Design of Secure Hardware for ATM Networks

Engr. JUNAID MAJEED
Faculty of Engineering Science and Technology
Hamdard University
Karachi Pakistan

Abstract:

Broadband networks based on the asynchronous transfer mode (ATM) are emerging rapidly. Both the technological component in terms of ATM infrastructure, as well as the area of applications requiring Quality of Services (QoS) by the means of bandwidth or delay constraints is covered by a variety of projects and products. However, given the increasing interest in applications such as governmental communication, transmission of medical information, or commercial applications, the necessity of providing secure means of delivering sensitive contents is apparent. In this paper, we focus on security services in ATM network. .Data security plays an increasingly important role in today's information technology. Potential data rates in the gigabit ranges such as offered by ATM networks, put many constraints on the design of a secure, but usable, network. In addition, the cell structure of ATM makes bulk data encryption as well as public key security services challenging tasks. This paper deals with agility of cryptographic algorithms that is the capability of an encryption device to change its algorithm. This feature appears to be very desirable for high speed networks because it facilitates design flexibility and future protocol additions and changes. We propose the use of reconfigurable hardware since they appear to be naturally suited for the task. The use of reconfigurable in cryptographic applications to our knowledge has not been systematically analyzed before and appears to be a highly interesting area within high speed network security.

Introduction

Adding security through cryptography to any system is almost never a trivial task. One must carefully weigh the issues when deciding how much or how little security will be provided. Obtaining security services through strong cryptography may require so much bandwidth and processing overhead or such a high production cost that the system would become infeasible for real world applications. On the other hand, providing a lower level of services may have better performance, but the security may be too weak. The ideal is to find a middle ground which satisfies security needs with a reasonable throughput cost/ ratio. Today's most relevant networking topics are high speed networks and wireless technology, both of which have separate issues with cryptography. On one hand, we need encryption rates fast enough to sustain high speed data streams. On the other hand, we need low bandwidth / low power consumption solutions for wireless operations.

ATM falls into the high speed category. It is difficult to design a universal solution to the ATM security problem, since so much of the design depends on the through put requirements of the network. ATM by nature does not specify the physical layer which will be used to transfer ATM cells. We therefore need some kind of scaleable architecture that can be implemented in today's ATM technology, but can adapt to tomorrow's with little effort or change in protocols. The mainstream developments with ATM support 155Mb/s and 622 Mb/s. The question is what cryptographic element is needed to support security services at these speeds. Unfortunately, software is too slow. A software implementation of a common private key block cipher (which are relatively fast encryptors) allows throughputs of 1- 10Mb/s [19]. Hardware versions typically run 3-4 orders of magnitude faster.

1.1 Potential Threats

The first step to take when developing a secure system is to identify the potential threats against the system. After the threat analysis is complete, the security services used to thwart the attack can be applied more effectively. Lane [9] describes a list of threats in an ATM Network. They range from passive listening to actively destroying network service. The complete list can be found in Table 1.1.

Object	Threat	Example of Threat
Information Privacy	Disclosure	Passive Listening
Resource Availability	Denial of Service	Flooding Network
Integrity	Insertion, removal, or modification of data	Modifying ATM Headers to alter channel definition(misrouting)
Attached workstations	Intrusion	Unauthorized login
Network or application provider revenue	Fraud	Using false identity to obtain resources

Table 1.1 Network Threats

1.2 Security Services

There are many security services that the use of cryptography provides. Each one attempts to thwart one or more attacks from Table 1.1. This section serves as an outline to some features desirable in a secure ATM link.

1. Privacy
   - Definition: The ability to send information in a manner so that only the intended recipients have the ability to "see" the data.
   - Solution: Use an encryption algorithm. Issues of key management will need to be resolved.

2. Integrity
   - Definition: The process of verifying that a payload was not tampered with in transit.
   - Solution: Include a cryptographic checksum or Message Authentication Code (MAC) with the payload or use digital signatures.

3. Authentication
   - Definition: Authentication is the process of one node calculating the true identity of a remote node and verifying that the payload has integrity.
   - Solution: Use digital signatures and/or MAC codes.

4. Access Control

- Definition: The ability to control access to objects and resources based upon the identity or the current level of access granted to an entity.
- Solution: For simpler discretionary access control, a password accessed system can be used. For mandatory access control, higher level labeling and compartmenting should be used. Access Control Lists (ACL) are usually kept to govern the access privileges of entities and objects.

5. Replay Prevention
   - Definition: Preventing an opponent from resending a once valid payload at a later time.
   - Solution: Include a (secured) timestamp with the payload. If the packet arrives at a time interval greater than the local security policy allows, discard the cell and/or update audit trail.
6. Non repudiation
   - Definition: The ability to prove the absolute identity of a payload.
   - Solution: Use public key signatures, Private Key signatures require that the two (or more) parties involved share a secret. If party A sends a private key signed packet to B. A can later cheat and claim that B sent the packet. However, public key signatures would allow B to prove that only A knows how to generate a given signature.

2. Background

2.1 Encryption Hardware

What encryption hardware can support these >100Mb/s throughput rates available in ATM links. There are currently two major forms of hardware suitable for cryptography, custom Application Specific Integrated Circuits (ASIC) or Reconfigurable Hardware i.e. Erasable Programmable Logic Devices / Field Programmable Gate Arrays (EPLD/ FPGA).

