



FPGA-based Cryptography for Internet Security*

Viktor K. Prasanna and Andreas Dandalis

{prasanna, dandalis} @halcyon.usc.edu

Department of EE-Systems

University of Southern California

3740 McClintock Ave, EEB 200C

Los Angeles, CA 90089-2562, USA.

Introduction

The enormous advances in network technology have resulted in an amazing potential for changing the way we communicate and do business over the Internet. However, for transmitting confidential data, the cost-effectiveness and globalism provided by the Internet are diminished by the main disadvantage of public networks: security risks. The significantly increasing growth in the confidential data traffic over the Internet makes the security issue a fundamental problem. Consequently, applications such as electronic banking, electronic commerce, and Virtual Private Networks (VPNs) require an efficient and cost-effective way to address the security threats over public networks.

Cryptography is the fundamental component for securing the Internet traffic. However, cryptographic algorithms impose tremendous processing power demands that can be a bottleneck in high-speed networks. The implementation of a cryptographic algorithm must achieve high processing rate to fully utilize the available network bandwidth. To follow the variety and the rapid changes in algorithms and standards, a cryptographic implementation must also support different algorithms and be upgradeable in field. Otherwise, interoperability among different systems is prohibited and any upgrade results in excessive cost. The ultimate solution for the problem would be an adaptive processor that can provide software-like flexibility with hardware-like performance.

1 The Internet Protocol Security Standard (IPSec)

For securing the Internet traffic, Internet Protocol Security (IPSec) standard was developed by the Internet Engineering Task Force. The IPSec standard [4] extends the IP protocol by securing the IP traffic at the IP level using cryptographic methods. IPSec has been adopted by all leading vendors and will be the future standard for secure communications over the Internet. It is also rapidly becoming the industry standard for VPNs [3]. The remarkable potential of the VPN market is an evidence of the significance of IPSec in the global electronic business. From US\$ 224 million in the year 1998, the VPN market is forecast to grow to US\$13,000 million in the year 2004 [3].

The IPSec standard is a framework of open standards and its major advantage is flexibility. IPSec is independent of any cryptographic method by providing an open framework to implement any cryptographic algorithm. The cryptographic methods to be used are negotiated

* This research was performed as part of the MAARCII project. This work is supported by the DARPA Adaptive Computing Systems program under contract no. DABT63-99-1-0004 monitored by Fort Huachuca.

between two communicating entities. As a result, the IPsec standard makes it possible for security systems developed by different vendors to interoperate.

Initially the two communicating entities negotiate and establish a security association for protecting the transferred data [4]. A security association determines the cryptographic methods and the related keys to be utilized. The cryptographic methods include both private and public-key cryptography, keyed-hash algorithms, and digital certificates. Each security association is restricted by its lifetime, after the expiration of which, a new security association has to be established. The lifetime of a security association can be determined in terms of absolute time or amount of transmitted data. Usually, a small amount of data is processed per key and the key-context switching occurs repeatedly.

In this paper we focus on the encryption component of IPsec. Encryption implementations for IPsec have to meet the enormous computing demands of cryptographic algorithms combined with the processing rate demands of high-speed networks. Since the security parameters are dynamically negotiated by the communicating entities, such implementations also have to be flexible enough to adapt to diverse security parameters. Known software-driven solutions can provide the required flexibility, but they are inadequate for data-transfer rates higher than those found in two T1 lines [6]. On the other hand, ASIC-based architectures can provide significant performance advantages, but are restricted by the design parameters during fabrication. Thus, any update of specialized encryption/decryption chips in commercial routers (e.g. Cisco, AT&T, Lucent) becomes very costly [3]. The ultimate solution for IPsec cryptography would be an adaptive cryptographic processor that can provide hardware-like performance with software-like flexibility.

2 Field-Programmable Gate Arrays (FPGAs)

FPGA technology is a growing area of research that has the potential to provide the performance benefits of ASICs and the flexibility of processors. Application specific hardware circuits can be created on demand to meet the computing and interconnect requirements of an application.

The basic feature underlying FPGAs is the programmable logic cell, which is realized by either using anti-fuse technology or SRAM-controlled transistors. FPGAs have a matrix of logic cells overlaid with a network of wires. Advanced architectures also include embedded memory blocks and multipliers. The computation performed by the logic cells and the connections between the wires can both be configured. A logic cell usually consists of look-up-tables (LUTs), carry logic, flip-flops, and programmable multiplexers. The multiplexers are utilized to form data-paths inside the logic cell and to connect the logic cells with the interconnection resources. The interconnection resources consist of nearest-neighbors and non-local wires that are controlled by programmable switches. Similar to the logic cells, the switches are realized by either using anti-fuse technology or by using SRAM-controlled transistors.

