Stream ciphers for secure display

Devlin, Iain

Stream Ciphers for Secure Display

Ph.D. Thesis

Iain Devlin

December 2007

Stream Ciphers for Secure Display

Iain Devlin

Abstract

In any situation where private, proprietary or highly confidential material is being dealt with, the need to consider aspects of data security has grown ever more important. It is usual to secure such data from its source, over networks and on to the intended recipient. However, data security considerations typically stop at the recipient's processor, leaving connections to a display transmitting raw data which is increasingly in a digital format and of value to an adversary. With a progression to wireless display technologies the prominence of this vulnerability is set to rise, making the implementation of 'secure display' increasingly desirable.

Secure display takes aspects of data security right to the display panel itself, potentially minimising the cost, component count and thickness of the final product. Recent developments in display technologies should help make this integration possible. However, the processing of large quantities of time-sensitive data presents a significant challenge in such resource constrained environments. Efficient high-throughput decryption is a crucial aspect of the implementation of secure display and one for which the widely used and well understood block cipher may not be best suited. Stream ciphers present a promising alternative and a number of strong candidate algorithms potentially offer the hardware speed and efficiency required.

In the past, similar stream ciphers have suffered from algorithmic vulnerabilities. Although these new-generation designs have done much to respond to this concern, the relatively short 80-bit key lengths of some proposed hardware candidates, when combined with ever-advancing computational power, leads to the thesis identifying exhaustive search of key space as a potential attack vector. To determine the value of protection afforded by such short key lengths a unique hardware key search engine for stream ciphers is developed that makes use of an appropriate data element to improve search efficiency. The simulations from this system indicate that the proposed key lengths may be insufficient for applications where data is of long-term or high value. It is suggested that for the concept of secure display to be accepted, a longer key length should be used.

Declaration

The work in this thesis is based on research carried out in the School of Engineering, Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

GLASS liquid crystal displays pervade the modern world, especially in the area of consumer electronics, where everything from portable music players to wide screen televisions use the technology to convey visual information. Such is the development drive to improve the efficiency, resolution and responsiveness of these displays, that the underlying glass substrate technologies are now capable of accommodating integrated circuit electronics.

These technologies have resulted in the integration of the display with previously external electronic systems, including that of video-display drivers found in mobile phones. As the technology develops, further integration of other more complex systems will become possible, including the merging of security functionally with the display. This security integration, called 'secure display', would enable secure communications with the panel substrate, and has potential applications in portable display systems dealing with valuable or sensitive data, where protection of the display link is desirable.

A secure display system will require integration of a number of security functions but due to the high throughput requirements of video data, one of the most crucial is data decryption. The choice of algorithm must not only achieve high throughput but must also be power and substrate area efficient, to meet the needs of portable devices. The class of data encryption algorithms called synchronous stream ciphers potentially satisfy these criteria, but find themselves in a period of turbulence after recent security failures. A European effort, 'eStream', set up to resolve these concerns offers hope that the necessary secure, minimal resource, high speed decryption needed for secure display can nevertheless be realised.

1.1 Thesis Outline

This thesis looks at the suitability of using modern stream ciphers for secure display systems. In doing so, a number of more general issues relating to stream cipher security have been addressed and the first public stream cipher key-search engine developed.

The remaining sections of this chapter introduce some basic principles of data security and outline how the field has become relevant to consumer electronics. Chapter 2 then looks at the value of the secure display concept and how it may be implemented, with Chapter 3 examining the related area of audio and video data protection schemes. The focus then shifts onto stream ciphers, with Chapter 4 detailing their history, usage and the current European effort to select a new primitive suitable for hardware applications. Chapters 5, 6 and 7 build on this background to look at stream cipher key length, evaluating specifically the security provided by 80-bit keys through the development of a key search machine – DELUGE. In Chapter 8, finally, the wider implications of the generated results are examined in respect of both stream cipher design and the choice of primitive for applications such as secure display.

A number of the research findings on key length and stream cipher key search outlined by this thesis, have previously seen presentation at the European workshops SHARCS 2006 [1] and SASC 2007 [2]. A further unpublished letter on the DELUGE key search engine written in conjunction with Prof. A. Purvis forms a concise summary of aspects surrounding the machine's development and is included in Appendix A.

1.2 Data Security

Data security is based around the principles of protecting the confidentiality, integrity and authenticity of data. The protection measures used include physical barriers though it is more common to focus on operations performed in the digital domain. Encrypting data is a fundamental tool in this matter, with its aim to provide confidentially by turning a large secret, raw data, into a relatively small secret that is the cryptographical key. It is easier to protect this small fixed amount of key material, than it is to protect possible gigabytes of original data.

It is impossible for an encryption scheme other than the one time pad (see

Section 3.2) to be perfectly secure as a computationally unbounded attacker can always recover the key. With unlimited resources an attacker can just search the finite key space by exhaustive search and check each one for its ability to decode the encrypted data (commonly known as ciphertext.) An assumption is therefore made, that in reality an attacker is always bounded by time, memory and computational effort. If all possible attacks use an unreasonable amount of one or more of these resources, then the cryptographic algorithm is deemed secure.

A wide variety of apparently secure algorithms often exist. However, choosing the most suitable one is not straightforward since it is impossible to know when a chosen algorithm might be broken. All that can be done is to choose the most suitable algorithm that has stood up to thorough cryptanalysis and hope the high probability that it will remain secure holds.

1.3 Multimedia and Security

Multimedia is one area in modern times where the consideration of data security has grown in significance. Historically copying media such as art works and books would be an arduous task that generally produced a copy of inferior quality to the original. However, the advent of digital media has removed these factors, and, as with any digital data set, multiple perfect reproductions are not only possible but, indeed, almost trivial.

Multimedia content providers place great value on protecting their data and so a number of methods to discourage and prevent the practice of unauthorised copying have been proposed and introduced through the years. The business rational behind these schemes, is that by limiting the consumers' ability to copy the media, its availability will be restricted to official channels, thus increasing revenue and sales. The alternative rational — that by restricting the consumer's ability to use data, value is reduced — has however, gained traction in some sectors with recent moves in online music directed towards selling digital rights management free content.

As is later discussed in Chapter 3, it is notable that almost all copy prevention and control schemes have been broken. This is often down to some poor security design choices, but is also due to the extremely hostile environment in which protected multimedia data finds itself, where the user themselves must be considered an adversary.

Chapter 2

Secure Display

In the wide range of systems where data security is of importance, great care is usually taken to secure network transmissions between each party. Further, where one party is in a possibly hostile environment, it is not unusual to process received data in a secure manner through the use of a specialist crypto-processor. This is fine for devices such as smartcards, but where the data is to be visually displayed, as in the case of some multimedia transmissions, a further unsecured stage (Figure 2.1a) is present from the processor to the display screen. Depending on the application environment and display method this may present a significant security vulnerability in respect to covert and malicious data capture. The vulnerability of displays to data interception is discussed in detail in Section 2.2.1.

Secure display (Figure 2.2) attempts to address this issue by securing the transmission all the way to the screen itself. This is unlike High-bandwidth Digital Copy Protection (HDCP) present in many high-definition video systems (Section 2.2.3) which attempts to secure the data to the display device but not the actual panel substrate. The applications and advantages of the secure display approach are examined in Sections 2.2.2 and 2.2.4 respectively, with design requirements and approaches looked at in the later sections of the chapter.

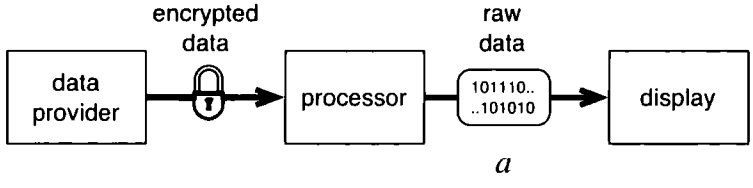


Figure 2.1: Typical display system set-up

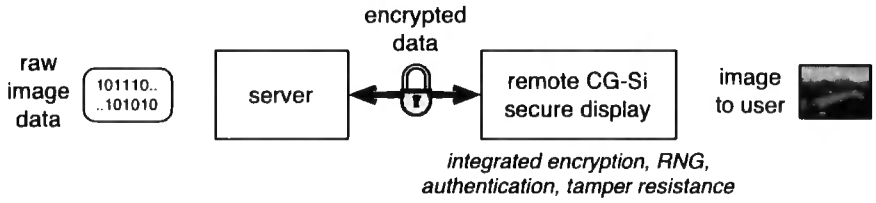


Figure 2.2: Secured display system set-up

2.1 System LCD

The idea of secure display comes from the wider concept of System LCD [3, 4] which makes use of recent advances in glass substrate technologies (see Section 2.1.1) to allow the integration of external circuitry onto unused display substrate space (Figure 2.3). This has typically involved integrating display drivers [5] to decrease the size and cost of implementing a LCD (Liquid Crystal Display) system but the System LCD concept has a wider aim of freeing the display to be used as a standalone device. The implementation of a small microprocessor on glass has already been demonstrated [6], and Sharp Corporation [7] have more recently proposed the integration of touch screen and optical scanner technologies on a LCD display.

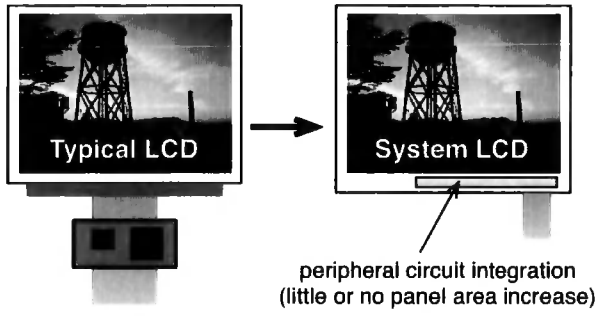


Figure 2.3: Outline of System LCD concept

The development of System LCD has relied on the ability of the glass substrate to sustain integrated circuits. In the next section we explore the characteristics of silicon on glass process technologies so as to understand the limitations placed on implementing the secure display concept.

2.1.1 Silicon on Glass

Silicon on glass (SOG) is a category of fully depleted silicon on insulator (SOI) technology that makes use of glass as the substrate material. Unlike other SOI

technologies, development has been driven by the desire to improve thin film transistor (TFT) display designs and not high speed electronics; this has led to very large substrate sizes, 1.5×1.8m [8] or larger, being used to drive down area costs. The constant need for higher resolution, faster response, lower power displays has demanded reductions in feature size and increased mobility; as a result a variety of process technologies now exist in the field. Two of these, amorphous silicon (a-Si) and low temperature poly silicon (LPS), will be outlined in the remaining paragraphs of this sub-section before the high performance substrate technology that lies behind System LCD — continuous grain silicon (CG-Silicon) — is examined. It should be noted that although none compete with crystalline silicon processes in terms of performance, all have the advantage of reduced substrate cost that ultimately lies behind much of the commercial interest in function integration.

Amorphous Silicon

The first glass process technology looked at is that of the once widely used amorphous silicon technology. Displays based on a-Si are some of the cheapest to manufacture and have a thin layer of silicon put on the glass surface by a process such as plasma enhanced chemical vapour deposition (PECVD). This layer lacks the very regular structure associated with crystalline silicon, causing the mobility to be just $1cm^2/Vs$ and thereby making the technology very slow at transistor switching.

Low Temperature Polysilicon

The later LPS process technology aimed to improve on the characteristics of a-Si display through use of a XeCl excimer laser to anneal the silicon layer on a-Si substrates so that the silicon would crystallise. This significantly increases electron mobility with figures around $200cm^2/Vs$ achievable [9]; allowing faster response times and higher resolution displays with further possibilities of implementing basic driver circuitry on the substrate. In view of this, LPS technology has largely replaced a-Si as the TFT technology of choice.