2.1.1 ASIC(s)

ASICs have some advantages over Reconfigurable (RC) hardware. First, they are usually faster, since they were designed specifically for the problem, whereas RCs are generic logic devices that are programmed using variable switching matrices and configurable logic elements. These switching matrices add variable delay due to increased parasitic capacitance and resistance of each switch, and the logic elements may sometimes exhibit poorer performance when compared to the gate implemented in custom silicon. Second, ASICs are usually smaller and consume less power because RCs have overhead logic for maintaining the reprogram able circuitry. However, ASICs cannot offer the flexibility of a reconfigurable device.

2.1.2 Reconfigurable(s)

Aside from the points made above, Reconfigurables RC have several major advantages over a custom ASIC design.

1. Algorithm Agility: Because the device can be reconfigured, it allows a designer to simply reprogram the device when a new algorithm is needed. With ASICs, a separate device must standby until it is needed.
2. Shorter design time and verification: The development time of RC hardware is significantly shorter that a full custom solution because the designs can be verified quickly without waiting for manufacturing delays.
3. Design changes are easily accommodated: If an algorithm changes in the future, the binary image can be distributed among the devices in the network allowing “hardware” upgrades to be done without actually changing any hardware.

Reconfigurables would appear to be the ideal choice for implementing cryptographic elements. However, there are some major problems. Aside from the points above regarding the inherent delay/power-consumption/size problems, it is unclear whether RC hardware can accommodate cryptographic applications. Up until recently, these devices have been extremely small and could only replace a few thousand equivalent gates. A modern link encryptor may consume tens (or hundreds) of thousands of gates. In addition the I/O resources required supporting ATM cells and encryption is fairly significant and may have problems mapping to a device. The speed problem could become an issue as the link of the ATM network increases as well. How many RC devices are needed? An array of RCs may exhibit enough throughputs to sustain an OC-12(approx. 622Mb/s) rate, but how many chips must run in parallel to achieve this. The most likely location for encryption will be in the ATM layer itself, before the data leaves a node. This means that all the encryption hardware must be on the local Network Interface Card (NIC). The card itself has size restraints and power consumption-cooling requirements. Will laptops using PCMCIA cards ever are able to enjoy secure ATM? If they do, it will most likely be under an ASIC control or an external device. These topics need further evaluation.

2.2 Symmetric Algorithms

Lane and Cohen [10] acknowledge that there are many symmetric algorithms that can be used effectively with ATM. A general set of criteria can be established to test an algorithm for eligibility:

1. The algorithm block size should be able to divide evenly into 384 bits the ATM (payload size). This allows for greater efficiency.
2. The block size should be relatively large (>=64 bits) so that small patterns in the plaintext do not generate cipher text that is easy to perform “cipher text substitution” attacks.
3. The combination of (1) and (2) limit the block sizes to be 64, 96,128 or 384 bits.
4. The algorithm should have at least the strength of DES [23], but preferably higher in order to provide long term security. This includes both the key length, and level of immunity to linear ([12],[13],[14])and differential ([2],[3]) cryptanalysis.
5. The algorithm should be easily implemented in hardware with either direct support of ATM speeds (45-622 Mb/s) or provisions for parallel execution to sustain these rates.
6. It should include provisions for key agility on a per-cell basis.
7. Details of the algorithm should be publicly available.

Table 2.1 is a list of algorithms which meet the criteria. All items have been derived from [10] , [19] and [22].

Algorithm	Block size	Key Length	Security	Speed
DES	64	56	Baseline	Baseline

Triple DES[15]	64	112	>> DES	1/3 DES
DESX[18]	64	56+64	>DES	=DES
RC2[17]	64	variable	variable	>DES
RC5[16]	variable	variable	variable	variable
IDEA[8]	64	128	>>DES	>DES
CA-1.1[7]	384	64+1024	unknown	
CAST[1]	64	64	>=DES	unknown
SAFER[11]	64	64	unknown	>DES
LOKI[4]	64	64	>=DES	unknown
3-Way[5]	96	96	Unknown	>DES

Table 2.1 Crypto algorithms suitable for ATM

Currently the ATM Forum is discussing which algorithm will become the standard. One problem holding the decision back is the US Government's restriction on exporting cryptography. The current law considers cryptography munitions and limits exportable encryption devices to 40 bits. This restriction is a matter of intense controversy. There are several policy proposals pending which would either increase the allowed bit length, drop the current restriction altogether, or would call for key escrow/recovery mechanisms.

2.3 Mode of Operation

A cryptographic algorithm usually provides only the core functionality of data encryption. Several methods or modes of operation exist to allow the customization needed for a particular application. For instance, one mode of operation, called Electronic Code Book (ECB) may be used to simply encrypt/decrypt in straight blocks without any feedback or additional operations. This creates a one-to-one mapping between plaintext and ciphertext. Other times, the previous input or output will actually change the next output. These are referred to as feedback modes. Feedback modes serve to randomize the output (thus producing a more non-deterministic output) and to make it harder to modify any given block of cipher text [19].

