An Algorithm-Agile Cryptographic Co-processor Based on FPGAs

Christof Paar[1], Brendon Chetwynd[2], Thomas Connor[3]
Sheng Yung Deng[4], and Steve Marchant[5]
ECE Department, Worcester Polytechnic Institute, Worcester, MA 01609 USA
[1] Email: christof@ece.wpi.edu
[2] Email: spunge@alum.wpi.edu
[3] Email: tommyj@ece.wpi.edu
[4] Email: y_ding_ding@hotmail.com
[5] Email: marchant@intraserever.com

Presented at: The SPIE's Symposium on Voice, Video, and Data Communications
September 20, 1999, Boston, MA

ABSTRACT

Cryptographic algorithm agility, or the capability to switch between several encryption algorithms, has become a desirable feature due to the algorithm-independent design paradigm of modern security protocols. Moreover, applications such as cell encryption in ATM networks require the ability to quickly change ciphers. A promising answer to algorithm agility in hardware is reconfigurable logic.

This contribution describes the design and implementation of an algorithm-agile cryptographic co-processor board. The core of the board is an FPGA which can be dynamically configured with a variety of block ciphers. The FPGA is capable of encrypting data at high speed through an ISA bus interface. The board contains a RAM with a collection of FPGA configuration files. In addition, the algorithms can be added or deleted during operation. The co-processor board also contains other reconfigurable logic and a microprocessor for control functions, and high-speed FIFOs for data storage. We report about the general design, our experiences with this proof-of-concept implementation, and recommendations for future work.

Keywords: cryptography, algorithm-agility, co-processor, FPGA, block copher, VHDL

1. INTRODUCTION

Implementations of cryptographic algorithms in hardware have several advantages over software implementations, including speed and security. Typical hardware implementations are ASICs or full-custom VLSI devices. In either case, the algorithm is fixed at the time of manufacturing. A problem can arise if the algorithm is "broken" and is no longer considered secure. In the previous case, the devices would have to be redesigned and re-manufactured, presumably at high cost. This is an example of the need for algorithm agility. Reconfigurable logic provides a hardware solution to algorithm agility.

We present an innovative approach to algorithm agility based on reconfigurable hardware. Our system uses a Xilinx FPGA (Field Programmable Gate Array) as a cryptographic co-processor. We begin with a brief introduction of cryptography in hardware and algorithm agility. We continue with a discussion of previous work in this area and then delve into a system overview of our design. We conclude with suggestions for future work.

This work is concerned with the implementation of symmetric ciphers. These algorithms are designed to use the same key to encrypt and decrypt information. There are two types of symmetric ciphers: block and stream. Block ciphers encrypt an entire block of data at once, while stream ciphers encrypt each individual bit of data as it is generated. This paper focuses on symmetric block ciphers.

All design descriptions referenced herein are an extension of the design work presented by Connor, Deng, and Marchant.[1]

2. ALGORITHM AGILITY, HARDWARE, AND HOW THEY CONNECT

2.1. Algorithm Agility

Algorithm agility can be defined as the ability to switch between cryptographic algorithms within a given system. Among the reasons for designing a system that supports algorithm agility is the algorithm-independent nature of today's security protocols and standards. Two such standards, Secure Sockets Layer (SSL) v3.0 and IPSec have been written to support many algorithms.

SSL v3.0 is included in almost all of today's world wide web browsers. Most times that a user accesses a secure web page, SSL is activated. During setup of the secure connection, a block cipher is chosen for encryption/decryption of the connection from the following list: RC4, RC2, DES, Triple DES, 40 bit DES, and Skipjack.[2]

IPSec, a proposed security architecture for the Internet Protocol (IP), is also defined to be algorithm independent. The IPSec specification discusses the implementation of security mechanisms for the protection of user traffic. These mechanisms are defined as algorithm-independent. This allows selection of different sets of algorithms without affecting other parts of the implementation.[3]

In addition to algorithm independent protocols, there is another argument for algorithm agility. What happens if a system is designed with a given algorithm, and that algorithm is broken or rendered insecure? Now the system, in either hardware or software, needs to be redesigned to support a new algorithm. This requires time and money. If the system was initially designed with multiple algorithms in mind, it would not be a problem if one of the algorithms was "broken". For example, the Data Encryption Standard (DES), was rendered insecure in 1998 and is used in SSL v3.0. Despite this fact, SSL v3.0 is still used to encrypt web sessions since, as previously mentioned, it supports many other algorithms.

This argument can be further strengthened when considering remote systems that are difficult to access. In the case of satellites, upgrading the hardware incurs tremendous cost. In addition to the cost of design and manufacturing the new hardware, the cost of upgrading is also extremely high.

These facts present a strong case for the design and implementation of algorithm agile (or independent) cryptosystems.