2.1.2 Continuous Grain Silicon

LPS is not without its problems: crystal grain size is small and misorientation angles are large at 30–60° [10] causing the substrate to be inconsistent enough

to adversely affect IC implementation. CG-Silicon is a technology developed by Sharp, in conjunction with Semiconductor Energy Laboratories, that builds on LPS technology to give higher electron mobility ($320cm^2/Vs$) by increasing grain size to larger than $1\,\mu m$ and reducing misorientation angles to less than 10° [10]. This also gives the substrate a greater consistency allowing IC electronics to be easily integrated onto to the same glass panel as a display.

CG-Silicon fabrication [10] starts with the deposition of thin layers of SiON and SiO_2 on the glass panel to avoid glass impurity diffusion. PEVCD is then used to form a very thin 45nm layer of a-Si before a spin-coating of Nickel solution is applied. This is followed by an annealing process at 450–600C in an atmosphere of nitrogen. Finally excimer laser annealing is used to further improve crystal quality so that all that is left is well-ordered crystalline silicon. The closeness to crystalline silicon significantly reduces the prominence of SOI characteristics such as the kink effect in comparison to LPS. However, dimensionally CG-Silicon is not comparable with modern silicon circuits with the order of magnitude larger, $0.8\,\mu m$, feature sizes quoted by Lee [6] for 2005 processes, meaning that any implemented logic on the substrate must operate in a comparatively restricted environment. Further substantial reductions in feature size may be achievable for CG-Silicon but will require the challenge of defining fine detail over a large substrate to be addressed [11]. Therefore, for in the foreseeable future, integration must principally be based on cost and area savings.

2.2 Value of Securing the Display

Glass LCD displays are prevalent throughout not only the modern consumer electronics sector but also many other sectors and these displays will on occasion be used for displaying valuable private, proprietary or highly confidential material. This data is normal transmitted securely in encrypted form to the host's processor unit and is sometimes even processed securely within the processor unit; however, on its final stage to the display the data is transmitted in the clear. This presents a significant security vulnerability in respect to covert and malicious data capture.

The usage restrictions placed on the display by this vulnerability in the display link is also problematic and has conventionally led to displays being shackled in close proximity to their parent processors: the case of a workstation computer being but one example. By utilising the mount area of the display, cryptographic functions

such as decryption could be implemented onto the glass substrate without increased cost. This integration would allow the freedom to create the 'display-only' devices where data is sent securely to and from a remote server, leaving the display as a low power, highly-compact portable device.

2.2.1 Display data interception

2.2.1.1 Attack Tree

In designing a system of secure display it is useful to consider the potential interception methods available to an adversary so that each attack's potential may be accessed and, if necessary, prevention methods taken. To do this it is normal to construct a tree diagram of the possible attack methods and so below is presented a series of attack trees for obtaining data transmitted to a display.

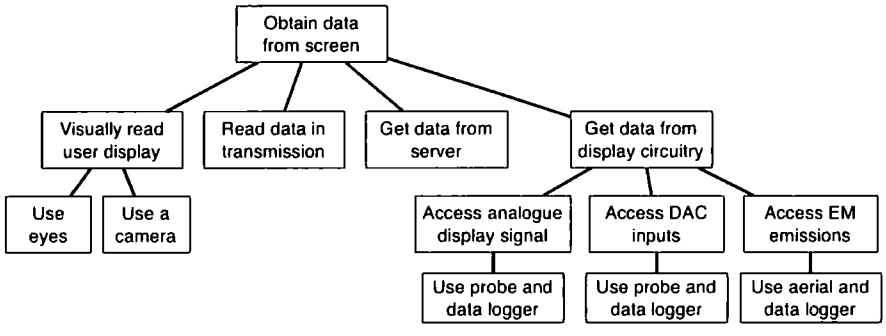


Figure 2.4: Basic attack tree for a server to display system

The first of three attack trees designed is shown in Figure 2.4 and represents the base level situation where unprotected display links are used and no countermeasures are in place. The attacker has four main options open to him: reading the displayed data, reading the data in transmission, getting the data direct from the transmitter and getting the data from probing the display circuitry. All of these options are potentially inexpensive and it is important to note that, although secure display targets the protection of data in transmission, a number of other simple attack scenarios always remain open.

The second tree (Figure 2.5) is a partial attack tree on the same display link but this time some simple form of encryption has been used to protect the data in transit. This tree closely represents the range of threats that are expected to be issues in a fixed key secure display system that is shown later in Figure 2.9. A number of

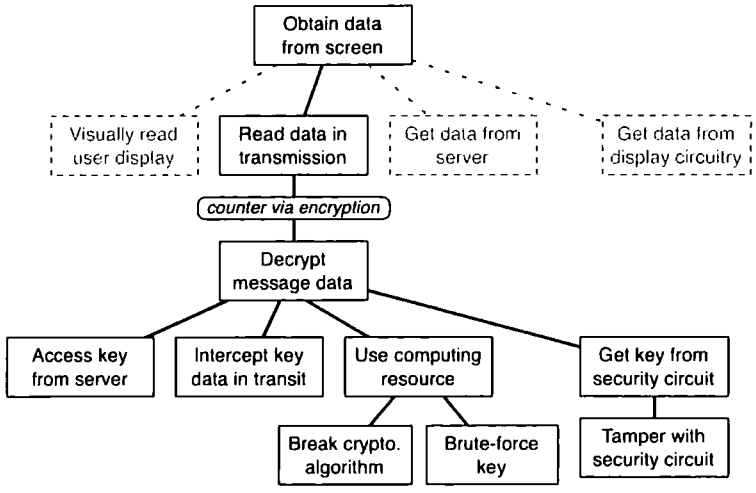


Figure 2.5: Attack tree for a server to display system with encryption countermeasure

relatively feasible options are still possible to an attacker, especially when focus is targeted towards obtaining the encryption key.

By introducing further countermeasures of a secure key exchange protocol and some form of tamper resistance to the cryptographical functions of the display circuitry, targeting the key becomes much more challenging to an adversary. A partial attack tree for this now improved form of display link protection is shown in Figure 2.6. The options available to a adversary are generally expensive or complex making this system relatively well protected and as such this is the unattractive attack scenario we want to achieve through secure display. The full secure display system shown later in Figure 2.10 was envisaged to closely reflect the countermeasures of this attack tree although it additionally uses authentication to add a further countermeasure to the system and prevent data alteration based attacks.

2.2.1.2 Display connector attacks

In reading the data in transmission, possibly the simplest attack that can be envisaged against an unprotected display link is that of an attacker tapping the cable at one end. This has the requirement that direct access to the display link must be available and so a suitable countermeasure against display connector attacks is to keep display and link in a physically secure location. As seen in Section 2.2.1.1 encrypting the data over the link is equally as effective.

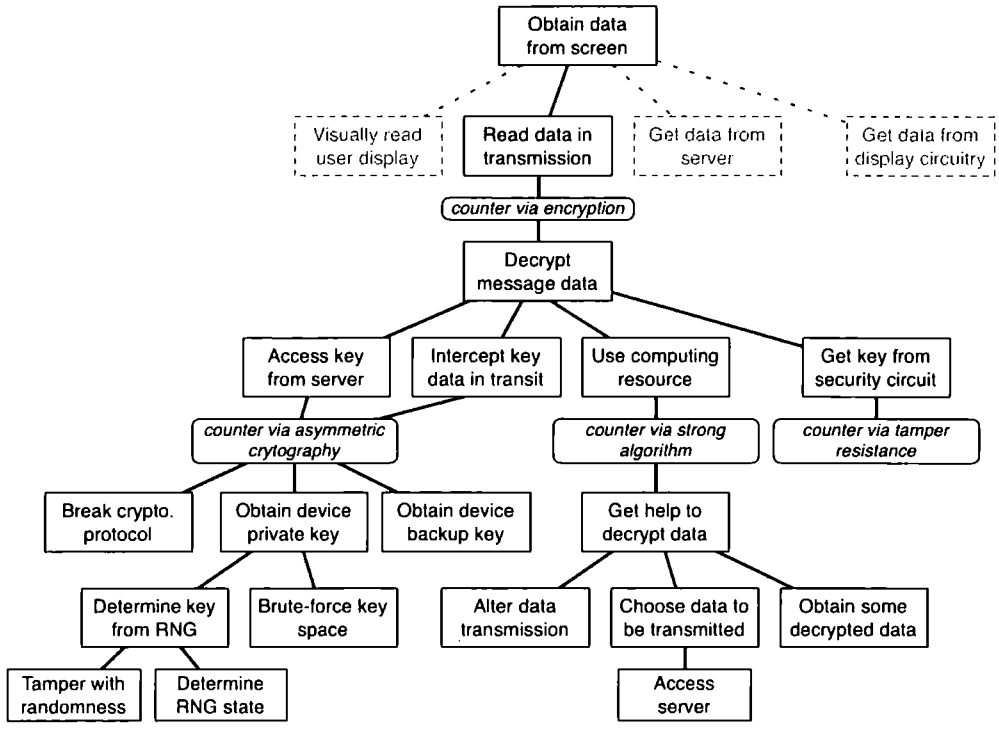


Figure 2.6: Attack tree for a server to display system with a second round of countermeasures

2.2.1.3 Remote attacks

In conventional digital circuit design it is normal to leave the design and layout as undefined as possible to allow automatic tools to generate the most efficient implementation for a particular architecture. In secure systems however, another factor must be taken into account: the variability of power consumption during operation. This is essential so that power and EM (electromagnetic) analysis do not lead to key or state information leaking. If one operation takes slightly longer for a particular data set this can usually be detected using little more than a modern oscilloscope and EM probe [12] making such techniques often by far the cheapest way to compromise supposedly secure circuits.

Worse still is the fact that these attacks do not physically affect the device, making detection impossible without securing the environment in which the device resides: a task that is all but impossible for consumer electronics. Security conscious industries, such as that of smarts cards, are willing to make huge sacrifices in terms of area and power consumption to mitigate the effects of side channel leakage and logic cell libraries such as WDDL (wave dynamic differential logic) are of growing interest despite using 3x area and 4x power of conventional cells [13].

Both time domain [14, 15] and frequency domain analysis must be considered as well as an ever increasing variety of other analysis operations. In larger area devices spacial layout may start become a consideration, however, present attacks are so effective that currently the spacial element is unnecessary. A prominent example of this effectiveness was the 1985 attack [14] on some RSA encryption (see Section 3.4) implementations which, with the observation that processing took different times depending on the key in use, rendered the problem of finding a key, impossible to retrieve though computing power, almost trivially solvable.

It has been long known that CRT computer monitors are vulnerable to eavesdropping through EM analysis attacks [16] but more recently Markus Kuhn [17] has shown that laptop and other TFT-LCD screen displays may be similarly vulnerable. In this later case it has been demonstrated that a screen can be read while in another room simply by scanning for emanations from the digital video link between the display controller and LCD or alternatively from the digital video interface (DVI) cabling. This example of EM analysis, although unrelated to data security algorithms, demonstrates the ability of side channel attacks to defy physical security. Securing the external cable signal is one part of the solution (HDCP described in Section 2.2.3 potentially does this) however to minimise emanations it is beneficial to secure the data flow as far up the system chain as possible: a task which forms the concept of secure display.