2.4 Interleaving

When the available hardware crypto chips are not fast enough to sustain the desired rate (say 622Mb/s) they must be interleaved (or run in parallel). For instance, if a given chip runs with 64 bit blocks at 100Mb/s, and we want to encrypt at the ATM layer would need

$$[(48/53*622\text{Mb})/100] = [5.63] = 6$$

chips to sustain an OC-12 line. Note the ratio 48/53 in the formula compensates for payload/cell-size differences.

There are implications involved when using chips in parallel because it is necessary to generate new IVs for each chip in some modes of operation.

2.5 Key Storage

A typical modern day cryptographic system uses two types of cryptographic functions to establish a secure connection, public-key and private key.

Public-key encryption allows users to publish a key, which any node can obtain in order to send encrypted information to the issuing party. Only the publisher of the key is able to decrypt the encrypted message with a private-key which is mathematically linked to the published key. Conversely, public-key signatures allow a user to publish a verification key, which any node can verify the signature of a block, but only the publisher can produce a valid signature. The advantage of public-key is that information, such as a private-key does not need to be shared among users.

The disadvantage is that the algorithms are inherently slow compared to private-key algorithms and the keys need to be very large (768-1024 bits are common)

Private Key algorithms require users to share a common key between the two (or more) communicating nodes. The advantages of private-key include: fast encryption and shorter key lengths. The disadvantages include users must share a secret and keys must be transferred over a secure channel.

By using a combination of the two a designer can create a secure channel using public-key cryptography, negotiate "session key" with the remote party and continue the rest of the communications with the speed and agility of private key algorithms. Also security services such as data integrity and sender authentication are often achieved through public key digital signatures. This is what is known as a hybrid scheme, because it uses the advantages of multiple types of algorithms to produce a robust security system with good performance.

2.6 Key Agility

Key Agility defines the ability of a system to switch cryptographic keys. A security device that uses a single key for all communications would be considered non-key agile. A system that can switch keys on a per connection basis would be considered highly agile. In almost all scenarios, it is desirable to enable a separate cryptographically isolated channel on each virtual circuit. In a worst case scenario for a key agile system every cell arriving would originate from a unique VC. An ATM system that incorporates a technology with data rate x in the physical layer must be able to handle 424 bits/cell/x if it is to support a new key for every incoming cell. For instance if x is derived from OC-12 SONET, the transmission speed is 622Mb/sec, yielding

$$424/622*10^6 = 681\text{ns}$$

This means that a new key (and initialization vector, if required) must be referenced and loaded in less than 681ns (minus the decryption time). This can have a significant impact on the overall design of the crypto unit. Agility can be realized by limiting the number of secure channels (thereby reducing the memory requirements) and to pipeline the crypto unit so that keys may be loaded before they are scheduled for decryption-encryption. [20] uses an address hash table and limits the secure connections to 2^{16}.

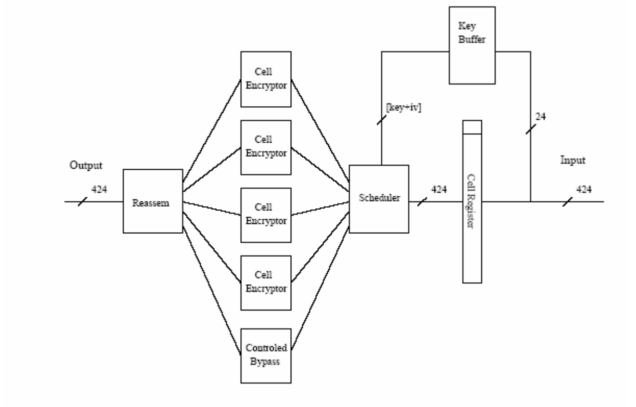

Figure 2.1 The Key Agile Architecture

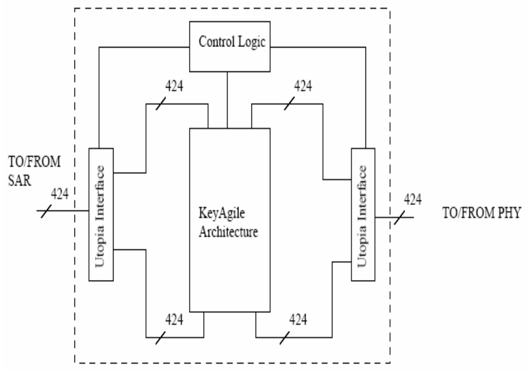

Figure 2.2 The General Security Architecture

The general layout of the components is given in Figure 2.1. The idea is to use enough encryption hardware in parallel to sustain the link speed.

2.7 Overall Layout
The module described in Figure 2.1 is the heart of the general architecture presented in Figure 2.2 which demonstrates the placement of the hardware with respect to the utopia bus. This design allows traffic flowing through a bi-directional bus to use a single bank of security hardware, rather than having separate devices.
Often times the hardware modules will not be local to one device, but rather spread out into separate modules. This allows larger memory and faster encryption hardware to be integrated. The layout may resemble Figure 2.3

2.8 Architecture Description
The Input Buffer
The input buffers job is to essential queue one single cell (424 bits + control information) while the memory unit is accessed. In the event that the memory unit is fast enough, this step is unnecessary. However, as we pointed out in earlier sections, as the speed of the network increases, this stage will become more and more important. If the network speed actually increases so high that the single buffer is not enough delay to accommodate the latency of the memory unit, additional units can be added as long as the memory unit has additional I/O ports.
The Cipher Array
The cipher array is a simple parallel configuration of two main components: The encryption block and a controlled bypass unit. Each block accepts one cell plus control data and will process that cell in a fixed time unit. The processing that occurs is essentially encrypting (or decrypting) the user payload of the cell (in the case of the encryption block) or simply passing the data through after the fixed amount of time has expired (in the case of the controlled bypass unit).