Current devices mainly use SRAM to control the configurations of the logic cells and the interconnection network. The configuration of a device can be modified by loading a stream of bits onto its configuration memory. The size of a configuration bit-stream can range from several Kbits to several Mbits of data depending on the size of the corresponding FPGA. Furthermore, FPGAs can be reconfigured very quickly, allowing their configuration to be altered according to the requirements of a computation. The complete device or a part of it can be reconfigured at runtime or before the computation commences. Depending on the frequency of altering the configuration, FPGA-based solutions span from implementations that are occasionally upgraded in field to implementations that evolve during computation.

The evolution of the fabrication technology has led to the development of FPGAs with exceptional computational power at a lower cost. Existing state-of-the-art FPGAs consists of two million gates and the next-generation FPGAs will consist of approximately ten million gates [7]. Finally, the cost, in terms of computational density as well as computational power, is rapidly decreasing making the FPGAs a cost-effective solution. Clearly, the density of FPGAs is growing faster than that of any general-purpose microprocessor.

3 Adaptive Cryptographic Engine (ACE)

Based on FPGA technology, we propose an Adaptive Cryptographic Engine (ACE) that can provide the speed and the flexibility required by IPsec. ACE consists of an FPGA device, a cryptographic library, and a configuration controller (Figure 1). The FPGA is configured on the fly by the configuration controller. Subsequently, adaptation to the input key occurs and the data encryption/decryption commences. The configuration controller determines the configuration to be chosen based on the requested security association.

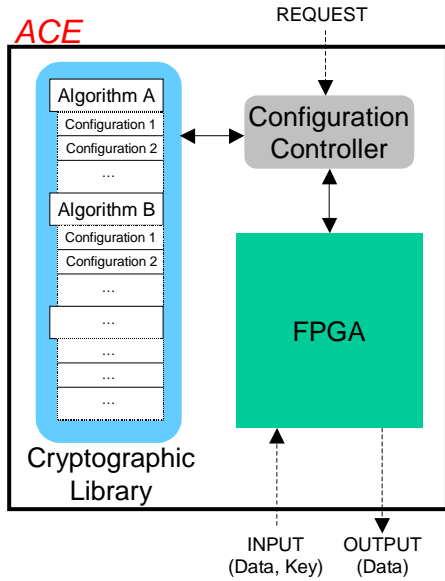


Figure 1 ACE Overview

The core of ACE is FPGA technology. FPGA-based implementations can provide the high processing rates required by high-speed networks. Moreover, FPGAs can be reconfigured on the fly providing the high degree of flexibility required in dynamically changing environments. The cryptographic library consists of FPGA configurations of private-key cryptographic algorithms. Each cryptographic algorithm can correspond to various configurations. Such algorithm variants differ from each other in terms of the cryptographic operation mode, the key length, the number of rounds, etc. The configurations are stored in memory and can be updated to expand the lifetime of ACE. The configuration controller is indispensable for configuring an FPGA. Besides being the interface between the cryptographic library and the FPGA, it also resolves the external requests (e.g. runtime security parameters).

A domain-specific approach is utilized to design the cryptographic library. The domain is defined by the algorithm and the target FPGA architecture. Each variant of the algorithm corresponds to a different sub-domain. Based on a specific domain, algorithm-specific configurations are derived. The cryptographic library is derived off-line while the configuration to be downloaded onto the FPGA is synthesized on the fly. Our key idea is to incrementally synthesize configurations at runtime by merging a skeleton with a sub-domain configuration. A skeleton is the intersection of all the configurations in a domain, that is, a parameterized configuration that corresponds to the elementary functions of an algorithm. Thus, a sub-domain corresponds to the modifications of the skeleton that lead to a configuration of a variant of an algorithm. As a result, the configurations of a domain can be efficiently represented while avoiding data redundancy, that is, the skeleton configuration data.

3.1 Performance Evaluation

The significance of the Advanced Encryption Standard (AES) [1] led us to choose the final AES candidates (MARS, RC6, Rijndael, Serpent, Twofish) as the base for the cryptographic library. By providing precise time-performance results, we can evaluate the practical

effectiveness of ACE. The AES algorithm(s) is a private-key cryptographic algorithm that supports at minimum block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits [1]. It will replace the aging Data Encryption Standard (DES). DES was adopted by NIST in 1977 as a Federal Information Processing Standard used by federal agencies and the private sector to encrypt information. Having chosen the final candidate algorithms, the AES development effort has entered its final stage where the final algorithm(s) will be chosen.