2.2. Cryptography in Hardware

Implementation of cryptography in hardware has two major advantages over software implementations: speed and physical security.

High speed encryption is becoming an increasingly important issue in the days of high speed networking. Today, Virtual Private Networks (VPNs) are widely used. VPNs are networks over unsecured channels. These channels are made secure using encryption. In order to supply the throughput required for VPNs implemented at ATM speeds, which can approach 622.08Mbps, hardware implementations are required. Even the fastest software implementations of the proposed algorithms for the Advanced Encryption Standard (AES) do not approach these speeds.[4] An extremely fast software implementation of DES, which operates at 137 Mbps,[5] still falls short of the slowest ATM speed of 155.54Mbps.

Physical security is an important component of a given algorithm's implementation. One important component of security in software is secure key management.[6] It is important that the keys used for encryption never be stored in cleartext on disk. However, the key may still reside in system memory. Even though techniques may be used to obscure the keys, a determined attacker may still be able to recover them.

Algorithms implemented in hardware are generally more secure than their software equivalent because hardware implementations provide the ability to protect keys. This is typically done by storing the key in special memory internal to the device. Thus, the attacker does not have easy access to it and therefore he/she cannot discover its value. In addition, hardware implementations provide a tamper-proof medium that ensures the correct implementation of algorithms. A hardware core can also assist with FIPS 140-1 certification.[7]

2.3. Reconfigurable Logic: How they connect

As mentioned, algorithm agility is now a desirable feature for cryptographic systems. Typically, systems implemented in hardware have less flexibility than those in software. Reconfigurable logic provides the flexibility of software with the potential of an order-of-magnitude increase in speed.[8]

FPGAs are large arrays of re-configurable blocks with logic to provide interconnect between these blocks. Typically the functionality is written in a hardware description language, such as VHDL or Verilog, and then special software tools are used to map this logic into the FPGA. FPGAs are configured at run time. This allows considerable flexibility. While the I/O of the device needs to be decided before manufacturing the PCB on which the FPGA resides, the functionality can be finalized much later. Stephen Brown and Jonathan Rose present a good tutorial on FPGAs.[9]

FPGAs are a trade-off. They typically do not provide as much throughput as a device that has been custom designed for a given algorithm, but they provide the flexibility of reprogrammability and thus, FPGAs can be configured to implement a variety of algorithms.

3. PREVIOUS WORK

In the field of cryptographic implementations, not a lot of work has been done on the implementations of block ciphers in FPGAs. The works cited here describe block cipher implementations in FPGAs.

Kaps describes the implementation of DES in Electronic Codebook Mode (ECB) within a Xilinx FPGA.[10] This work was able to achieve data rates of up to 400 Mbps depending on the device and the amount of pipelining used.

The Swiss Federal Institute of Technology presents a generalized architecture for implementation of block ciphers in FPGAs.[11] The CryptoCore, which is algorithm specific, interfaces to the other components of the design, such as the host interface and session memory. These blocks are constant for all algorithms. This architecture was designed to take advantage of partial reconfigurability of today's newest FPGAs. Partial reconfigurability allows a portion of the FPGA to be reprogrammed, while keeping the remainder constant. This takes less time than a complete reconfiguration, and is thus more desirable. The International Data Encryption Algorithm (IDEA) was implemented at 200 Mbps without any pipelining. Additional algorithms are planned.

A new, generalized reconfigurable architecture called PipeRench was developed at Carnagie Mellon University.[12] PipeRench provides near-custom hardware perfomance with the flexibility of a software implmentation. PipeRench has three important characteristics in regards to implementing cryptographic algorithms: hardware virtualization, pipelined datapaths for word-based computations, and zero apparent configuration time. The PipeRench architecture was used to implement IDEA. It achieved data rates up to 86 Mbps.

Elbirt describes the implementation of the CAST-256 algorithm, an AES candidate, in a Xilinx FPGA.[13] The resource requirements of CAST are immense, and thus the largest available Xilinx FPGA was used (Virtex family). This implementation achieved a data rate of 11.03Mbps. On the other hand, a software implementation of the CAST-256 algorithm on 200MHz PentiumPro PC achieved a data rate of 40.4Mbps.[14] There are several factors which cause the software implementation to perform significantly faster than the hardware implementation. First, typical FPGA clock rates are in the order of 50 MHz. The Pentium Pro, in this example, has a clock rate of 200MHz. In addition, the nature of the CAST-256 algorithm is such that it is not easily implemented in FPGAs (large table lookups). These tables require memory and in today's FPGAs memory is a much more limited resource than in a Pentium Pro system.

4. SYSTEM OVERVIEW

In verifying the proof-of-concept, it was decided that implementing a test system in hardware was the best approach. Simulation is certainly a viable form of verification, but implementation in hardware provides greater confidence that the system will function as desired.

