the capstone poster	5 days	Mon 5/21/18	Fri 5/25/18	Whole Team
Finalize the capstone poster and submit for print	2 days	Mon 5/28/18	Tue 5/29/18	Whole Team
Final Presentation	**6 days**	**Fri 6/1/18**	**Fri 6/8/18**	
Practice Presentation	6 days	Fri 6/1/18	Fri 6/8/18	Whole Team
Final Presentation	1 day	Fri 6/8/18	Fri 6/8/18	Whole Team

Engineering Task Timeline

Task Name	Duration	Start	Finish	Task Lead
Initial Research	**33 days**	**Mon 1/8/18**	**Wed 2/21/18**	
Arduino Research		Mon 1/8/18	Wed 2/21/18	Whole Team
FPGA Research	**21 days**	**Wed 1/24/18**	**Wed 2/21/18**	
Research Xilinx FPGAs	19 days	Wed 1/24/18	Sun 2/18/18	Gomathy
Research Microsemiconductor FPGAs	19 days	Wed 1/24/18	Sun 2/18/18	Dustin
Research Lattice FPGAs	19 days	Wed 1/24/18	Sun 2/18/18	Ryan
Research Intel FPGAs	19 days	Wed 1/24/18	Sun 2/18/18	Meiqi
Compare and choose FPGA for project (dependent on success of prototype)	3 days	Mon 2/19/18	Wed 2/21/18	Whole Team
Research possible development boards	4 days	Mon 1/22/18	Thu 1/25/18	Whole Team
Initial software research	19 days	Fri 1/26/18	Wed 2/21/18	Meiqi
Initial popular AES algorithm research	14 days	Fri 2/2/18	Wed 2/21/18	Meiqi
Detailed Research	**23 days**	**Fri 2/9/18**	**Tue 3/13/18**	
Development Board Research	**23 days**	**Fri 2/9/18**	**Tue 3/13/18**	

Research documentation on the Mojo v3	7 days	Fri 2/9/18	Sun 2/18/18	Ryan, Gomathy
Research documentation on the Papilio One 250k	7 days	Fri 2/9/18	Sun 2/18/18	Dustin, Meiqi
Research documentation on the PulseRain M10	7 days	Fri 2/23/18	Sat 3/3/18	Whole Team
Summarize the findings on the Papilio 250k	5 days	Mon 2/19/18	Fri 2/23/18	Dustin, Meiqi
Summarize the findings on the Mojo 3v	5 days	Mon 2/19/18	Fri 2/23/18	Ryan, Gomathy
Summarize the findings on the PulseRain M10	5 days	Mon 3/5/18	Fri 3/9/18	Whole Team
Choose a development board for prototyping	2 days	Mon 3/12/18	Tue 3/13/18	Whole Team
Specific Software and Crypto Research	**11 days**	**Wed 2/21/18**	**Wed 3/7/18**	
Cryptol software research	6 days	Wed 2/21/18	Wed 2/28/18	Ryan, Meiqi
SAW software research	6 days	Wed 2/28/18	Wed 3/7/18	Ryan, Meiqi
AES algorithm research/AES benchmarks	11 days	Wed 2/21/18	Wed 3/7/18	Meiqi
Initial Testing/Prototype	**23 days**	**Wed 3/14/18**	**Sat 4/14/18**	
Development Board Testing	**18 days**	**Wed 3/14/18**	**Fri 4/6/18**	
Load wrapper library to the prototype	3 days	Wed 3/14/18	Fri 3/16/18	Whole Team
Test crypto algorithm #1	7 days	Sat 3/17/18	Sat 3/24/18	Dustin, Meiqi
Test crypto algorithm #2	7 days	Sat 3/17/18	Sat 3/24/18	Ryan, Gomathy

Test crypto algorithm #3	7 days	Sat 3/24/18	Sat 3/31/18	Gomathy, Ryan
Test crypto algorithm #4	7 days	Sat 3/24/18	Sat 3/31/18	Dustin, Meiqi
Test reconfigurability with all 4 crypto algorithms	6 days	Sat 3/31/18	Fri 4/6/18	Whole Team
Initial Prototype	**15 days**	**Sat 3/24/18**	**Sat 4/14/18**	
Measure the benchmarks of the crypto algorithms	12 days	Sat 3/24/18	Sun 4/8/18	Dustin, Meiqi
Confirm high-assurance	6 days	Mon 4/9/18	Sat 4/14/18	Ryan, Gomathy
Final Prototype (stretch goal)	**28 days**	**Sun 4/15/18**	**Tue 5/22/18**	
Determine which components are not needed by FPGA shield	2 days	Sun 4/15/18	Mon 4/16/18	Gomathy, Meiqi
BOM, schematic, and board layout	4 days	Mon 4/16/18	Thu 4/19/18	Dustin, Ryan
Order components need for shield	1 day	Fri 4/20/18	Fri 4/20/18	Gomathy, Meiqi
Send schematics and board layout for PCB production	6 days	Fri 4/20/18	Fri 4/27/18	Dustin, Ryan
Testing	**17 days**	**Sat 4/28/18**	**Tue 5/22/18**	
Bring up the FPGA shield prototype	7 days	Sat 4/28/18	Sat 5/5/18	Gomathy, Ryan
Test functionality of interfacing with Arduino	5 days	Sun 5/6/18	Thu 5/10/18	Dustin, Meiqi
Load wrapper library and algorithms	5 days	Fri 5/11/18	Thu 5/17/18	Whole Team
Test functionality of the crypto algorithms	3 days	Fri 5/18/18	Tue 5/22/18	Whole Team