Among the various time-space tradeoffs, we focused primarily on time performance. The time performance metrics are the key-setup latency and the throughput. The key-setup latency time and the throughput are associated with the key-agility and the bulk-encryption efficiency respectively. We have exploited the inherent parallelism of each cryptographic core and the low-level hardware features of FPGAs to enhance the performance.

We have implemented “single-round” encryption implementations of 128-bit data blocks and key size. Our goal was to maximize the throughput for the cryptographic core of each candidate algorithm. Moreover, the key-setup latency issue was of primary interest, that is, the cryptographic core had to commence as early as possible. Based on the achieved throughput, we designed the key-setup component to sustain the processing rate of the cryptographic core and to achieve minimal latency. Even if an algorithm does not support on-the-fly key generation (in the software domain), the key setup can be executed concurrently with the cryptographic core. Since one round is implemented and reused repeatedly, the throughput results correspond to $128/(n \cdot t_{\text{round}})$, where n and t_{round} are the number of required rounds and the encryption time per round respectively. Similar performance analysis can be performed for larger sizes of data blocks and keys as well as for implementations that process multiple blocks of data concurrently.

As a hardware target for the proposed implementations, we have chosen the Xilinx Virtex family of FPGAs [7]. For mapping onto Virtex devices, we used the Foundation Series v2.1i software development tool [7]. All the results were based on placed-and-routed implementations that included both the key-setup component and the cryptographic core along with their control circuit. Our performance results are compared with the software-based results of the NIST Efficiency Test for Round 1 AES Candidates [1]. The reference platform for that efficiency testing was a Pentium Pro with 64 MB RAM running at 200 MHz.

3.1.1 Key-setup latency time

The key-setup latency time is the time required by an implementation to derive the key-dependent data required (e.g. sub-keys) to commence encryption. Our FPGA implementations achieve significant reduction in the key-setup latency time by a factor of 20-700. On the contrary, the key-setup latency time of the software implementations [1] is equal to the time for encrypting 3-13 blocks of data. In FPGAs, each cryptographic round can commence as early as possible since the key-setup process can run concurrently with the cryptographic core. In the case of software implementations, the cryptographic core cannot commence before the key-setup process is completed for all the rounds. Thus, while FPGA implementations favor agile key-context switching, the software implementations require relatively long time for key-context switching.

3.1.2 Throughput

The throughput corresponds to the amount of data encrypted per time unit. Our FPGA implementations achieve throughput improvements of 4-20 times compared with the software-based results. While the software implementations do not achieve processing rates higher than 30 Mbits/sec [1], our FPGA implementations achieve processing rates higher than 100 Mbits/sec. For one thing, software implementations cannot exploit the inherent parallelism of a cryptographic round. For another, the operations required by each cryptographic round can be executed more efficiently in FPGAs than in a general-purpose computer. By using a superior platform configuration than the reference platform for the

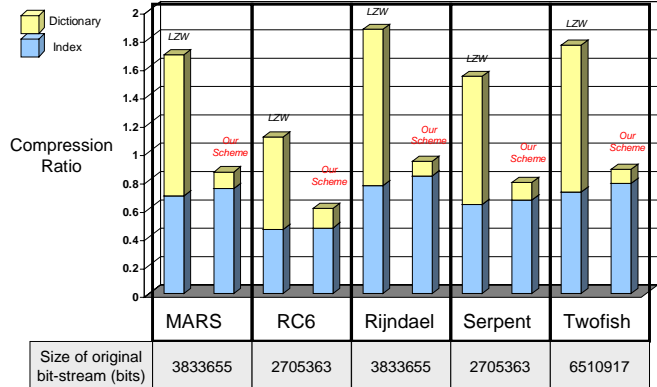
NIST Efficiency Test for Round 1 AES Candidates [1], higher throughput can be achieved for the software implementations. However, even in this case, the speed-up of the FPGA implementations would still be remarkable (i.e., Rijndael, Serpent). By realizing “multiple-round” FPGA implementations, the throughput speed-up can be significantly improved for operation modes that allow concurrent processing of multiple blocks of data.

3.2 Configuration Compression

The cost-effectiveness of the cryptographic library is strongly related to its memory requirements. Given the variety of the cryptographic algorithms and the corresponding configuration size, configuration compression can result in significant savings in memory. Usually, the configuration size is in the order of Mbits. The main idea is to store compressed configurations in the memory and perform decompression at runtime. Finally, the decompressed data is loaded to configure the FPGA. Our goal is to develop a lossless compression technique that leads to high decompression rates without the need for dedicated hardware resources.