In designing the test system, the following list of requirements was generated:

- *Algorithm Switching on-the-fly*
- *Downloading of New Algorithms*

- *High-Speed Data Encryption*
- *Graphical User Interface*
- *Upgradable Firmware*
- *Upgradable Hardware*

Our implementation consists of a full-length, 16 bit ISA card installed in a standard PC (herein referred to as the Host PC). A custom GUI was developed to allow the user to download algorithms into the FPGA and process data. This overview focuses on the hardware functionality.

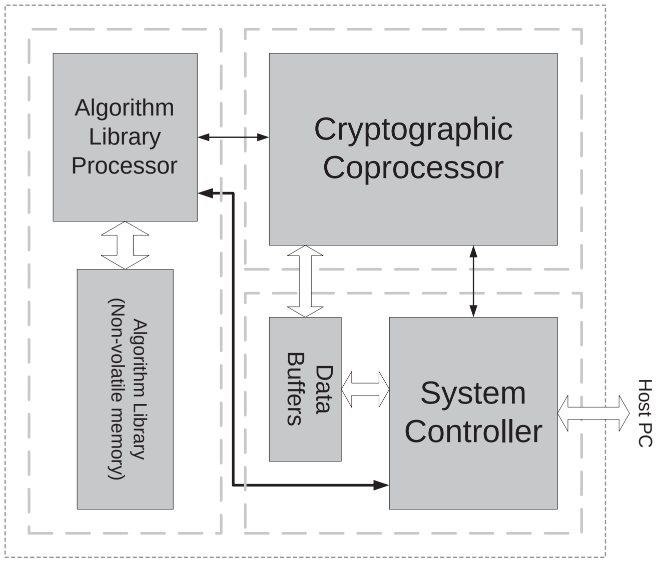
Algorithm Library Processor
Cryptographic Coprocessor
Algorithm Library (Non-volatile memory)
Data Buffers
System Controller
Host PC

Figure 1. Test System Overview

Figure 1 presents a block diagram of the test system. The system has three major functional blocks:

- *Cryptographic Co-processor*
- *System Controller*
- *Algorithm Library*

The cryptographic co-processor performs all encryption and decryption of data. Upon power-up, the FPGA, in which the co-processor resides, is unprogrammed. Upon a selection of an algorithm by the user through the GUI, the system controller commands the algorithm library processor to download the selected algorithm to the FPGA. At this point, the co-processor is ready to process data.

When the user selects a file on the host PC, the system controller begins downloading the data to the data buffers via the ISA bus. Once data has begun downloading, the system controller instructs the co-processor to begin encrypting/decrypting data. Processed data is passed back through the data buffers, through the system controller, and back to the host PC.

4.1. Cryptographic Co-processor

The cyptographic co-processor resides in a Xilinx FPGA. FPGAs are volatile devices that must be programmed when power is applied. Any previous state information is lost when power is removed. In our system, the Algorithm Library programs the FPGA based on the user's choice of a particular algorithm. Once programmed, the co-processor is ready to begin processing. Upon a signal from the system controller, the co-processor will accept a key, calculate the corresponding key schedule for the selected algorithm, then read in the first block of data.

All algorithms were coded using VHDL and implemented using the Xilinx Foundation tools on a PC based platform running Windows NT. Simulation of algorithms were performed using the Aldec Active-VHDL simulation tool.

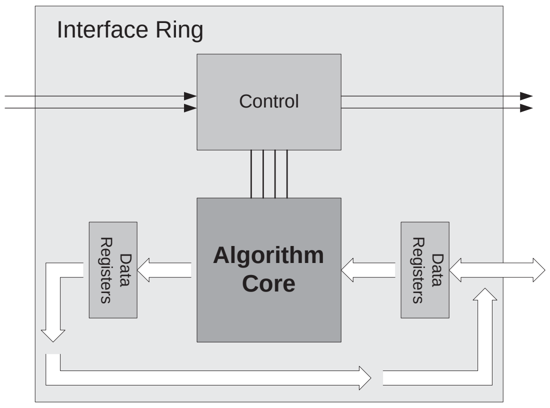

Figure 2. Cryptographic Co-processor Block Diagram

One of the requirements of the cryptographic co-processor was to have a standard interface for all algorithms. Since the system level design is fixed, an *interface ring* was developed. This structure provides an interface between the algorithm and the system (see Figure 2).

The interface ring is composed of three main blocks:

- *Control logic*
- *Algorithm Core*
- *Data Registers*

The control logic interprets system control signals and control words. It then passes this information to the algorithm core. In addition, any status information reported by the core goes through the control logic before leaving the FPGA and reaching the system level.

The algorithm core contains any algorithm specific logic. All key scheduling and encryption / decryption processing is contained here. Any state machine required to control processing is contained at this level.