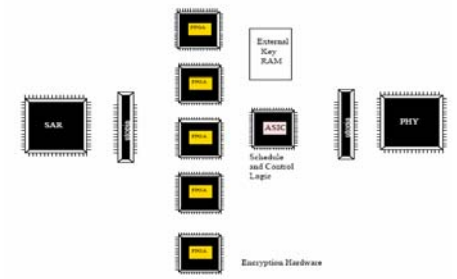

Figure 2.3 Model Layout

2.9 Algorithm Agility
Algorithm agility is the ability to switch cryptographic algorithms. An implementation that has one algorithm in hardware would be non agile. An implementation that could switch crypto routines on a per-cell basis would be highly agile. Algorithm agility is easy to accomplish if encryption is performed in software. However, ATM speeds dictate that hardware approaches must be used. Hardware algorithm agility can either be realized through providing all algorithms of interest on an ASIC, or to use reprogrammable logic (FPGAs/ EPLDs). One problem with the latter approach is that achievable data rates might be too slow for ATM. With today's technology, it is not possible to reprogram the chip on a per cell basis. In fact, it takes orders of magnitude more time to program the chip as compared to the cell arrival rate. The FPGA method is good for allowing the flexibility of the protocol design, without allowing algorithm agility per cell.
If the algorithms fit onto a feasible amount of ASIC chips, the ASIC approach offers a fast throughput to size ratio and may offer better overall performance in the ATM network.
The ATM Forum is designing the system to allow algorithm type/version information to be exchanged at secure call setup time. This allows the two remote hosts to guarantee that both parties are using the same security devices.

3. Implementation
3.1 Introduction
The next step in our investigation is to determine which device architecture works best with which type of cryptographic algorithms. We need both speed and efficiency for a wide variety of algorithm types. Some algorithms will use a very wide data path, while others will need many registers and flip flops. It is unclear which device architecture will perform adequately in an ATM type environment so we must design a methodology that will allow us to predict which one will.
The task of assessing all cryptographic algorithm performances in all RC hardware is an extremely complex task. To overcome this problem, we decided to analyze the algorithms for their components (such as XOR, ADD, SHIFT, etc.) and then run extensive tests on those components based on certain classes of hardware. Through this research:

- We can derive general statements about cryptography on RC hardware.
- Our findings can be extended to future algorithms, as they are proposed.
- We could gain some insight into which architecture might work acceptably well in an ATM environment.

There are many different branches of cryptographic algorithms, such as private-key ciphers, public-key ciphers and hash functions. For the purpose of ATM, we are only concerned with symmetric block ciphers, since they seem more promising for the use in the encryption unit. We start by explaining how symmetric encryption is accomplished.

3.2 Methodology

By studying existing algorithms see Table 3.1 we have determined that there is a small finite group of components that is common to all algorithms. Some of these components may have been discussed above regarding design theory, etc

Algorithm	Parameters	Components
DES	64 bit I/O, 56 bit key	P-BOX,XOR,SBOX,ROT
MADRYGA	var I/O, var key	XOR, SFT,ROT
NewDES	64 bit I/O, 120 bit key	XOR, N/A f box
FEAL	64 bit I/O, 64 bit key	XOR, ROT
REDOC II	80 bit I/O, 160 bit key	P-BOX,SBOX,XOR
REDOC III	var key up to 20 kbits	XOR,STOR
LOKI	64 bit I/O, 64 bit key	XOR,SBOX,P-PBOX
KHUFU	64 bit I/O, 512 bit key	XOR,Dyn-SBOX
KHAFRE	64 bit I/O, 64-128 bit key	XOR,SBOX
IDEA	64 bit I/O, 128 bit key	XOR,ADDER,MULT
MMB	128 bit I/O, 128 bit key	XOR,MULT,STOR
GOST	64 bit I/O, 256 bit key	SBOX,ROT,ADDER
CAST	64 bit I/O, 64 bit key	XOR,SBOX
BlowFish	64 bit I/O, 0-448 bit key	XOR,SBOX,ADDER
SAFER	64 bit I/O, 64 bit key	XOR, ADDER,ROT,GFMULT
3-Way	96 bit I/O, 96 bit key	XOR , P-BOX, ROT
CRAB	1024 bytes I/O, 128 bit key	P-BOX, XOR,AND, OR, NOT

Table 3.1 The available algorithms and their component breakdown

3.3 Component Breakdown

Table3.2 summarizes what we found in the analysis. Each component must be mapped into hardware, but we need a method that will yield accurate results across all platforms.