Testing Wrap-up and Documentation	6 days	Wed 5/23/18	Wed 5/30/18	Whole Team

Appendix C - Mnemonics

Appendix D - Markdown for Hyperlinks(Oxygen)

SAW: https://galois.com/project/software-analysis-workbench/

Appendix E - Code References

A. Deshpande, M. Deshpande and D. Kayatanavar, "FPGA implementation of AES encryption and decryption", IEEE, 2009. Available: http://ieeexplore.ieee.org/document/5204421/

A. Gielata, P. Russek and K. Wiatr, "AES hardware implementation in FPGA for algorithm acceleration purpose", 2008 International Conference on Signals and Electronic Systems, 2008. Available: http://ieeexplore.ieee.org/document/4673377/

S. Varhade and N. Kasat, "Implementation of AES Algorithm Using FPGA & Its Performance Analysis", International Journal of Science and Research, vol. 4, no. 5, pp. 2484-2492, 2015. Available: https://pdfs.semanticscholar.org/3dcd/de6d1e3329d69083c10714866023da2ae068.pdf

A. Borkar, R. Kshirsagar and M. Vyawahare, "FPGA implementation of AES algorithm", 2011 3rd International Conference on Electronics Computer Technology, 2011. Available: http://ieeexplore.ieee.org/document/5941780/.

W. Fischer and N. Homma, "Cryptographic hardware and embedded systems" -- CHES 2017, 1st ed. Springer International Publishing, 2017. Available: http://www.springer.com/us/book/9783319667867

Wang W., Szefer J., Niederhagen R. (2017) FPGA-based Key Generator for the Niederreiter Cryptosystem Using Binary Goppa Codes. In: Fischer W., Homma N. (eds) Cryptographic Hardware and Embedded Systems – CHES 2017. CHES 2017. Available: https://eprint.iacr.org/2017/595.pdf

Joe Kiniry, "Galois Ultra Low Power High Assurance Asynchronous Crypto", Galois, 2016. Available: https://www.nist.gov/sites/default/files/documents/2016/10/19/kiniry-presentation-lwc2016.pdf

Bernstein D.J., Chou T., Schwabe P. (2013) McBits: Fast Constant-Time Code-Based Cryptography. In: Bertoni G., Coron JS. (eds) Cryptographic Hardware and Embedded Systems - CHES 2013. CHES 2013. Available: https://link.springer.com/chapter/10.1007/978-3-642-40349-1_15

Cherkaoui A., Fischer V., Fesquet L., Aubert A. (2013) A Very High Speed True Random Number Generator with Entropy Assessment. In: Bertoni G., Coron JS. (eds) Cryptographic Hardware and Embedded Systems - CHES 2013. CHES 2013. Available: https://link.springer.com/chapter/10.1007/978-3-642-40349-1_11

Hardin D., Hiratzka T.D., Johnson D.R., Wagner L., Whalen M. (2009) Development of Security Software: A High Assurance Methodology. In: Breitman K., Cavalcanti A. (eds) Formal Methods and Software Engineering. ICFEM 2009. Available: https://link.springer.com/chapter/10.1007/978-3-642-10373-5_14

Appendix F - Competitor Research

Sno	Product	Attributes	Cost	Availability
1	Sparkfun CryptoShield (Daughter Board)	• Encrypted and Decrypted Hardware installed • Encrypted EEPROM • RSA encryption/decryption • SHA-256 • Uses elliptic Digital Signature Algorithm	Retired Product	Retired Product
2	Arduino Compatible FPGA shield	• Spartan 6 FPGA shield includes SPI Configuration Flash, programmable from Arduino	Hackaday Project	Can be manufactured with the .sch, .brd and .gbr files
3	PicoEVB (Embedded Soc)	• Xilinx Artix XC7A50T • PCIe Gen 2 • PCIe Design prototyping(plug into PCIe slot in Laptop and program)	$249	Available on crowd supply currently
4	Snicker Doodle (ARM based SoC)	• Xilinx Zync • Dual Core ARM Cortex A9 • Wifi and BLE module • Crypto EEPROM	$95	Available on crowd supply currently

