A Reconfigurable Arduino Crypto FPGA Shield

Sponsored by Joe Kiniry and Dan Zimmerman, Galois: galois.com

Background and Motivation

Arduino and other small, inexpensive open hardware platforms have had a huge impact on the embedded systems industry, particularly in the world of the Internet of Things (IoT). One extends an Arduino device by attaching daughterboards called “shields”. Commonly available shields support I/O, sensors, motors, displays, etc.

Historically, and unfortunately, embedded system architects rarely pay attention to security as a first class component. Today, crypto code is either ad hoc, manually written, and likely insecure, or is provided by one of a small number of portable libraries like Sodium. In either case, the crypto facilities are used by developers who know little about security or cryptography.

Unfortunately, as witnessed by a exponentially growing number of vulnerabilities exposed—and the increasingly serious nature of threats from state actors—this attitude toward security is not just insufficient, but negligent.

This project focuses on remedying this situation through the development of a *high-assurance crypto FPGA shield* for Arduino. In the next few paragraphs we’ll explain what each of these key scientific terms means.

An *FPGA shield* is a daughterboard that contains an FPGA, which can be used by the main Arduino CPU as a computational partner. Several FPGA shields are available on the market at low cost. Such FPGAs are programmed using Verilog and standard FPGA tooling from companies like Altera and Xilinx.

Crypto means that the FPGA, along with a software library for its use, contains implementations of commonly used cryptographic algorithms such as random number generators, hash functions, symmetric ciphers, asymmetric ciphers, and digital signature algorithms. We wish to implement these functions in an FPGA for two reasons: (a) the performance of software crypto on inexpensive Arduino devices is very poor; and (b) the threat model for hardware crypto is significantly different than that for software-only crypto.

High-assurance means that the implementation of software, firmware, and hardware in the crypto FPGA shield includes objective evidence of the system’s correctness and security. Typically, such evidence comes in the form of a hand-written test bench that can be executed to vet the system under test. Our work at Galois focuses on *formal assurance*, or what is known as *formal methods*. We build tools that support *rigorous systems engineering* so that everyday

software and hardware engineers can use hidden formal methods—the application of mathematics to systems design, engineering, validation, and verification—to create systems that are proven correct and secure. In this domain, we have the ability to (a) automatically generate high-performance high-assurance cryptographic algorithm implementations in C and Verilog, and (b) provide formal or rigorous assurance that the generated implementations are correct and secure.

Objectives and Learning Outcomes

The team that works on this project will, with the support of Free & Fair experts, achieve the following objectives and learning outcomes. *Outcomes that are italicized are optional and will be supported/encouraged if there are team members with interest in the topic and relevant skills.*

- gain expertise in Arduino programming;
- gain expertise in FPGA programming, particularly for a specific FPGA shield;
- learn about the Sodium library, with respect to both design and use;
- *learn how to use Galois tools such as Cryptol and SAW for the synthesis, validation, and verification of cryptographic systems (Galois experts can simply provide implementations and assurance artifacts directly);*
- develop a variant of Sodium, or a wrapper cryptographic library, for Arduino that automatically leverages an FPGA shield; and
- *design a new Arduino FPGA shield from scratch that is dedicated to use in a security context (for a great team that wants to do a custom board design).*

Milestones

The team is expected to meet these milestones (reaching the stretch goals is optional):

1. The team sets up the development environment and is able to generate System Verilog code from existing Galois Cryptol demos and upload the code on the FPGA shield, and is able to interact with the FPGA shield from Arduino (for example via API)
2. The team develops a variant of Sodium, or a wrapper cryptographic library, for Arduino that automatically leverages an FPGA shield
3. The team develops a suite of examples, benchmarks and test applications to demonstrate a usage of the cryptographic Arduino library from Milestone 2
4. *(stretch goal)* The team submits the code from Milestones 2 and 3 as a contributed library on arduino website, and publishes appropriate announcements on related blogs and forums to gain visibility for the library
5. *(stretch goal)* design a new Arduino FPGA shield from scratch that is dedicated to use in a security context.

Student Skills

A team that tackles this ECE project will need team members that have expertise in, or are willing to build upon intermediate expertise in, each of the following:

- embedded systems programming and debugging in C;
- FPGA programming and debugging;
- the concepts of cryptography and their implementation in C and Verilog; and
- the concepts of applied formal methods and their realization via Galois technology.