2.2.2 Applications of Secure Display

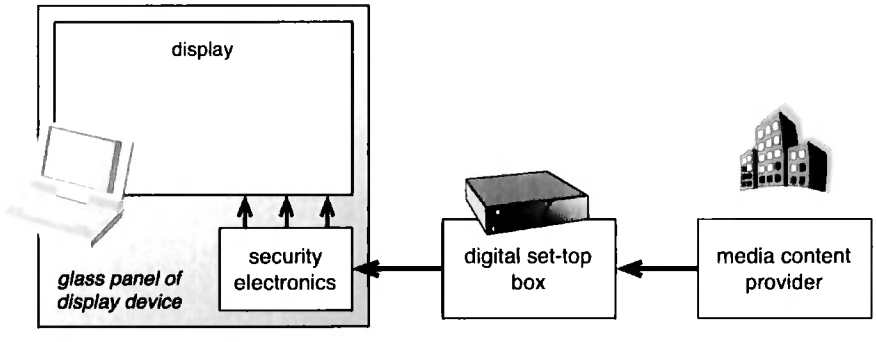


Figure 2.7: Audiovisual content application of secure display

The improvement in display data security and the possibility for untethering the display creates the potential for a number of applications of the secure display concept. However, as in many consumer systems the most commercially attractive area of these is that of multimedia content protection. A system diagram for just

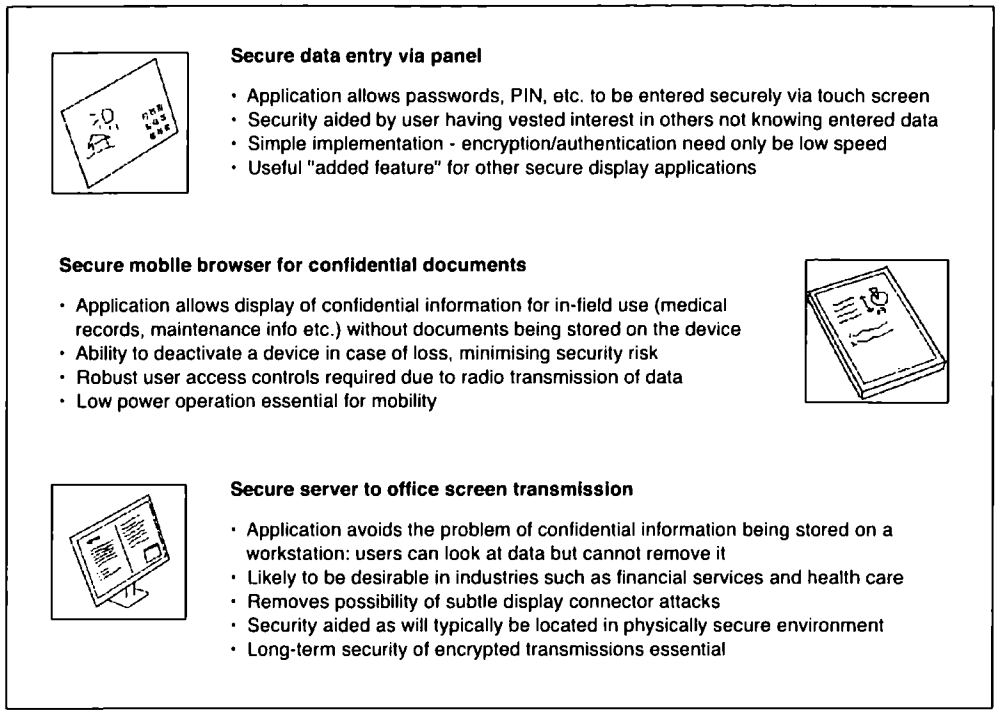


Figure 2.8: Three possible applications of the secure display concept

such an application of secure display is shown in Figure 2.7: the set-top box shown may also be reconsidered as a service provider who wants to transmit digital rights managed content, such as a movie, to the users mobile. This multimedia streaming application has a number of security challenges associated with it, including that such data is typically commercially attractive to criminal organizations making a well funded attack a real possibility. There is also a certainty that the intended mobile devices will fall into the hands of hostile users, willing and able to potentially probe and alter signals in the security circuitry and necessitating further counter measures in the system design.

Securing media content transmissions is by no means the only possibility open to a secure display system and three other application ideas for the secure display concept are presented in Figure 2.8. All four application ideas have their own attractions; currently, however, high speed, high resolution display technologies, such as CG-Silicon, are better suited to small sized displays. This implies that the most relevant application areas at present and in the near future will be those that involve small portable displays. In view of this, minimising power usage without compromising security and usability will be an essential part of the design process.

2.2.3 High-bandwidth Digital Copy Protection

The most commercially attractive application area of secure displays is that of media content protection, due to the strong attitude the media industry takes to access control of copyrighted material. A standard with clear similarities to the secure display concept, HDCP [18] (High-bandwidth Digital Content Protection), was published back in 2000 [19] to address this very issue. This standard attempted to secure the transmission of TV and movie data from sources such as computers, DVD players, and set-top boxes so that it could only be played back on an authorised display device. However, HDCP did not last long as a secure standard: Ferguson suggested that a practical attack was possible [20], and Irwin informally published an attack [21] within 18 months of the standard's publication. Further cryptanalysis of HDCP confirmed the weakness of the protocol [22]. Despite this, HDCP has found its way onto a wide range of recent TVs, game consoles and high-definition video disc players, although the actual activation of the protocol through an image constraint flag on media content has yet to happen.

2.2.4 The Secure Display Advantage

The concept of secure display in securing data right to the display substrate not only protects against physical and remote display link attacks but also should help prevent attacks utilising emanations from the display circuitry itself, as demonstrated by Kuhn [17]. If well designed, it will provide a much more effective solution to the issues surrounding streamed content protection than has previously been seen with HDCP.

The integration aspects of secure display are especially attractive to mobile and portable devices in comparison to externally mounted circuitry, providing potentially reduced costs, component count and final product thickness. The combining of security and display functionality is also another step towards freeing the display to be used as an independent device, as envisaged in the concept of System LCD.

2.3 Design Requirements

A clear requirement of any secure display system design is that it must be capable of processing a very high data-rate signal due to the nature of video display signals. For applications such as mobile phone displays a typical size LCD is 320×240. Assuming 6-bit RGB colour and 30Hz refresh rate, this leads to a data input rate of around 42Mbits/s, and while this figure is high it would not normally cause problems for implementing display link security. However, on the power and area constrained substrates of even advanced glass displays (see Section 2.1.2) it is an aspect which will have a significant bearing on hardware logic design.

As an intended application's LCD pixel size is decreased or display area increased the data throughput rate will rapidly increase: a 1280×1024 24-bit colour display operating at 60Hz indeed requires a throughput rate of nearly 5.7Gbits/s. Data compression could significantly reduce this figure and would help wireless connectivity tremendously in reducing the data transmitted. However, such compression would simply turn the decompression stage into the main system bottleneck with decreased output quality and increased latency being amongst a number of further concerns. Low latency is an especially desirable attribute for secure display as this would improve the user experience, particularly in applications specifying user interaction or involving live video, where it would minimise potential problems of audio synchronisation.

From a security perspective the attack trees of Section 2.2.1.1 indicate that apart from securing the data in transmission, it is useful to consider implementing a form of secure key exchange and tamper resistance when designing secure display systems. Tamper resistance will not be specified in the system design of Section 2.4 but it is assumed that a final implementation of secure display would include measures to physically protect the logic operation of security circuitry. It is expected that these measures in mobile systems will be similar to those used in modern smartcards [23, 24].

2.4 System Design

From the attack trees of Section 2.2.1.1 a number of secure display system designs can be considered. Two designs: one basic, suited only to prototype implementation; and one fully featured, suited towards secure display in a final product; are presented in this section.

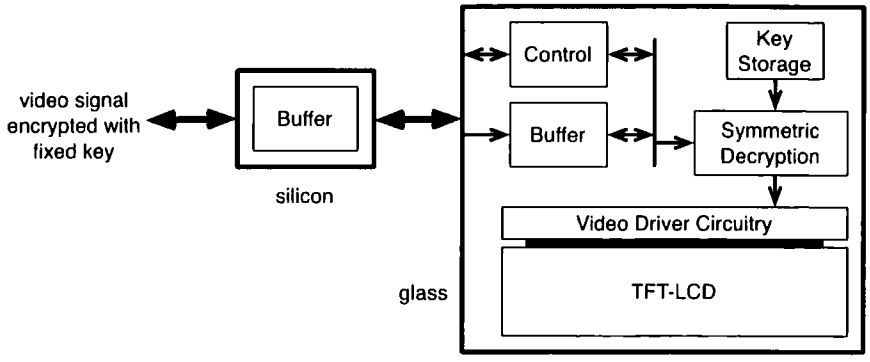


Figure 2.9: Basic fixed key system model for a secure display system

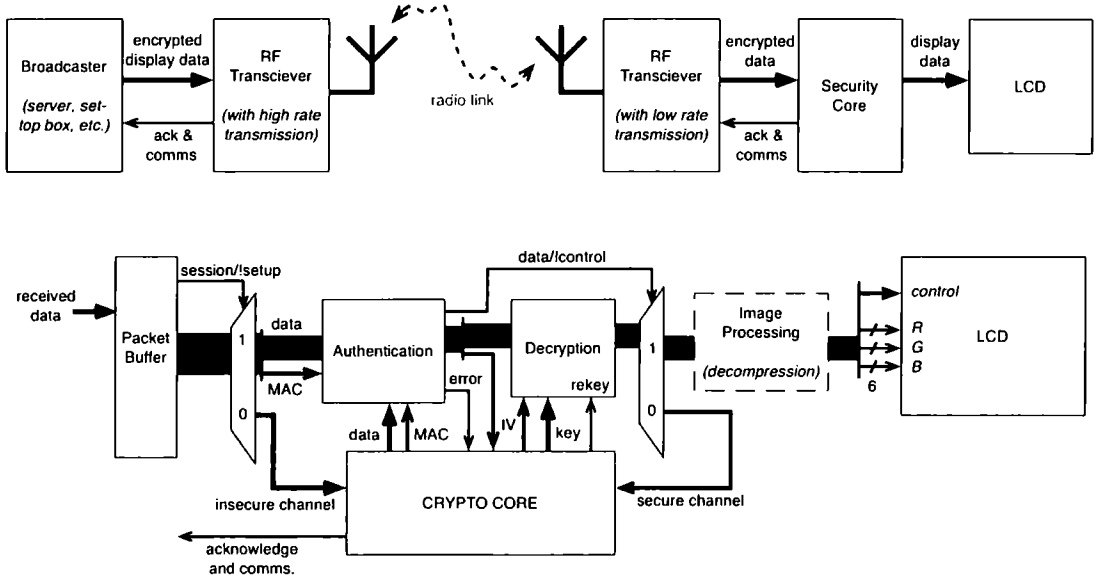


Figure 2.10: Secure display system layout

The basic secure display system (Figure 2.9) involves implementing a single countermeasure to display link attacks based on reading data in transmission. This system requires only the implementation of symmetric decryption and uses an embedded key. Although, as discussed in Section 2.2.1.1, it only provides a minimal level of security it would nonetheless be suitable for demonstration of the secure display concept.

Figure 2.10 shows a more fully featured secure display system, appropriate to final product implementation, where keys are no longer fixed but instead exchanged at the initialisation of a protected content transmission. This design targets the scenario where the display data involves wireless transmission to a mobile display device with throughput of data transmissions sided towards the server. To minimise the problems of maintaining device portability it is proposed that initially only the functionality

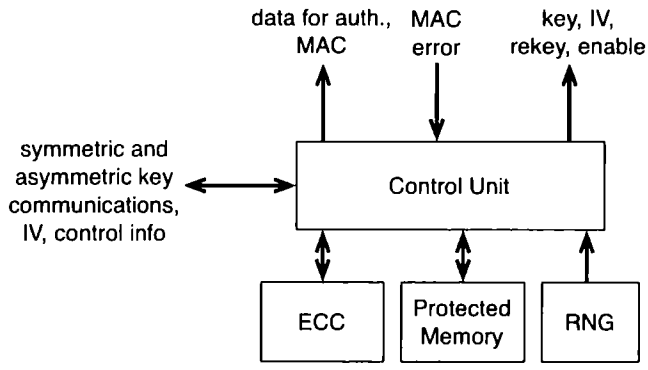


Figure 2.11: System diagram of the cryptographic processing core

of the security core is implemented on panel, although, as glass substrate technology improves, further integration of the transceiver may become possible. The security core, also shown in Figure 2.10, implements the basic functionality of secure display and takes the transmitted protected data, checks it for errors and then decrypts it before passing the decoded video information to the LCD.