Appendix G - Glossary

High Security - Using cryptography to prevent unauthorized access to digital information. Data integrity and authenticity.
Formal Assurance/Formal Methods - Formal assurance requires reliability, which translates to 99.999% functional over a legitimate timespan. The software library should be functioning for all possible inputs and there should be no loss of data from memory. A model checking tool (Formal Methods technique) like SAW can be used for exhaustive logic based testing. Cryptol can also be used for exhaustive testing and can generate a mathematical proof of the algorithm used.
Crypto Algorithms - Mathematical algorithms, usually implemented in software, that are able to encrypt or decrypt data as a measure of security.
SAW (Software Analysis Workbench) - Formal verification software that is primarily used to verify cryptographic algorithms.
AES (Advanced Encryption Standard) - Software standard used implement reasoning, performance and accuracy.
Threat Modeling - Optimizing security by identifying vulnerabilities and defining countermeasures to prevent threats to the system.
Model Checker - Technique for automatically verifying correctness of all possible states within a system.
High Performance - Take reference benchmark, small program or case study, check before and after loading using the Sodium wrapper (Sodium wrapper dictates benchmark specs). Motivation for this is to exceed the processing speed of the original software benchmarks.
Random Number Generator (RNG) - An algorithm that generates a random number between some specified minimum and maximum value.
Hash Functions - A function that verifies that input data maps to a given hash value. This value is usually stored in a hash table that links the input to the corresponding hash value.
Ciphers - It is an algorithm in cryptography for performing encryption and decryption. To encipher or encode is to convert information into cipher or code.
Symmetric Ciphers - In a symmetric cipher(also known as secret key), the key that deciphers the cipher text is the same of can be derived from the key enciphers that clear text. Ex. AES and DES
Asymmetric Ciphers - The asymmetric cipher uses two keys : private and public that define who can view the content. These two keys cannot be derived from each other. Ex. RSA and DSA.
API (Application Programming Interface) - A set of clearly defined methods and/or specifications for communicating between various software components.

Bibliography

[1] “Arduino - Introduction.” Arduino. [Online] Available: https://www.arduino.cc/en/Guide/Introduction

[2] M. El-Abd, “A Review of Embedded Systems Education in the Arduino Age: Lessons Learned and Future Directions,” *International Journal of Engineering Pedagogy*, vol. 7, no. 2, pp. 79–93, Apr. 2017. [Online] Available: http://online-journals.org/index.php/i-jep/article/view/6845/4454

[3] D. Burrus, “The Internet of Things is Far Bigger Than Anyone Realizes,” *Wired*, vol. Innovation Insights, Nov-2014. [Online] Available: https://www.wired.com/insights/2014/11/the-internet-of-things-bigger/

[4] G. Kessler, “An Overview of Cryptography.” 01-Dec-2017. [Online] Available: https://www.garykessler.net/library/crypto.html#intro

[5] “What is an FPGA? Field Programmable Gate Array.” Xilinx. [Online] Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[6] “What is High Assurance.” Free & Fair. [Online] Available: http://freeandfair.us/articles/what-is-high-assurance/

[7] S. Freedberg, “Faster Than Thought: DARPA, Artificial Intelligence, & The Third Offset Strategy.” Breaking Defense, 11-Feb-2016. [Online] Available: https://breakingdefense.com/2016/02/faster-than-thought-darpa-artificial-intelligence-the-third-offset-strategy/

[8]“SparkFun CryproShield.” SparkFun. [Online] Available: https://www.sparkfun.com/products/retired/13183

[9]N. Landman, E. Ross, and C. Williams, “Secure Hash Algorithms.” Brilliant. [Online] Available: https://brilliant.org/wiki/secure-hashing-algorithms/

[10] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication.” IBM, Feb-1997. [Online] Available: https://tools.ietf.org/html/rfc2104

[11] A. Fablo, “A Low Cost Arduino FPGA Shield,” Hackaday, 05-Mar-2014. [Online] Available: https://hackaday.com/2014/03/05/a-low-cost-arduino-fpga-shield/

[12] “Welcome to OpenSSL” https://www.openssl.org/ [Online] Available: https://www.openssl.org/

[13] “The Language of Cryptography.” Galois. [Online] Available: https://cryptol.net/

[14] “SAW (Software Analysis Workbench).” Galois. [Online] Available: https://galois.com/project/software-analysis-workbench/

[15] “NaCl: Networking and Cryptography Library.” Computer Aided Cryptography Engineering. [Online] Available: https://nacl.cr.yp.to/

[16] “Application Programming Interface.” Wikipedia. [Online] Available: https://en.wikipedia.org/wiki/Application_programming_interface

[17] S. A. Browning, M. Carlsson, L. Erkok, J. Matthews, B. Martin, and S. Weaver, *“Empowering the Experts: High-Assurance, High-Performance, High-Level Design with Cryptol.”* [Online] Available: https://cryptol.net/files/empowering_the_experts.pdf [Accessed February 14th 2018]