Component	Type
XOR/AND/OR/NOT	Boolean Logic
SBOX	Substitution Box
SFT/ROT	Shift/ Rotate Element
STOR	Storage Element
ADDER	Modulo Addition
MULT	Modulo Multiplication
PBOX	Permutation Box
GFMULT	GF(2^n) MULT[1]

Table 3.2 Component Description

3.4 Implementation

The next step is to build behavioral models using a hardware description language (HDL). Using these models we can map the given components into various architectures and record the results. HDLs were chosen for their portability across platforms. The advantage is that the differences in entry style can be factored out of the overall equations, because the resulting design is derived from the same source code. After the models are built, they were synthesized and mapped using the tools appropriate for the device. Various optimizations were selected over multiple runs to average the results. The most interesting results are the % resources consumed and the critical path delay.

Device Selection

In order to determine the final results of our experiments, two stages of processing were needed. The first is the synthesis stage, which compiles the HDL into device specific logic maps or netlists. The second is a place and route stage where the netlists are mapped into actual hardware entities. Because of the availability of these tools, we were only able to work with two vendors, namely Xilinx Corporation and Altera Corporation. Fortunately, these two vendors are the market leaders and also provide some of the largest devices for us to work with.

Unless otherwise noted all implementations of the algorithms were performed on speed grades 3. Devices are selected based on their availability in the market place and their relative size/speed/price/package selection. For example, XILINX has a multitude of devices in various families. However, only the XC4000 family is large enough to accommodate the large scale design of a cryptographic algorithm (typically 20K- 60K gates), so it is the only one used here. The same holds for Altera, where only the FLEX10K devices can support the needs of our application.

Initial research has revealed an average PAD delay for each device tested, which is always subtracted from the critical path delay in cases where clocking was not used to determine component speed. The results of this calculation are in Table 3.3

Device	Input delay(ns)	Output delay(ns)	Total
EPF10K70RC240-3	5.6	5.3	10.9
XC4020EPG223-3	2.5	8.5	11.0

Table 3.3 PAD delays in experimental hardware

3.5 Component Description

3.5.1 Permutation Boxes

Permutation boxes are diffusion elements that are easily implemented in hardware. Because the operation is essentially a remaping of input pins to output pins, the synthesis of such a circuit utilizes very few resources in RC hardware if sufficient routing resources are available. In cases where drivers are not needed, the realization of such a circuit only changes the pin mapping of the components connected to it.

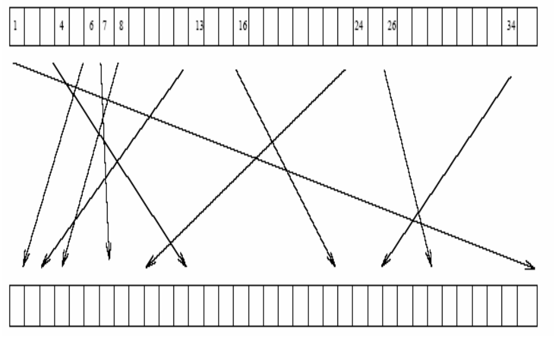

Fig 3.1 Permutation Box

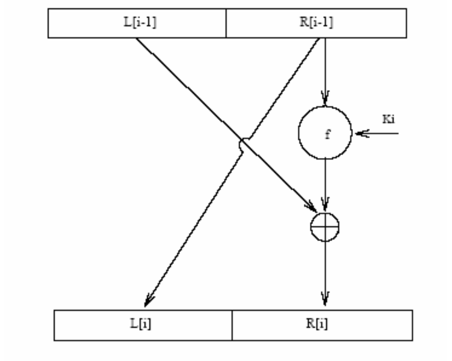

Fig.3.2 Feistel Network

3.5.1.1 Feistel Networks

Many block ciphers use Feistel network architecture, named after the inventor, Horst Feistel, who made major contributions to the field in the 1960s and 1970s while working on ciphers for IBM Research. The Feistel network is simple in concept. One of its most attractive features is that it allows for particular algorithms to be used for both encryption and decryption due to its inversion properties. The basic architecture is as follows. The datapath is split into two halves, the right and left sides. The right half is operated on by function f which incorporates elements of confusion, diffusion and key material and produces an output of the same size.

The output from the f function is XORed with the left half which is then stored into the right half. A copy of the original right half is swapped into the left half, thus completing the cycle (see Figure 3.2). It should be noted that only the left part L[i-1] is encrypted in one round, whereas R[i-1] is passed through in the clear.

3.5.2 Substitution Boxes

Substitution-Permutation networks or SP networks are elements that add hybrid confusion-diffusion to the input data. F-functions in Feistel networks are often based on them. Substitution elements take a *m* bit input and provide an n bit output (see Figure 3.3). The elements can be implemented as look-up tables or as combinatorial logic, but both are rather expensive so to minimize the cost, m and n are often kept small.

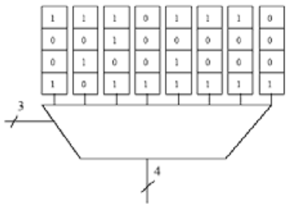

Fig 3.3 3x4 Substitution Box

Implementing these components can be rather expensive in reconfigurable logic because they are not typically tuned to look-up table type architecture. In fact, our studies have shown that s-boxes are often the largest component in a synthesized design, requiring tens or hundreds of logic elements to implement even small tables, such as a 6 x 4 which has 64 four bit values (256 bits each). Hardware architectures that support RAM and ROM components faired extremely well in these tests. It should be noted that in both examples that use ROM architecture, special design parameters were used to utilize the special hardware.