The cryptographic processing core of the system controls the general functions required to maintain system security. It can be split down into a further level of sub-blocks, as shown in Figure 2.11, with a hardware elliptic curve cryptography (ECC) unit and random number generator (RNG) providing the ability to handle key establishment protocols for setting up the decryption key. The control unit also handles a number of other bits of control data including any initialisation value (IV) transfer, message authentication code (MAC) transfers, acknowledgements and authentication errors. The whole processing block, in essence, acts as a small crypto-processor and would seem likely to have similar throughput and processing requirements to those of a smart card.

It may be noted that full secure display system of (Figure 2.10) is intended to provide the confidentiality, data integrity and authentication typical of many cryptography applications. Non-repudiation, the fourth goal of information security, as set-out in [25] *(pp.4)*, prevents a user denying an action or commitment and would only be needed where a user has direct interaction with the display. This might be the case with a touch screen display system; however, data rates are likely to be low, thereby making large changes to the system unnecessary, as only the operation of the cryptographic processing core sub-block would be affected.

2.5 Focus

There are four sub-systems that require high levels of performance to implement the system shown in Figure 2.11: authentication, decryption and image processing, due to their high throughput requirements; and the cryptography processing core, due to the complexity of securely completing key agreement. This final section assesses the relative importance of further research in each of these core functions to the development of a demonstration system for secure display.

Imaging processing such as decompression and resolution enhancement is useful and possibly adds value to a commercial system, however, the function block is not considered fundamental to the concept of secure display. In view of this, it is reasonable to disregard this system component and concentrate research effort elsewhere, as the design of early prototype systems and even a final fielded system would not need to include it.

Authentication is useful in preventing forgery attacks by immediately discarding invalid messages. It is particularly valuable when a two-way protocol, with send and receive functionality, is desired such as for a touch screen display. Nevertheless for systems where denial of service attacks are not a concern it is possible that only authenticating control signals would be sufficient so long as measures were taken to avoid replay attacks: this may include disallowing out of sequence initialisation values by expecting the value to be greater each time or the more memory intensive option of disallowing initialisation value reuse. This potentially reduces the throughput requirements to the extent where minimum area implementations of the core functions underlying the message authentication algorithms may be used; indeed given that the speed may be much reduced it should be possible to base the MAC around the chosen decryption algorithm and let the crypto-processor periodically borrow the hardware for this function.

One caveat to this is the role the authentication function plays in providing data integrity within the system, which allows a viewer to be certain that the on screen image has been unaltered. This is a particular problem for systems that use synchronous stream cipher encryption functions (Section 3.3.2), or related block cipher modes of operation, since bit flipping by an adversary would fail to cause a catastrophic breakdown in the decrypted output and may allow the rendering of unwanted messages on screen. Nonetheless, although data integrity is desirable in commercial security systems, it does not affect confidentiality. Applications such

as the audio-visual one described earlier deliver data over a fully secured network connection and so data is only exposed on the final stage to the screen. In this situation data integrity attacks are unlikely as the user will usually be in control of this environment and has no reason to corrupt their own data as it does not help them access the underlying content. In light of this, at least at the development stage of secure display, data integrity protection of the raw video data is not essential.

The asymmetric encryption (Section 3.4) required for the initial key agreement between display and data server will generally only be needed at the start of a new transmission and is not time critical like the main video data path. This leaves area and power considerations the primary concern and makes the requirements of this subsystem similar to that of smartcards. Elliptic curve cryptography (ECC) is typically the core function of the asymmetric encryption in these devices making it likely to be the preferred choice for secure display. As a highly commercial sector of data security, smartcard optimised ECC is an increasingly mature technology with the scope for further improvements limited and off the shelf modules easily available to purchase. Furthermore, at the development stage of secure display a fixed key based system should be sufficient to demonstrate the potential of the concept and may even have practical uses.

The provision of confidentiality provided by encryption is fundamental to the concept of secure display and cannot be avoided. With this in mind, the research work of the thesis concentrates on this remaining sub-block of the secure display system and the next chapter will examine the issues surrounding maintaining security in high throughput, minimum area data decryption scenarios, as well as looking at commercial attempts at multimedia protection.

Chapter 3

Protecting Stream Data

THE failure to properly secure streamed data is an issue that is certainly more widespread than HDCP; indeed, almost all popular media formats that use encryption and access control have seen some form of successful attack. In this chapter various methods of securing data transmissions are examined, with particular focus being paid to the suitability of encryption methods for the secure display system described in Chapter 2. Current schemes for video and data stream scrambling are briefly reviewed before the cryptographically standard structures for encryption are examined. With special consideration to the resource constrained nature of secure display systems and the need for high throughput, the symmetric encryption method of using stream ciphers is concluded to be the most promising direction for development.

3.1 Past and Current Methods

As touched upon in Chapter 2, there is great value attached to protecting stream data when it is in the form of multimedia content and during the past few decades a number of schemes have been proposed and implemented in an attempt to achieve this goal. The difficulty is that this form of data, be it voice, television, or more recently digital video, requires the processing of high volumes of information rapidly and at low cost to be economically viable. This has led to deployed schemes often focusing on providing something short of full encryption of communications.

Some of the earliest attempts at protecting stream data were focused at audio

with concerns about cassette copying leading to schemes devised to spoil copies, Anderson [26] *(pp.421)* describes one such attempt for the 1967 hit record "Sergeant Pepper" where an inaudible 20kHz tone was included which was intended to produce, through mixing with the bias frequency, a very audible 1kHz tone in any copy. Similar attempts would later be made to discourage copying of commercially produced VHS tapes with the Macrovision Analogue Protection System [27] inserting pulses into the non-displayed signal to affect the recorder's automatic gain control making the picture in copies unpleasant to watch due to brightness variations. The move to digital data saw these spoiler schemes continue and many commercial DVDs now contain a couple of control bits set to cause outputted analogue signals from playback devices to include an Analogue Protection System signal [28] with the aim of discouraging transfer of content to video tape. Nonetheless in the digital world such spoiler schemes have a number of shortcomings in that they do not protect the raw data and, although they have minimal processing cost and may frustrate a general consumer's attempt at copying, a more determined attacker will have very little difficulty in overcoming the protection.

In the case of digital video many schemes have been proposed to provide a more comprehensive scrambling of the video signal which, although they do not meet the criteria for a cryptographically secure system, should prove harder to avoid. Methods such as Lian's [29] use encryption of part of the signal to destroy most of the visual information while providing little in the way of performance penalty to the already existing compression and decompression processes. However a scheme described as 'secure in perception' [29] is unlikely ever to provide protection of the underlying data from a determined attacker and, indeed, with the ease with which digital data can be cloned, from the wider population at large. Li's analysis of a number of other published stream data encryption schemes has shown there to be numerous insecurities [30, 31].

Such security failings in the design of stream data encryption systems are not restricted to the research community, with many commercially conceived designs likewise failing on scrutiny. These schemes have in the past practised 'security through obscurity', using strong licensing conditions to prevent implementation details being available for analysis. The philosophy is that by restricting the knowledge potentially available to attackers, any attacks on a scheme will be more difficult. However, the approach has several disadvantages in that a scheme will no longer receive thorough external analysis before release into a hostile and well-connected environment. Leaking or reverse engineering of a scheme's details is almost inevitable;

the reverse engineering of the Content Scramble System (CSS) digital rights management scheme for DVDs being one notable example and the leaking [32] of the Global System for Mobile communications (GSM) encryption standard being another. The consequences of this information release can be devastating for data security and the original DeCSS software that let users recover unencrypted raw video data from their DVDs in 1999 has since gone on to spawn numerous software applications which now allow easy and quick removal of CSS encryption to all but the most casual of attackers. Similar encryption based failings in the propriety GSM standard for mobile phones were found [33] a few years after the original leak, a subject further discussed in Section 4.4.1.

Security failings are not just the preserve of unpublished standards, the well published WEP (Wired Equivalent Privacy) scheme [34] widely used in protection data across wireless local area networks was shown to be weak only a couple of years after its 1999 ratification due to insecurities in the chosen encryption method (see Section 4.3.4). Bluetooth systems [35] were similarly be shown to be weak at protecting transmitted personal area network data (see Section 4.4.2) and as already discussed in Section 2.2.3 the copy protection standard for device to display transmissions, HDCP, is widely regarded as broken.

Even more recently devised encryption schemes with strong commercial drivers have suffered problems. A prominent example of this is the content protection scheme AACS (Advanced Access Content System) [36] developed to protect data on the high definition video disk formats Blu-Ray and HD DVD. The system uses conventional cryptography mechanisms such as hash functions, digital signatures and encryption in form of AES (see Section 3.3.1) yet although its specification is more public than that of its DVD predecessor and has largely resisted analysis it has not remained secure. The necessary use of embedded keys has led to acknowledged attacks [37] where encryption keys retrieved from software video players can be used to decrypt content. Attempts to expire the compromised keys by the AACS licensing administrator [38] may provide some short-term rest-bite for new releases but content on older disks will remain vulnerable.

Satellite and cable operators have an advantage over media disk distributors in providing secured content as their systems need not include a passive fixed element. In fact due to the commercially attractive nature of attacking such a system and the critical nature of conditional access to service providers' business models, pay-TV operators have been at the forefront of stream data protection mechanisms. A detailed outline of the progression from early analogue 'cut-and-rotate' scrambling techniques,

where each line would be rotated by set amount before transmission, through digital scrambling techniques, to the ups and downs of developing smartcard based systems can be found in [26] *(pp.423–430)*. Indeed with these experiences and the priority commercially, current pay-TV schemes are one of the more secure examples of stream data protection, employing conventional cryptography mechanisms as well as active links back to the service centres giving providers an ability to push security updates and to constantly update encryption keys. The strength of this protection has become even more critical in recent years with the advent of personal television recorders such as [39] where data is increasingly accessible due to its storage on a customer's set-top-box.

3.2 One-Time Pad

Protection schemes as have been outlined are often far from perfect. However it is possible to achieve perfect secrecy and the one-time pad cipher has provided this capability to cryptographers since the early 20th century. In this section we will provide a brief overview of one-time pad encryption.

In understanding the one-time pad we can consider two parties, Alice and Bob, who each have a copy of the same random bit-stream that is the symmetric key for the exchange. Alice wants to send Bob her message data or plaintext in a secure manner and so decides to mix it one bit at a time through a modulo-2 addition with the key material. Bob, on receiving this encrypted data or ciphertext, then uses his copy of the key to reverse the mixing process through another modulo-2 addition, this operation being self-inverting, and recovers the original data. He and Alice then forget the key material used. Indeed if the key material agreed by Alice and Bob remains secret, not reused and is truly random the transmitted ciphertext will appear to any third party man-in-the-middle attacker as a truly random bit-stream and no information of the original plaintext will be gleamed. The theory behind the unbreakable nature of this stream and the unique nature of the one-time pad was shown by Shannon in 1949 [40].

A notable feature of the one-time pad cipher is that as the modulo-2 addition (XOR) used in the combining function is its own inverse, identical hardware may be used for both encryption and decryption processes. However, securely generating, transporting and storing large quantities of truly random key material is far from an easy task and so the disadvantage of the one-time pad's requirement for key material

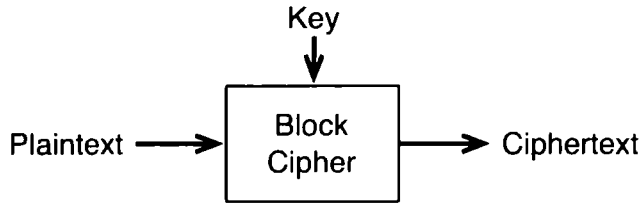


Figure 3.1: Block cipher encryption

equal in size to the plaintext data is a significant one. This is especially true of applications requiring large data sets to be communicated, as in video transmission, or where one of the parties is inaccessible, as in satellite communications and sealed consumer devices. As such, in most data security applications, methods of encryption which require only a small fixed length key are typically preferred.