Our compression technique is based on the principles of dictionary-based compression [5]. Even though statistical methods can achieve higher compression ratios, we preferred a dictionary-based approach because statistical methods require dedicated hardware resources for decompression. Decompression can be as simple as a look-up table operation. The main idea of dictionary-based approaches is to encode variable-length strings of symbols as single codewords. The codewords form an index to a phrase dictionary. In our scheme, the dictionary corresponds to the configuration data while an index corresponds to the way that a configuration is being synthesized. Similarly, decompression corresponds to configuration synthesis. For each cryptographic algorithm, we derive a dictionary and its index off-line. The decompression occurs at runtime by parsing the dictionary with respect to its index. As a result, only memory read operations are required and thus high decompression rates can be achieved.

We derive each dictionary based on the principles of the LZW algorithm [5]. As compression proceeds, the dictionary index is derived simultaneously with the dictionary. Compared with the conventional LZW, our technique utilizes an enhanced dictionary representation. Moreover, after having derived the dictionary and its index, we reduce the memory requirements of the dictionary by selectively decomposing phrases in the dictionary. The main idea is to replace frequently occurred strings among phrases by a new phrase. Another difference with conventional LZW algorithms is the calculation of the compression ratio. In conventional LZW algorithms, only the index that is stored in a secondary storage media or transmitted over the network is considered in the calculation of the compression ratio. There is no need to store or transmit the dictionary since it can be reconstructed on-line based on its index. However, in embedded environments like ACE, the dictionary size should also be considered in the calculation of the compression ratio.



The bit-streams of the AES algorithms were compressed utilizing our technique. Both the index and the dictionary memory requirements are considered in the calculation of the compression ratio. The compression ratio metric indicates the fraction of the compressed configuration over the original one. Our compression results are

Figure 2 Configuration Compression Results

compared with the results derived by using the LZW algorithm to generate the dictionary and its index.

In Figure 2, the compression ratios achieved by the LZW algorithm and our technique are shown. The blue and the yellow columns correspond to the fraction of the index and the dictionary respectively over the compressed configuration. At the lower part of the chart, the size of the original configuration for each AES algorithm is shown. The sizes of the dictionaries that were derived by LZW were comparable with the sizes of their indices. Therefore, negative compression occurred, that is, the memory requirements of the dictionary and its index were greater than the memory requirements of the original configuration. However, by using our technique, the dictionary sizes were reduced by a factor of 5-11 while the index sizes were slightly increased. As a result, 7-40 % savings in memory were achieved.

4 Conclusions

In this paper, an Adaptive Cryptographic Engine (ACE) was described which performs the encryption/decryption tasks required to secure the Internet traffic. ACE is an FPGA-based platform that can provide the speed and the flexibility required for IPSec architectures [4]. We considered the AES [1] final candidate algorithms as the base for our cryptographic library. Compared with software-based implementations [1], throughput speed-up of 4-20 was achieved while the key-setup latency time was reduced by a factor of 20-700. We also addressed the equally important issue of compressing the configuration bit-streams of the library. Though data compression has been extensively studied in the past, we are not aware of any prior work that addresses the compression problem of FPGA-based platforms on the basis of the implementation cost. Using our technique, we demonstrated up to 40% savings in memory for various configuration bit-streams.

The work reported here is part of the USC MAARCII project (<http://maarcII.usc.edu>). This project is developing novel mapping techniques to exploit dynamic reconfiguration and facilitate run-time mapping using configurable computing devices and architectures. A domain-specific mapping approach is being developed to support instance-dependent mapping. Moreover, computational models and algorithmic techniques based on these models are being developed to exploit self-reconfiguration using FPGAs. Finally, the idea of "active" libraries is exploited to develop a framework for automatic dynamic reconfiguration.

References

- [1] Advanced Encryption Standard, <http://www.nist.gov/aes/>
- [2] Dandalis, A. (Under Preparation) *Dynamic Logic Synthesis for Reconfigurable Devices*. PhD Thesis, University of Southern California.
- [3] Fowler, D. (1999) *Virtual Private Networks. Making the Right Connection*. Morgan Kaufmann Publishers, Inc.
- [4] IPSec Developers Forum, *What is IPSec?* http://www-ipsec.com/IPSec_info.html
- [5] Nelson, M. (1996) *The Data Compression Book*. M&T Books
- [6] Robinson, B. (1999) *Plans for a Secure Future*. Tele.com magazine
- [7] Virtex Series FPGAs, <http://www.xilinx.com/products/virtex.htm>