3.5.3 Shift-Rotate Registers

Shift and Rotate registers are commonly found in cryptographic applications. They are commonly found in the key scheduling logic and there are various architectures which work with different types of ciphers. Combinatorial shifters are the simplest to map into hardware because they are essentially a permutation. As was pointed out earlier, these permutations take hardly any hardware resources. The more advanced shifters, such as decisive and sequential shifters require more logic and will therefore be analyzed below.

Decisive shifters

Decisive shifters are components that take in two inputs. The first is the data word to be shifted, and the other is a binary value, allows the shifter to decide when to shift or not. All processing is done combinatorially and therefore doesn't require clocking or registered output. However, the decision circuitry requires a multiplexer so this unit is more than a simple permutation.

Sequential Shifters

Sequential shifters are components that register the input and shift based on a clock edge. They require the most amounts of hardware resources, but can offer the advantage of a register and a combinatorial shift in one component. For some designs this may offer the perfect element for key scheduling or round iteration processing.

Design	device	Optimization	LE Utilization	Max Delay
Sequential	FLEX10K	Speed	64	10.9ns
		Area	64	10.4ns
Decisive	FLEX10K	Speed	32	5.2ns
		Area	32	5.2ns
	XC4000E	Area	16	19.3ns
Combinatorial	N/A	N/A	N/A	N/A

Table 3.4. 32 Bit rotation box

Observe that the performance was equal across the two devices for the sequential shifter implementation, but dropped off significantly in the FPGA for the multiplexer based design.

3.5.4 Adders

There are various types of standard adder architectures which have various speed/area properties and are well suited to different types of hardware architecture. Rather than try to model and analyze each one in each different piece of hardware, we just used the built-in functions provided with the hardware vendor. For Altera, we used the LPM Module LPMADDSUB and for Xilinx, we used the XBLOX module ADDSUB. The results are in Table 3.5

Device	Size	Optimization	Utilization	Max Delay
EPF10K10TC144-3	32 bit	Area	63 LEs	102.4ns
		Speed	110 LEs	43.1ns
EPF10K10TC356-3	64bit	Area	127 LEs	196.0ns
		Speed	240 LEs	73.2ns
XC4020EPG223-3	32 bit	-	17 CLBs	24.1ns
	64 bit	-	33 CLBs	45.7ns

Table 3.5 a Adder in various hardware

3.6 Component Conclusion

This section has presented a description of some of the components found in cryptographic algorithms. In addition, the results of our analysis accompanied each description. It was shown that neither architecture analyzed has a distinct advantage over that other. One may excel in one area, while the other will excel in an unrelated area. Making a choice for a particular vendor is a question of the specific cryptographic algorithm and the cost of each chip verses the efficiency of the component placement.

3.7 The Data Encryption Standard

3.7.1 Introduction

The Data Encryption Standard (DES) is probably the most commonly used algorithm in the world for symmetric encryption of data. Especially important is the fact that the algorithm has been approved for use in ATM by the ATM Forum. For an explanation of DES,(see [6],[21, page 70]).We should be able to assess whether RC hardware is principally acceptable for use in high speed secure networks.

3.7.2 Design

DES uses Feistel Network architecture with 16 rounds. Each round can be implemented as separate hardware with pipe-line stages between each one for high throughput applications. However, this consumes major silicon real-estate and generally will not work in reconfigurable hardware because it is too large. For designs with less than 16 rounds of fixed hardware, some kind of feedback loop must be established.

When we set out to implement DES in reconfigurable logic for high speed networks, there was a set of design criteria that we wanted to meet.

- It must be targeted for high performance (as opposed to smallest size).
- It must complete an operation (such as encrypt or decrypt) in the fewest possible cycles (which is 16 for a simple design).
- It must fit into a commercially available chip (as opposed to one that is only in beta test).
- It must provide for loop unrolling for future speed improvements.

Figure 3.4 shows the layout of the components from a schematic point of view. Note the use of two 64bit registers: one on the inputs and another in the feedback loop. This design allows us to "steal" an extra clock cycle at the expense of 64 flip-flops and one gate more of complexity through the Feistel network (through the MUX). Without the secondary register at the data feed, we would be required to go to INIT state after the sixteenth round completed so that the outputs could stabilize to the correct data (DONE is asserted) before new data is read in, With this register in place, we can successfully read a new data set in at the conclusion of round 16 thus producing the cycle chain INIT, R1, R2,…, R16, R2,…. Etc. Without it the state transition diagram must conclude to INIT every time before starting the next set. This addition saves a single pulse width of latency and increases system throughput by 6.25%.