3.3 Symmetric Encryption

Encryption schemes such as the one-time pad cipher, where the key is shared between the transmitter and receiver, are part of a broad category of encryption primitives referred to as symmetric encryption. The confidentiality provided by symmetric encryption schemes forms a fundamental part of data security and has led to their ubiquitous presence in implemented security systems.

Symmetric encryption primitives can be broadly split into two main subcategories: block ciphers, that operate on discrete blocks of data; and stream ciphers, that work on continuous data streams. During the rest of this section we look at each of these with a view to their suitability for secure display.

3.3.1 Block Ciphers

Block cipher encryption (Figure 3.1) takes a small block of data and maps it through a function that aims to mimic that of a truly random permutation so that the equally sized output ciphertext block contains no information for an attacker. By making this mapping function one-to-one and purely dependant on the key used, the generated ciphertext can be decrypted by a party in possession of the same key through use of the inverse map. The block size chosen in the cipher's design must make observing a match via a dictionary attack impractical.

In modern block ciphers the described mapping process is generally achieved by mixing the data with key material through a series of simple operations in a repeated round structure similar to Feistel's 1973 scheme [41]. By using a regular structure in the process of diffusing and confusing plaintext data, these networks enable flexibility in design implementations and typically it is possible for resource reuse between encryption and decryption mapping processes as key material just needs to be added in the reverse order. The use of a repeating structure combined with fixed length key and data inputs has a further design advantage in that it allows for easier analysis of their security, thus giving confidence to users of the algorithm. Unlike stream ciphers,[1] a number of generally regarded secure algorithms exist including perhaps the most prominent symmetric encryption scheme used today, the Advanced Encryption Standard (AES) [42]. The 128-bit keyed AES is formed around a 10-round structure and 128-bit block size with modern throughput efficient implementations requiring 12kGE (NAND Gate Equivalents) [43].

The forerunner of the AES, the Data Encryption Standard (DES), was at its time of equal prominence, and indeed its introduction in 1977 [44] was driven by a desire to popularise cryptography usage in commerce. The cipher operated on shorter 64-bit blocks and used a Feistel based 16-round structure in a relatively hardware efficient design. The DES would also have the unintentional affect of popularising public cryptographic research as pre-standardisation changes to the algorithm were at the time viewed with suspicion. These suspicions proved unfounded;[2] however, the DES's small 56-bit key size would eventually lead to its obsolescence as it was found to be vulnerable to exhaustive search [46, 47]. A longer keyed version of the cipher, triple-DES [48] is still currently in use.

Block ciphers would appear to have a great deal of promise for application in secure display with data throughput rates of 5.163Gb/s possible even on 0.5 μm silicon CMOS processes [49] *(pp.61)*. This is ample for the 42Mb/s needed for a small mobile display even when further speed restrictions caused by integration onto glass and need to reduce area usage are accounted for. Where block ciphers fail to deliver is in absolute efficiency as they need to process a full block of plaintext all at once. As we will explore next, stream ciphers have a fundamental advantage in this regard.

[1] It is often wise to use a block cipher in a mode that imitates a stream cipher so as to prevent data regularity leading to security vulnerabilities, this is a topic further discussed in Section 4.2.

[2] Coppersmith [45] would later describe that the changes to DES were actually intended to strengthen the algorithm against the advanced attack techniques of differential cryptanalysis.

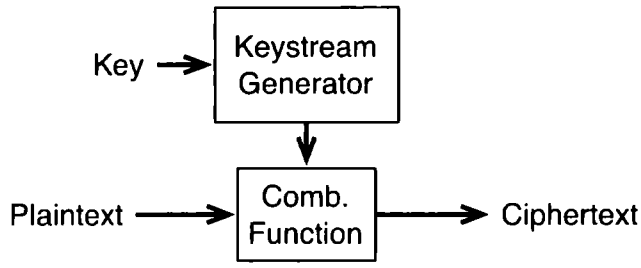


Figure 3.2: Stream cipher encryption

3.3.2 Stream Ciphers

The stream cipher aims to produce ciphertext by creating a stream of characters so complex that it is indistinguishable from a truly random stream (see Section 3.5) and then combining each of these random characters with those of the plaintext. When implemented in a binary context this closely shadows the implementation of the provably secure one-time pad scheme but has the advantage of using much less key material as only a small, fixed length key is needed to create the pseudo random bits of the keystream. A typical stream cipher structure is shown in Figure 3.2.

As stream ciphers aim to mimic the one-time pad and generally are expected to operate in a bit-wise manner the combining function typically used is a XOR operation. The reversibility of this operation having the added benefit of allowing stream cipher encryption and decryption to utilise the same configuration with the only difference being the reversal in directions of plaintext and ciphertext streams. Other mixing functions have been suggested [50] but hold few benefits. In fact the use of XOR is so widespread that the combining function becomes redundant when comparing stream ciphers.

Further implementation advantages transpire from the independence of plaintext data to keystream generation, in that stream ciphers only have to process the incoming data in the most minimal of ways. For the combining function this means incoming data only need be processed through a single XOR gate before outputted, thereby giving the stream cipher a low latency characteristic highly desirable for video transmissions. The absence of data processing in the keystream generator also benefits resource usage, although initialisation to mix the keystream generator's state after re-keying is still required to avoid security vulnerabilities.

Generating the complexity necessary to securely imitate a random stream with a keystream generator has proved to be a challenging task for cryptographers with

a number of high profile schemes succumbing to vulnerabilities [51, 52]. However, as will be seen when we look at stream ciphers in greater depth in Chapter 4 the resource efficiency of this form of symmetric encryption appears to be a good match to that of protecting data in a restricted hardware environment and newer algorithms would appear to resolve some of the security concerns. Indeed the recent stream cipher SNOW 3G specified for confidentiality protection [53] in third generation mobile phones not only shares a setting similar to that intended for secure display but has also proven resistant to attack.

3.4 Asymmetric Encryption

A further type of cryptographically secure encryption exists where the parties involved in a transmission do not share the same key and instead have their own individual keys. This asymmetric situation is very different to symmetric encryption in that no secret need be transmitted. The actual encryption process revolves around each party generating a fixed length public key which may be used by any other party to securely send it information but which on encryption becomes impossible for any party apart from the receiver to decrypt. This concept of a public key means that asymmetric encryption is more commonly referred to as public-key cryptography.

Public key cryptography relies on mathematically hard problems that are easy to solve in one direction but are extremely difficult to solve in the reverse direction. Two problems have typically been used: the factorisation problem, of trying to determine factors of a large integer; and the discrete log problem, of trying to determine the exponent of a value for a given base. Early public-key schemes making use of these problems would respectively be Rivest, Shamir and Adleman's method (RSA) [54] and El Gamal's method [55].

The price paid for the one way nature of asymmetric algorithms and the associated channel security is that of computation cost. Both encryption and decryption require computational intensive operations, such as modular exponentiation (calculating $c = m^e (mod n)$) to perform the underlying mathematics making them orders of magnitude slower than symmetric key based ciphers and unsuitable for use in high volume data applications. However, for small data quantities such as session key transmission and email, these slow speeds are a perfectly acceptable exchange for being able to transmit securely over an unsecured channel. In fact, asymmetric encryption is of critical importance for key sharing as well as identity authentication and therefore

its implementation is desirable for almost all data security devices. Indeed Diffie and Hellman, who actually introduced the idea of public key cryptography to the research community back in 1976 [56], focused their initial scheme [55] at the area of key agreement.

As is touched upon in Section 6.2, the security of the key length used in public encryption schemes has been subject to much analysis, and this increased understanding of their vulnerability has led to a growth in key sizes used. Currently it is thought that the RSA algorithm requires a 3072-bits to achieve a similar security level to a 128-bit keyed symmetric cipher [57] *(pp.19)*. Storage and manipulation of these large values and the need for the costly modular exponentiation operations are a significant barrier to the implementation of the discussed public-key cryptography schemes in resource constrained environments such as smart cards.

In view of this, another approach to public-key cryptography was developed by Koblitz and Miller based upon the harder elliptic curve discrete log problem [58, 59]. Although still requiring complex mathematical operations, the modified problem allows elliptic curve cryptography (ECC) schemes to utilise smaller 256-bit key sizes when providing an equivalent level of security, enabling more efficient implementation. Nonetheless both ECC encryption and decryption performance still fall well below the capabilities of symmetric encryption schemes with Dormale and Quisquater [60] indicating high speed implementations require 35kGE.

3.5 Random Number Generation

Universal in both asymmetric and symmetric encryption is the importance of the key in providing a system with security. Data protection schemes therefore demand a high quality of random variable be used for this value and so in the penultimate section to the chapter we briefly explore the concept of randomness and random number generation.

3.5.1 Randomness

To understand the security provided by cryptography a good foundation is needed in the idea of randomness. In the real world something is fully random if events are entirely unpredictable: often radioactive decay is described as an example of

this since a nucleus will decay at a time independent of other nuclei and in an unpredictable fashion. While undoubtedly true, this source of randomness does not satisfy the cryptography definition of a random source due to the exponential distribution of events with time.

In cryptography, ideal randomness means that not only do outcomes of events have to be independent, but also their probability distribution must be flat and equal. In the digital world this translates to the outcomes *logic 1* and *logic 0* each having a probability of 0.5 and that successive events be independent of all previous events. The information entropy [61] content of each bit of such a binary stream is 1-bit, meaning that it is impossible to deduce any outcomes by obtaining another part of the stream and it is not possible to use compression on the stream as the information content is already 100%.

In cryptographic encryption it is aimed to produce a ciphertext that closely mimics this ideal random stream of bits, while ideal randomness is also desired for key generation due it rendering all key guessing attacks no more effective than exhaustive search. The lack of ideal or at least near ideal randomness is often the source of failure in cryptography schemes, with the randomness failings of the MD4 implementation in Netscape [62] being one well publicised example.

3.5.2 Random Number Generators

To produce a truly random bit-stream, cryptographic applications use a dedicated random number generator (RNG) to process some fundamental noise source into a usable form. It is desirable to avoid any form of predictability in this source and so typically thermal noise, shot noise and flicker noise are used, harnessing the randomness of events at the quantum level.

With such low level noise sources and the need to avoid bias in the generated output, the hardware implementation of a RNG is relatively complex. Typically it requires mixed-signal design with the combination of resister noise amplification and oscillator jitter detection being a common implementation [63, 64, 65]. Designs of RNG which avoid the need for amplification have also been suggested, including utilising flip-flop metastability [66, 67, 68, 69, 70] and transistor based electron traps [71]. To remove any remaining bias and aid the resistance of the RNG to active attacks post processing logic is used in fielded designs; this as in the Von Neumann corrector [72], where only differing pairs of bits are processed, will usually reduce

the output rate of the RNG as well as make it irregular.

Much of the recent interest in RNG design has focused towards their use in smartcards and implementation results by Bucci [64] on these devices have indicated that high quality random number generation uses around a quarter the area needed for a compact AES implementation [73]. RNG use in the equally restricted application environment of secure display should have a similar associated cost.

3.6 Discussion

This chapter has considered a wide variety of protection schemes used to secure commercial multimedia data in a consumer environment as well as a number encryption types that are cornerstones of modern cryptography. It is seen that the history of multimedia protection schemes has been littered with vulnerabilities and failures, again highlighting the difficulties faced in securing multimedia transmissions in a secure display system. These failures have been particularly prevalent among pseudo-secure and passive security systems where keys cannot be revoked. However, the fully secure active systems of recent commercial pay-TV are also unsuited to a secure display setting, due to high processing requirements.