The data registers provide a location for the incoming and outgoing data to reside since the data bus is bidirectional. I/O control is performed by the control logic.

4.2. System Controller

The system controller arbitrates the interaction between the cryptographic co-processor, algorithm library, and the host PC. It is implemented in a non-volatile Altera CPLD (Complex Programmable Logic Device). All functionality has been described in VHDL and programmed using the Altera Max-plus-II development system.

4.3. Algorithm Library

The algorithm library can hold up to 7 algorithms for the cryptographic co-processor. As previously mentioned, the algorithm library will download a specific algorithm to the co-processor upon a user's request. In addition, the user, through the host PC, has the option of adding additional algorithms to the library. All interaction between the host PC and the algorithm library is through the system controller.

The algorithm library processor is implemented in a Motorola micro-controller. The processor interacts with the algorithm library memory. The library memory is composed of non-volatile memory. This is to prevent the algorithms from being lost every time power is removed from the test system.

5. FUTURE WORK

Several opportunities for improvement exist when developing the next generation of our Algorithm-Agile Cryptographic Co-processor. Due to the relatively slow speed of the ISA bus, the current test system does not test the full throughput of the co-processor. Thus, it is desired to upgrade the ISA bus to the PCI bus. The PCI bus allows wider (32bit vs. 16bit) and faster (33MHz vs. 8Mhz) data transfers.

The current FPGA used to implement the co-processor has a medium size equivalent gate count on the order of 45,000.[15] Today's largest FPGAs have gate count in the order of 300,000+. By upgrading to a larger FPGA, we will be able to implement algorithms that require more resources. Currently, Altera,[16] Xilinx,[15] and Lucent[17] all have FPGAs in this range. With a larger FPGA, implementing a wide range of AES candidate algorithms will be feasible.

REFERENCES

1. T. Connor, S. Deng, and S. Marchant, *Cryptographic Coprocessor with Algorithm Agility.* ECE Dept., Worcester Polytechnic Institute, Worcester, MA, March 1999. Major Qualifying Project (Senior Thesis).
2. A. Freier, P. Karlton, and P. Kocker, *The SSL Protocol, Version 3.0.* Netscape Communications Corporation, Mountain View, CA, March 1996.
3. S. Kent and R. Atkinson, *RFC 2401: Security Architecture for the Internet Protocol.* Corporation for National Research Initiatives, Internet Engineering Task Force, Network Working Group, Reston, VA, November 1998.
4. Security Technology Group, Information Technology Laboratory, National Insitute of Standards and Technology, *NIST's Efficiency Testing for Round1 AES Candidates*, Second AES Candidate Conference (AES2), (Gaithersburg, MD), March 1999.
5. E. Biham, "A fast new DES implementation in software," in *Fast Software Encryption. 4th International Workshop, FSE'97 Proceedings*, pp. 260–272, Springer-Verlag, (Haifa, Israel), 1997.
6. B. Schneier, *Applied Cryptography*, Wiley & Sons, 2nd ed., 1995.
7. R. Doud, *Electronic Engineering Times*, ch. Hardware crypto solutions boost VPN, pp. 57–64. CMP Media Inc., Manhasset, N.Y., April 1999. Issue 1056.
8. D. Connor, "Reconfigurable logic. Hardware speed with software flexibility," in *Australian-Electronics-Engineering*, vol. 29, no. 8, pp. 260–272, 1996.
9. S. Brown and J. Rose, "FPGA and CPLD architectures: A tutorial," in *IEEE Design & Test of Computers*, vol. 13, no. 2, pp. 42–57, 1996.
10. J.-P. Kaps and C. Paar, "Fast DES implementations for FPGAs and its application to a universal key-search machine," in *5th Annual Workshop on Selected Areas in Cryptography (SAC '98)*, 1998.
11. E. Mosanya and et al., "Cryptobooster: A reconfigurable and modular crytographic coprocessor," in *Proceedings: Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, (Worcester, MA), August 1999.
12. R. R. Taylor and S. C. Goldstein, "A high-performance flexible architecture for cryptography," in *Proceedings: Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, (Worcester, MA), August 1999.
13. A. Elbirt, "An FPGA implementation and performance evaluation of the CAST-256 block cipher," technical report, CRIS Group, ECE Dept., Worcester Polytechnic Institute, Worcester, MA, May 1999.
14. B. Gladman, "Implementation experience with AES candidate algorithms," in *Proceedings: Second AES Candidate Conference (AES2)*, (Gaithersburg, MD), March 1999.
15. Xilinx, Inc., San Jose, California, USA, *The Programmable Logic Data Book*, 1999.
16. Altera Corporation, San Jose, CA, *Data Book*, 1999.
17. Lucent Technologies, Allentown, PA, *Field Programmable Gate Arrays Data Book*, 1999.