In addition to designing the main data path for high speed, the key scheduler must also be designed to deliver data at the same rate and with correct framing with respect to the data path. In order to do this we placed a single register to sample data coming from a multiplexer. The multiplexer is fed by both the feedback and the outside key input. The output from the registers feeds a unit chain of schedulers, one for each unrolled loop in the path. Each key unit schedules the keys for one round, receiving control information from its respective command line (stkeyunit0, stkeyunit1, stkeyunit2, etc.). The output from each key unit is fed to its respective Feistel network and the next unit in the chain. The last unit in the chain feeds its Feistel network and then loops the output back into the master key scheduler for storage in the registers. This operation is displayed in Figure 3.5.

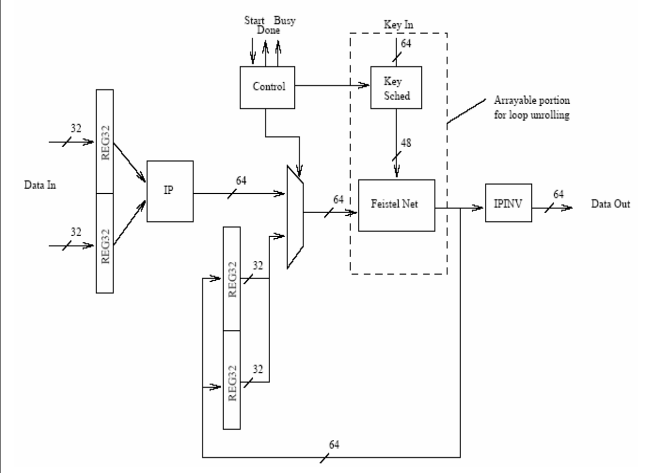

Fig 3.4 Schematic map of DES algorithm

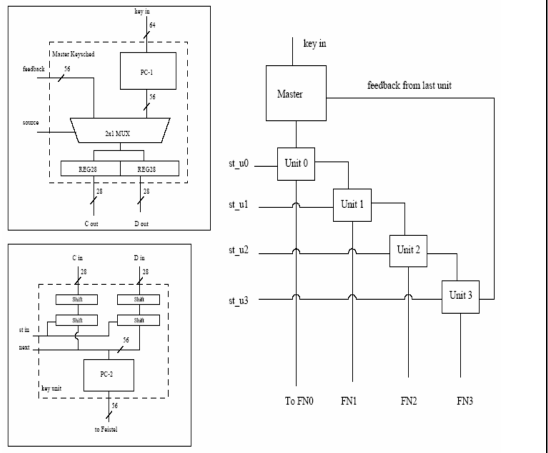

Fig 3.5 Schematic map of key schedule logic

3.8 Comparing the Results in Reconfigurable Hardware

In our experiments we compared two high end devices (manufactured by Xilinx and Altera) in a series of tests that we hope show the relative performance of cryptographic algorithms. It may be unclear which device actually performs better, and in actuality, they were very close. It is difficult to estimate the actual resources consumed in a device since there are so many discrepancies in the way information is provided by the companies. Often times the transformation between logic elements (LEs) and typical gate counts is overestimated and can confuse the user.

Component	Type	LEs	REs	LeW($)	ReW($)	% Resources	Speed(ns)
XOR	A	32	0	1.92	0	2.78	12.0
	B	16	16	8.16	8.16	2.04	10.4
1 SBOX(ROM)	A	0	512	0	87.04	133.33	18.3
	B	80	64	40.80	40.80	10.20	15.8
I SBOX(AREA)	A	113	0	6.78	0	9.81	49.6
	B	18	18	9.18	9.18	2.29	51.1
1 SBOX(SPEED)	A	119	0	7.14	0	10.22	31.0
	B	89	89	45.39	45.39	11.35	36.3
Shifter(DecArea)	A	32	0	1.92	0	2.78	5.2
	B	16	16	8.16	8.16	2.04	19.3
Adder(32bitArea)	A	63	0	3.78	0	5.47	102.4
	B	17	17	6.67	6.67	2.17	24.1
Adder(64bitArea)	A	127	0	7.62	0	11.02	196.0
	B	33	33	16.83	16.83	4.21	45.7
Buffer(256x32)	A	0	256	0	43.52	66.67	9.5
	B	360	256	186.6	130.56	45.92	60.6

Table 3.6Components Evaluated with Size Comparison

3.9 DES Comparison

After completing the design using VHDL modeling and synthesis tools, we realized an entire DES implementation. We also determined the following performance ratings. The results are posted in Table 3.7. For a maximum throughput, we measured 62.5Mb/s from the Xilinx device without a single loop unrolled. Close behind it was the Altera device which maxed out at 57.60Mb/s. One important difference, however, is that the Altera device cannot be unrolled any more because the memory EABs have been exhausted. From this point on, we will consider the designs that are ROM mapped only in the Xilinx devices because of the limitations in the Altera chips. So this yields comparatively 62.5Mb/s for Xilinx and 39.96Mb/s for Altera. Both of these designs support loop unrolling. We will now analyze these designs in the same manner as the individual components.

4. Conclusion

This research paper hopefully gave the reader some insight into ATM networks, the issues with providing security over those networks and an introduction to issues using reconfigurable logic for the main encryption hardware. We also provided data regarding the implementation of cryptographic algorithms in reconfigurable hardware in general, such as the cost vs. speed, and how to asses an algorithm for its size and delay characteristics before any design work begins. Although there has been substantial work done in the area of reconfigurable architectures, very little has been done in terms of cryptographic algorithms. We hope that this work will alert both potential developers of security devices and reconfigurable hardware vendors about the viability of cryptographic applications and the need for further study. Often times crypto algorithms exhibit patterns in there architecture that may be exploited by new hardware designs. With the recent interest in cryptographic technologies by mass market companies, the use of new hardware technologies will be of value.