The standard cryptographic forms analysed offer greater potential for use in the decryption function block of the proposed secure display system. Symmetric encryption is well suited to the task, processing high throughput streams of data; be this in small blocks, as a block cipher, or as individual characters, as a stream cipher. From a security perspective the use of a block cipher to protect display data has many attractions as there is a great deal of confidence in the security of these modern algorithms, due to their wide use and well understood structure.

Stream ciphers have had, as noted in Section 3.3.2, numerous high profile algorithmic vulnerabilities and understanding of the issues surrounding secure keystream generation has been at a very embryonic stage. These ciphers do, however, have inherent characteristics relative to block ciphers, of low resource usage, high throughput efficiency, and low power consumption, that makes them particularly attractive to the constrained environment of secure display. Furthermore, the lack of latency seen in stream cipher decryption, makes them ideally suited to time-critical data applications such as live video streaming and user interaction scenarios where fast response times are critical. The ability of the stream cipher to reuse identical hardware

for decryption and encryption may also provide a small additional resource efficiency benefit to the application of secure display in data entry panels (Section 2.2.2).

The attractive characteristics of stream cipher hardware implementation makes this category of cryptographic primitive the preferred candidate for use in secure display if new secure hardware focused designs can be realised. In late 2004 an announcement was made [74] of a new stream cipher competition called 'eStream' which aimed to explore just such secure and efficient algorithms. This competition and a more generalised discussion of hardware stream cipher design forms the topic of the next chapter.

Chapter 4

eStream

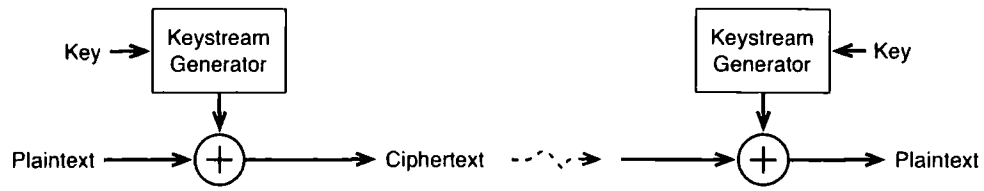
SPEED and efficiency have long been temptations of cipher designers and it is the stream cipher's potential to offer both that has been captivating cryptographers since its conception almost a century ago. However, recent security failings in a number of high profile primitives has led to the validity of the stream cipher as a useful form of encryption being questioned.

This chapter charts the rise and fall of the stream cipher and outlines the potential road to recovery offered by the recent and ongoing European effort, eStream. A number of new cipher proposals are considered in detail so as to both assess their potential for usage in secure display and to determine whether the algorithms offer sufficient advantages for them to be chosen over the widely used AES block cipher.

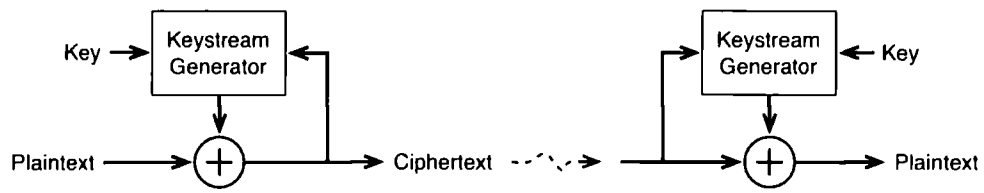
4.1 Introduction to the Stream Cipher

As mentioned briefly in Section 3.3.2, the stream cipher is a core form of cryptographic primitive that is capable of providing symmetric encryption. Its structure is based around two core parts: the keystream generator, producing a pseudo-random sequence of bits that forms the keystream; and the combining function, mixing this stream with an incoming plaintext stream to encrypt, or an incoming ciphertext stream to decrypt. Stream cipher combining functions are generally chosen to be self-inverting so the decryption process will be identical to that of encryption. A diagram of the stream cipher encryption structure can be found back in Figure 3.2.

Two categories of stream cipher exist: synchronous and self-synchornous. The



(a) Synchronous stream cipher



(b) Self-synchronising stream cipher

Figure 4.1: Stream cipher types and their method of operation

synchronous stream cipher is the most common form and uses only key material for its generation of the keystream, while the less common self-synchronising stream cipher uses both key material and previous ciphertext symbols. An overview of the operation of synchronous and self-synchornous stream ciphers is shown in Figure 4.1. As it is almost universal in modern stream ciphers to use the binary additive form of keystream mixing, both figures are shown with an XOR operation replacing the combining function.

The independence of a synchronous stream cipher's keystream generation from the message data provides this type of cipher with a number of properties not present in block ciphers. These include resistance to both chosen-plaintext and ciphertext attacks, and an ability to prevent bit-flipping errors in a transmitted ciphertext symbol from propagating into any other plaintext material except that of the corresponding plaintext symbol. The self-synchronising stream cipher's ciphertext input to the keystream generator means that it is data dependent, and thus possesses none of these properties. However, it enjoys another unique property — that of an ability to resynchronise with the incoming ciphertext stream after an insertion or deletion error occurs in transmission. For this to be useful the self-synchronising cipher's keystream generator must lose the complete time dependance of the synchronous cipher, so that after a fixed number of ciphertext symbols sufficient information is obtained to regain synchronicity.

Internally the keystream generator of both synchronous and self-synchronous stream ciphers typically contain three main elements: a state memory, which stores

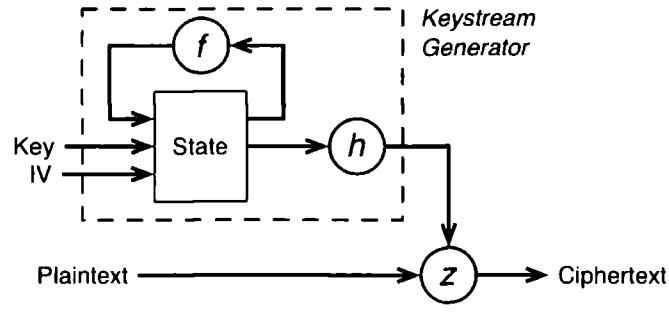


Figure 4.2: General structure of a synchronous stream cipher

information on the generator's current state; an update function f, which updates the state memory in a manner that provides complexity to the system; and a filter function h, which disguises the keystream output so that state information remains secret. The generalised structure of a synchronous stream cipher's keystream generator is shown in Figure 4.2. For a self-synchronous stream cipher this structure will differ substantially, with the state update function essentially being the entire cipher, as a result of the inclusion of ciphertext.[1]

As previously mentioned, stream cipher keystream generation is dependent on the symmetric key. However, also shown in Figure 4.2 is a further insertion to the keystream generator in the form of an initialisation vector or initialisation value (IV). This additional public variable is mixed with the key material at the keystream generator's initialisation, making the cipher's starting state a unique combination of these two values. Multiple IVs can be used for any single key with the resulting advantage that the secret key may be retained for more than one transmission — thus avoiding the costly overhead of repeatedly performing secure key exchange. In frame based communications, where the IV is typically taken directly as the frame number, this advantage is particularly pronounced, leading to widespread acceptance of IV use. The incorporation of IVs into the keystream generation process has a notable further consequence, with the ease of frame retransmission caused by the inclusion of an IV diminishing the impact of synchronisation errors in synchronous cipher schemes — effectively leaving the primary advantage of the self-synchronising stream cipher nullified.

[1] More details on self-synchronous stream cipher structures may be found in the Handbook of Applied Cryptography [25] *(pp.194–195)*.

4.2 Cipher Categorisation

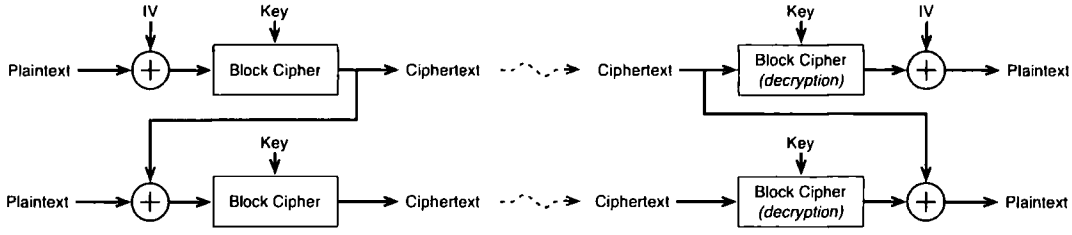
The difference between stream ciphers and block ciphers as described in Section 3.3 would appear significant. However, the distinctions can become muddied depending on how we choose to perceive the algorithm's structure. Indeed a stream cipher can be considered as block cipher with a very small block size, and a block cipher likewise as a stream cipher with a very large character size.

This overlap in the distinction made between forms of symmetric encryption is supplemented by some of the usage configurations seen in these ciphers. Self-synchronising stream ciphers are a prominent example of this — they have a data dependant output characteristic more usually associated with block ciphers, but equally they retain the time dependent property of a stream cipher that means encrypting the same block of data with the same key may produce differing outputs due to the dependence on the sequence of previous operations. Similarly, it is unusual to utilise block ciphers in their base form — the electronic codebook (ECB) mode — due to its propensity to leave data patterns in the output ciphertext. More commonly cipher-block chaining (CBC), cipher feedback (CFB), counter (CTR) and output feedback (OFB) modes of operation [75] are used (Figure 4.3), all of which make the encrypted output sequence dependant. The later two modes, CTR and OFB, actually transform the block cipher into a synchronous stream cipher, while CFB mode instead gives the resulting cipher self-synchronising properties.

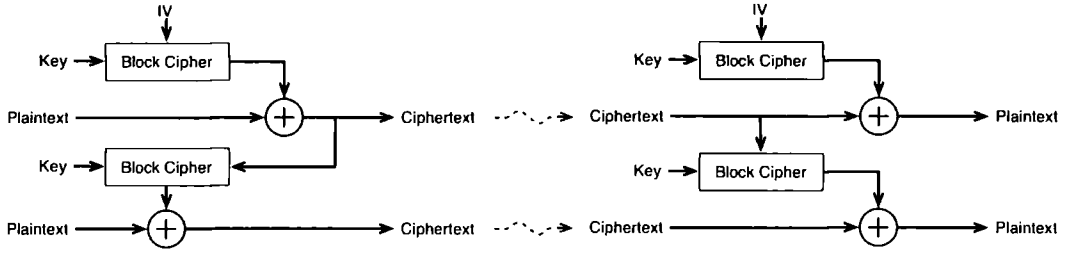
This uncertainty as to categorisation also affects consideration of the origins of the primitives themselves — for instance it can be argued that the 16th century Vigenère cipher is a form of stream cipher (see Section 4.3.1). It is notable, however, that this cipher is not binary additive, as data and keystream are combined on a character by character basis with the letter output dependant on an alphabetic rotation. As this operation provides much of the complexity to the Vigenère cipher, it is probably more correctly categorised as a block cipher with a key being mixed with a block of data to produce an equal length ciphertext output.

A number of cryptographers have attempted to develop better definitions of synchronous and self-synchronising stream ciphers, with two of the more promising being provided by Ekdahl [76]. By combining both these definitions it is possible to derive a complete definition for the extent of stream cipher encryption primitives:

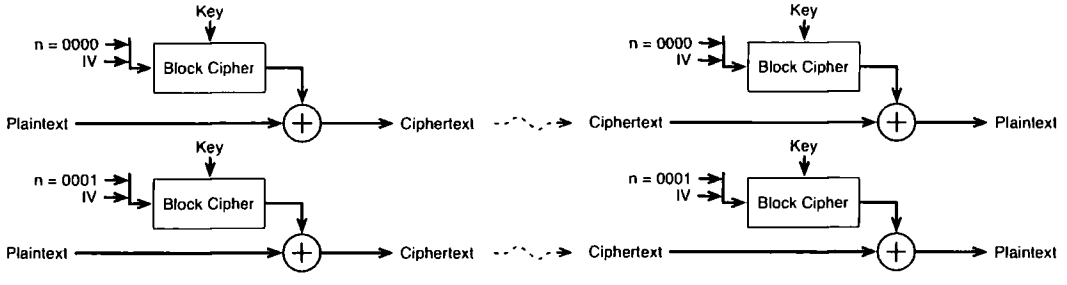
Definition 4.1: *A stream cipher is a finite state machine for which the keystream output is generated from the key and a fixed number of previous ciphertext symbols,*



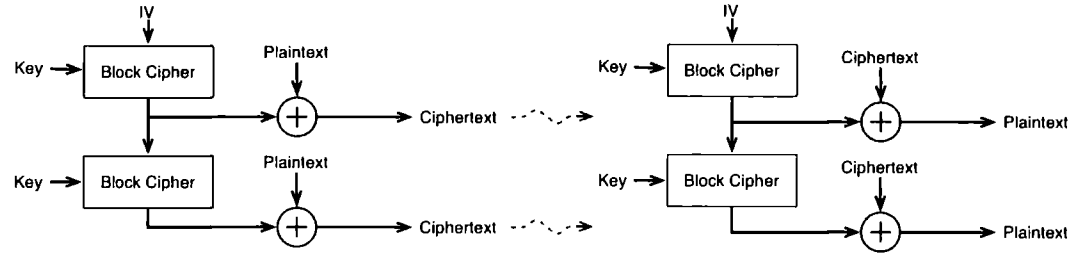
(a) Cipher-block chaining (CBC)



(b) Cipher feedback (CFB)



(c) Counter (CTR)



(d) Output feedback mode (OFB)

Figure 4.3: Block cipher additional modes of operation

but independently of the plaintext message.

Under this definition the synchronous stream cipher is seen as special case of the self-synchronising stream cipher, where the fixed number of ciphertext symbols has been set to zero. It will be noticed that despite not defining the combining function the simple exclusion of plaintext material immediately excludes all block ciphers except when used in the CTR, OFB and CFB stream cipher style modes of operation, and defines Vigenère's autokey cipher [77] as being distinct from a stream cipher.

A further curious consequence of Definition 4.1 is that the recent stream cipher Phelix (see Section 4.6.5) is not classified as being a stream cipher, due to its plaintext input to the keystream generator. The performance characteristics outlined later in Section 4.2(c) and the cipher's overall structure would indeed seem to fit more closely into the category of a block cipher — perhaps confirming that this definition is still accurate. Nonetheless, it is probably reasonable to continue to refer to Phelix as a stream cipher, as when the algorithm's authentication functionality is not needed the offending plaintext input can be eliminated.

4.3 Stream Ciphers and History

The origins of stream ciphers are as uncertain as their definition, with the phrase notably absent from Kahn's 1967 comprehensive history of secret communications [78]. Nevertheless, the origin of the concepts behind stream ciphers would seem to lie many years prior to this date, and so a brief examination of their early history is useful.

The stream cipher's recent history is also shrouded in mystery. Their application has mainly been in the military and telecommunication industries, where there is a strong incentive to keep primitives confidential: an example of this reluctance to publish stream cipher schemes can be seen as recently as the 1990s with the GSM standard for mobiles (see Section 4.4.1). Laterally however, security concerns and a drive towards greater openness in the telecommunications industry have meant that a number of stream cipher designs have been publicly disseminated.

4.3.1 Early History of Stream ciphers

In 1553, [79] Giovan Belaso would define a form of encryption that would later become know as the Vigenère. This cipher was one of the first primitives that could truly be called a stream cipher, with its operation involving the generation of a keystream simply by repeating the key, and a combining function consisting of the alphabet based modulo-26 addition of individual plaintext characters with keystream characters. The balance of complexity provision in this structure — as mentioned in Section 4.2 — leads however, to the conclusion that it is better described as a block cipher. Nonetheless, as a form of encryption it would remain publicly secure until the 19th century.

The next stage in the development of stream ciphers would occur during World War I with the development of the Verman cipher [80]. This cipher, based around a paper-tape driven electromechanical device, is arguably the true origin of the stream cipher, and its design would later be outlined in Gilbert Vernam's 1919 patent [80]. The Vernam cipher introduced the idea of mixing plaintext with keystream material through the use of a bit-wise XOR combining function — in this case electric relay based. It also brought automation to the encryption process by using 5-bit Baudot coded characters throughout the system and tape inputs for key and plaintext material.

One significant aspect of stream ciphers that was missing from the Vernam design was any sort of complexity in the keystream generation process, with the output keystream formed directly from key material. This direct use of the key would lead to Vernam having a number of problems in managing long reels of key material,[2] but would also lead to one of the most important developments in cryptography. Repeated use of key material and the use of non-random key material had not specifically been prohibited by Vernam, and according to [25] *(pp.274)* shortly after seeing the cipher U.S. Army cryptologist Joseph Mauborgne suggested these modifications to improve its security. The resulting cipher — the one-time pad — has since been proved to be theoretically unbreakable [40] and is thought to be used in some high level communications even to this day.

Subsequent to the Verman design, ciphers moved to a greater range of automation and a number of rotor based cipher machines were developed for use with teleprinters. These used various sized rotating wheels to provide a long period keystream which

[2] To overcome the problem of long tape reels Morehouse [81] would in 1918 propose generating keystream through the combination of two short tapes that were relatively prime in length.

could be used for polyalphabetic substitution to encrypt transmissions. Various designs existed for these machines but a prominent example that shared much with the modern synchronous stream cipher was that of the 1940 Lorenz SZ 40 cipher.[3]

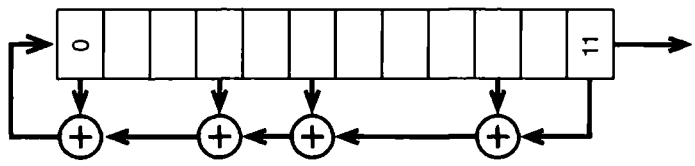
This early Lorenz cipher was used by the German Army High Command during World War II and had 12 rotor wheels of varying sizes forming its state. Five of these would be rotated incrementally while another five would be variably clocked depending on an output generated by the remaining two. A long period, pseudo-random keystream produced by combining the incremented and variably clocked wheel sets was then mixed with incoming plaintext characters through a bit-wise XOR operation to form ciphertext characters on a 5-bit wide teleprint. The initial rotor positions set by the signal operator were effectively the cipher's IV. The Lorenz cipher in its later SZ 42 form actually became an autokey cipher and used plaintext information along with the key to generate keystream in a fashion not dissimilar to a self-synchronous stream cipher's use of ciphertext.

The adjustable pinwheels used in both machines had sizes $\{43, 47, 51, 53, 59, 61, 37, 41, 31, 29, 26, 23\}$ giving an effective state size and key length of 501-bits while an IV based on the 1.6×10^{19} starting positions was used to maintain the practicality of regular message encryption. However, the infrequent alteration of wheel patterns, the alphabetic labelling of pinwheel positions, the pairing of wheels for each of the 5 teleprinter lines and the limited inter-linking through the variable clocking would mean the strength of the cipher was much lower than these figures appear to indicate. The lack of true randomness, repeating messages and use of repeated settings would indeed allow cryptographers at Bletchley Park in the U.K. to develop a full understanding [82, 83] of the Lorenz machines without actually seeing one and would further enable many of the initial settings to be retrieved. These key recovery attacks developed throughout the war were famously automated in 1944 by the first electronic digital computer, Colossus — a topic further discussed in Section 6.2.

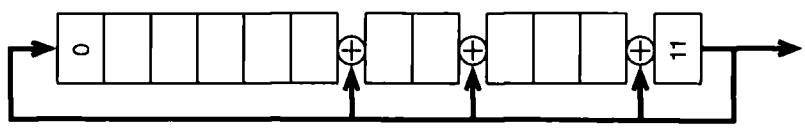
4.3.2 Linear Feedback Shift Registers

The advent of digital electronics meant that the rotors of the teleprint machines were replaced by the binary logic structure of a Linear Feedback Shift Register (LFSR) which is both easy to implement and capable of producing large period pseudo-random streams with good statistical properties. Indeed in the United States

[3] The slightly earlier Siemens and Halske T52 cipher also shared many characteristics with modern stream ciphers though it included an unusual state driven transposition at its ciphertext output.



(a) Fibonacci LFSR



(b) Galois LFSR

Figure 4.4: Linear Feedback Shift Register structures

this component formed the basis of secure government teletypewriter communications from 1955 onwards through the use of the electronic KW-26 machine [84]. The cipher developed for the KW-26 is one of the first recognisable stream ciphers, using a LFSR driven keystream generator, a modulo-2 binary addition combining function similar to the Vernam cipher and a fixed length key from a punched card.

The LFSR itself is based around a shift register tapped at points along its length so as the next input is a linear function of its state. If these tap points are chosen carefully and seeded with a non-zero starting state it is possible to produce a maximal sequence of the input bits that produces every non-zero combination possible in an n bit register and results in a period $(2^n - 1)$ in length. Two types of tapping structure exist; the Fibonacci LFSR: where shift register taps are combined every clock cycle through an XOR operation and fed back into the input; and the Galois LFSR, where the shift register output directly provides the input while also altering the shifting process at each tap point via an XOR operation. The two types of LFSR are shown in Figure 4.4.

The Galois structure has an advantage over Fibonacci LFSRs in that the need to combine multiple taps is avoided, thus reducing the propagation time between states to that of a single XOR. Despite this, the regular shift register structure of the Fibonacci configuration means it is commonly favoured.

Both LFSR configurations have a capability for use as a stream cipher's pseudo-random keystream generator where the chosen tap positions form the secret information of the key. However, the Berlekamp-Massey algorithm developed in 1968 and

later presented by Massey [85] can attack the linearity of these structures and this has seen their direct use for keystream generation fall out of favour. This iterative algorithm requires just twice the state size in outputs to resolve the tap positions and effectively forces ciphers founded on the LFSR to maintain huge state registers half the message length long if they are not to be vulnerable. As such, in its pure form an LFSR based stream cipher provides little practical advantage over that of the one-time pad.

Attempts have been made to suggest periodic functions capable of replacing the LFSR [86] but the good statistical properties, long period and hardware efficiency of the LFSR has meant it has stubbornly remained as the underlying component behind most stream cipher update functions to this present day.

4.3.3 Other Update Functions

With the insecurity of practical keystream generators based directly on the LFSR, cryptographers have looked at other ways to improve the complexity of keystream generation. A primary focus has been on destroying the linearity present in the LFSR through the addition of a non-linear element. Various functions have been proposed to achieve this, including: non-linear feedback shift registers, where the feedback to the shift register is altered to no longer be a linear function of the tap point values; non-linear filters, where the output of an LFSR is passed through a non-linear function before being sent to the keystream output; non-linear combination generators, where multiple LFSRs are combined in a manner designed to increase complexity; and variable clock control where the LFSR is no longer clocked in a regular fashion. The theory and implementation of these mechanisms are outlined in much greater detail in the Handbook of Applied Cryptography [25] *(pp.202–212)* but many can also be seen in the various stream ciphers examined later in this chapter.

4.3.4 RC4

While derivatives of the LSFR are efficient to implement in hardware, their efficiency in software is much less pronounced. So when Ron Rivest of RSA Laboratories came to developing a software stream cipher in 1987 [87] he did not use the LFSR device at all, instead relying on byte manipulations in the permutation of its internal state. This provided great software efficiency and led to RC4 becoming one of the

most widely used stream ciphers, with applications as varied as securing internet communications [88], wireless networks [34, 89] and document files [90]. However, despite this popularity, the history of the cipher has been far from untroubled.