Device	Opt.	Floorplan	Resources	Delay(ns)	Clock(MHz)	Tput.(Mb/s)
XC4020EPG223-3	Area-low Collapse=off Redundancy=off P/R=2.2	Manual	448	154.9	7.00	27.99
	Same as above P/R=4.4	Automatic	646	115.0	9.10	36.40
	SBOX=RAM	Manual	359	145.5	6.80	27.20
	SBOX=RAM P/R = 4.4	Automatic	549	76.0	14.10	56.40
	SBOX=RAM	Auto w/o Bus	545	70.5	15.60	62.40
EPF10K30RC240-3	Norm/Speed/Area	Automatic	1319	112.7	9.99	39.96
	SBOX=RAM	Automatic	403+8EAB	69.4	14.4	57.60

Table 3.7 DES Performance

In closing, there has been a lot of work done in the last few years regarding ATM security. I believe that the concept of using reconfigurables for this technology is a promising and interesting addition to the growing interest in the field of high speed secure networks. It is hoped that through the experiments performed in this study any designer can make an intelligent decision as to which hardware will meet the needs of their application.

5. Bibliography

[1] C.M. Adams and S.E. Tavares, Designing S-Boxes for ciphers resistant to differential cryptanalysis. Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, pages181-190, Feb1993.

[2] E. Bilham and A. Shamir, Differential cryptanalysis of DES like cryptosystems.
In Advances in Cryptology CRYPTO '90 Proceedings, pages 2-21 Springer Verlag, 1991.

[3] E. Bilham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems. In Journal of Cryptology, volume 4 pages 2-21,1991

[4] L. Brown, J. Pieprzyk, and J. Seberry, LOKI: a cryptographic primitive for authentication and secrecy applications. In Advances in Cryptology – AUSCRYPT'90 Proceedings, pages229-236 Springer-Verlag 1990.

[5] J. Daemen, R. Govaerts, and J. Vandewalle. A new approach to block cipher design. In Fast Software Encryption, pages 18-32. Cambridge Security Workshop Proceedings, Springer-Verlag,1994.

[6] W.F. Ehrsam, C.H.W. Meyer, R.L. Powers, J.L. Smith, and W.L. Tuchman. Product block ciphers for data security. U.S. Patent Number 3,962,539 June 1976.

[7]H.Gutowitz. Cryptography with dynamical systems. Cellular Automata and Cooperative Phenomenon,1993.

[8] X. Lai and J. Massey. A proposal for a new block encryption standard. In Advances in Cryptology-EUROCRYPT '90 Proceedings, pages 389-404. Springer-Verlag, 1991.

[9] S. Lane. Security issues in moving from private to public ATM service. In ITU Americas Telecom 96, Technology Summit, Rio de Janeiro, Brazil, June 1996. ITU.

[10] S. Lane and G. Cohen. Security in ATM networks. Proceedings of the Technical Conference on Telecommunications RD in Massachusetts, pages 23-32, 1995.

[11] J.L. Massey. SAFER K-64: a byte oriented block-ciphering algorithm. In Fast Software Encryption, pages 1-17. Cambridge Security Workshop Proceedings, Springer-Verlag, 1994.

[12] M. Matsui. Linear cryptanalysis method for DES cipher. In Advances in Cryptology-EUROCRYPT '93 Proceedings, pages 386-397. Springer-Verlag,1993.

[13] M. Matsui. Linear cryptanalysis of DES cipher (I). In Proceedings of the 1993 Symposium on Cryptography and Information Security (SCIS 93), pages 3C.1-14, Shuzenji, Japan, Jan 1993.(In Japanese).

[14] M. Matsui. Linear cryptanalysis method for DES cipher(III). In Proceedings of the 1994 Symposium on Cryptography and Information Security (SCIS 94), pages 4A.1-11, Lake Biwa, Japan, 27-29 Jan 1994. (In Japanese).

[15] R.C. Merkle and M. Hellman. On the security of multiple encryption. Communications of the ACM, 24:465-467, 1981.

[16] R.L. Rivest. The RC5 encryption algorithm. Dr. Dobb's Journal, 20:146-148 Jan 1995.

[17] M.J.B. Robshaw. Block ciphers. Technical report, RSA Laboratories, Jul 1994.

[18] M.J.B. Robshaw. Personal communication, 1995.

[19] Bruce Schneier. Applied Cryptography Protocols: Algorithms, and Source Code in C. Wiley 2nd edition, 1996.

[20] Daniel Stevenson, Nathan Hillery, and Greg Byrd. Secure communications in ATM networks. Technical report, MCNC 1995.

[21] Doug Stinson. Cryptography: Theory and Practice. CRC Press, 1995.

[22] Douglas R. Stinson. Cryptography: Theory and Practice. CRC Press, 1st edition, 1995.

[23] ANSI X3.92. American national standard for data encryption algorithm (DEA).American National Standards Institute, 1981.