The algorithm itself was never actually published, remaining a secret until anonymously being revealed in 1994 [91]. It used a range of key sizes from 40–128-bits and had a state size of 2048-bits which was arranged into 256-bytes. These bytes are iteratively swapped and summated during every cycle with the modulo-256 summation result forming the keystream output. The initialisation of the cipher would take 256 cycles. In software terms, RC4 was highly efficient — using fewer than 80 lines of code and operating at high speed. However the large state size was a significant hindrance to its hardware implementation.

In terms of security, the main problems with RC4 would come from its weak initialisation process, which led to a number of effective attacks focused on detecting bias in the first few bytes of outputted keystream [92]. The vulnerability would be increased by the fact that RC4 lacks any form of IV input mechanism, so when, in wireless network protocols such as WEP (Wired Equivalent Privacy) [34], it was desired to encrypt new data without a costly re-keying process, simple concatenation of the key and IV was chosen — with devastating consequences [51, 93, 94].

These related key attacks on WEP would eventually lead to the development of the Wi-Fi Protected Access (WPA) that implemented most of the IEEE 802.11i standard [89]. This temporary standard still used RC4 but saw the addition of the Temporal Key Integrity Protocol (TKIP) to its front end which provided a number of features, including a key distribution management structure, to avoid the weaknesses in WEP. TKIP crucially used a hash of the 128-bit secret key and IV to generate unique keys for each data packet instead of simply prepending a 24-bit IV to a 104-bit or 40-bit key to form the RC4 seed as in WEP. WPA was not ideal from a performance perspective, but did provide a temporary security solution that would allow RC4 legacy hardware to be reused. The subsequent full implementation of the IEEE 802.11i standard WPA2 would see the introduction of block cipher, AES, based encryption.

In recent years attacks against WEP and RC4 have grown ever more effective [95, 96] and there have even been attempts to attack the RC4 keystream generator itself [97] although this has remained relatively secure.

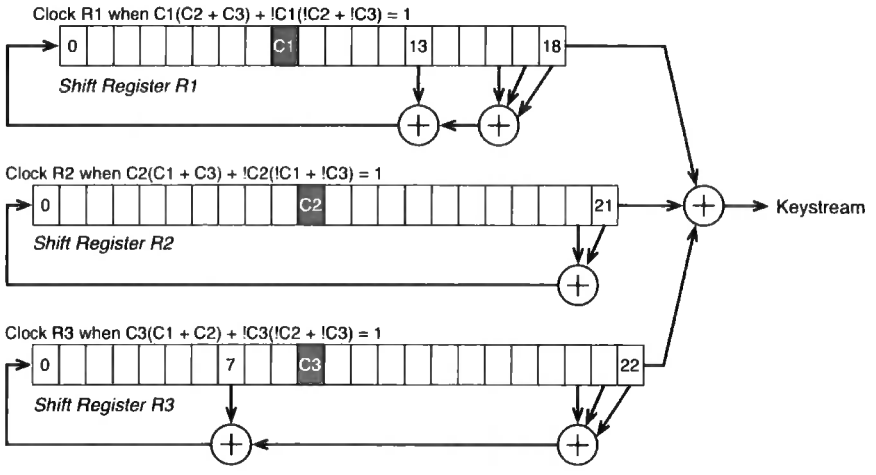


Figure 4.5: Schematic description of the A5/1 keystream generator

4.4 Stream Ciphers in the Modern World

Following RC4 a number of other high profile stream ciphers would be fielded in the mobile communications industry: A5/1, used in GSM communications; E0, used in Bluetooth; and SNOW 3G, used in third generation mobile phones. Each of these ciphers' history and hardware implementation are examined in detail in this section to give an overview of the issues affecting stream cipher use in modern mobile devices.

4.4.1 A5/1

The A5/1 stream cipher was developed for use in the GSM (Global System for Mobile) standard for mobile phone communications to encrypt transmissions between the base station and handset. It saw widespread use during the 1990's and is still found in many modern devices to this day. Like RC4, the details of the cipher were never officially published, instead being leaked [32], and later reverse-engineered [98].

A5/1 is a very compact, hardware focused, synchronous stream cipher which was designed to securely encrypt the small data frames that form GSM communications. A5/1 is intended to produce two 114-bit keystreams for each frame and uses a 64-bit key in conjunction with a 22-bit frame number that operates as an IV. The design of its keystream generator (Figure 4.5) consists of three linear feedback shift registers of lengths 19, 22 and 23, providing a state size of 64-bits. Each of these registers are clocked dependant on their own control bit being in the majority and it is this irregular clocking that helps give A5/1 a degree of non-linear complexity.

In operation the cipher is initialised by setting the registers to zero and then, with the irregular clocking deactivated, feeding in the 64-bit key a bit at a time to the LFSR feedback loops via an XOR operation. Following this the 22-bit IV is similarly fed in before the whole cipher is clocked for 100 cycles with the irregular clocking activated and the output disabled. The keystream that is outputted for the next 228 cycles is collected and XORed with the GSM frame data.

The first attack on A5/1 was presented by Golic in 1997 [33] and targeted the cipher's small state through a time-memory trade-off based arrangement. Shortly after the full reverse engineering of the specification in 1999, [98] two more refined attacks against the cipher were found: Biham's guess and determine attack [99] requiring 2^{40} operations, a similar number of precomputations and 32GB of memory; and Biryukov's [100] short active time attack based again on time-memory trade-off and requiring 2^{42} pecomputations and 300GB of memory. Subsequent to this a number of very practical attacks have been described against A5/1, including a correlation attack by Ekdahl and Johansson that takes just a few minutes of precomputation on a standard PC and requires 2^{16} data frames. As such, the A5/1 stream cipher is now considered insecure, although the continued strong user base of the GSM network means its withdrawal is impractical.

4.4.2 E0

The synchronous stream cipher E0 is widely used in short range, low power communication devices with the cipher being created for the Bluetooth specification [35] to encrypt packet payload data. Originally specified in 1999 the security of this compact cipher was compromised [101] in only one year and by 2005 practical conditional correlation attacks would be being specified against the cipher [102] using just 2^{24} data frames and 2^{38} computations.

The cipher itself uses four LFSRs of length 25, 31, 33, and 39-bits to provide a long period and a small 128-bit state. E0's keystream is generated by combining the outputs of these shift registers with a function calculated by the very same outputs (Figure 4.6). The summation combiner and blend function make up a finite state machine that introduces crucial non-linearity into the system.

In operation the E0 has quite a complex initialisation procedure which takes the agreed 128-bit key and feeds it through a function involving the taking of a modulus of individual bytes and ANDing them with a constant. This resulting 128-bit

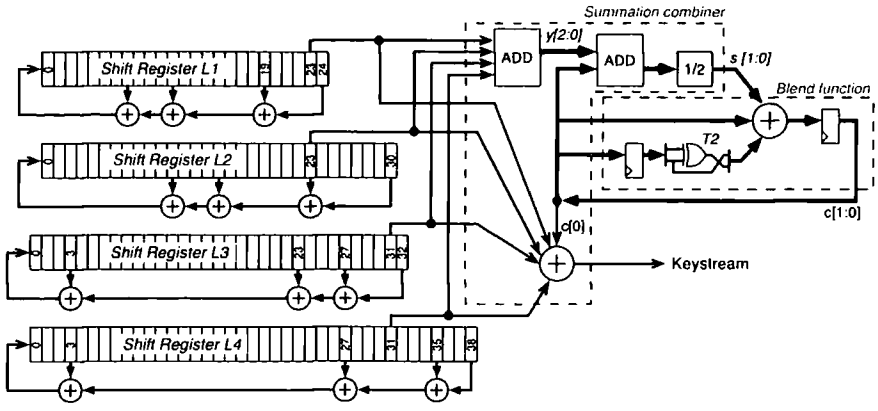


Figure 4.6: Schematic description of the E0 keystream generator

cipher key, combined with an IV consisting of a 48-bit bluetooth device address, a 26-bit clock time and the constant 111001, are then shifted into the four LFSRs with feedback initially deactivated. As each LFSR is filled, that register's feedback is activated and on the filling of *L4* the blend function's registers are set to zero before a further 200 cycles are used to input the remaining bits and mix the state. The last 128-bits of produced output is used to refill the LFSRs in a parallel load operation before the cipher finally outputs its first bit of keystream.

Despite the insecurities in the E0 cipher other security concerns in the Bluetooth specification have been of more pressing concern to the Bluetooth Special Interest Group with the recent enhanced data rate version of the specification concentrating on addressing the user security issues in areas of key agreement and key strength [103].

4.4.3 SNOW 3G

As has been illustrated by the discussions of the ciphers RC4, A5/1 and E0, few of the stream ciphers released in the recent past have survived the security tightrope. SNOW 2.0 [104] is one of the more efficient exceptions to this trend, making it an attractive candidate for ISO standardisation and, in its later incarnation as SNOW 3G [53], widely used in 3GPP UTMS mobile phones.

The origins of SNOW 3G and SNOW 2.0 lie in the synchronous stream cipher SNOW [105] first proposed in 2000 as part of the NESSIE evaluation project (see Section 4.5.1). The structure of SNOW, is shown in Figure 4.7 and is based around a LSFR providing a long period to the internal state and a finite state-machine

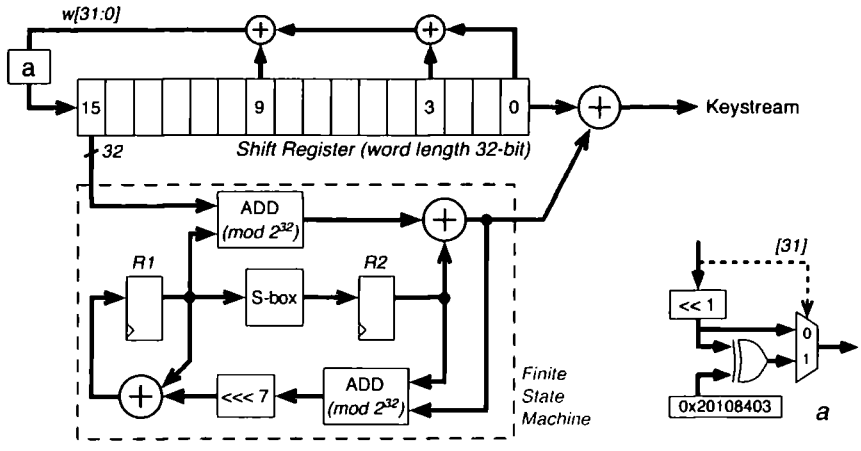


Figure 4.7: Schematic description of the SNOW keystream generator

manipulating this in a non-linear way to provide sufficient unpredictability to justify 128-bit and 256-bit key lengths. The cipher also specifies a 64-bit IV which when used requires an initialisation period with the keystream disabled of 32 cycles.

The novel aspect of SNOW in comparison to similar software stream ciphers is that all operations take place on 32-bit blocks of data instead of single bytes or bits — this allows software to work in a word-orientated manner and facilitates very efficient processing of state updates. This wider register structure has a number of negative effects on hardware implementation in that it creates a large state size for the cipher of 576-bits while increasing the complexity of the modular addition operations and the substitution-box transformation in the finite state machine. Nevertheless, in applications requiring purely throughput efficiency, the 32-bit/cycle keystream output largely mitigates any area disadvantage.

The design goals outlined by the authors [105], of being faster than the AES in software and having a lower implementation cost in hardware, would largely be met by this primitive. The final goal however, to maintain a similar security level, was, at the second phase of the evaluation process, proved to be unfulfilled. Two insecurities were found in the form of a distinguishing attack [106] and a guess and determine attack [107], and although most likely impractical to implement, their complexity — at least in the case of the former attack — falls below that of exhaustive search. As a result, in 2002 a modified version of the cipher was suggested in the form of SNOW 2.0 [104].

SNOW 2.0, like its predecessor, uses a keystream generator structure based around a shift register and finite state-machine as shown in (Figure 4.8). The cipher takes both 128-bit and 256-bit keys as well as a lengthened 128-bit IV and again operates

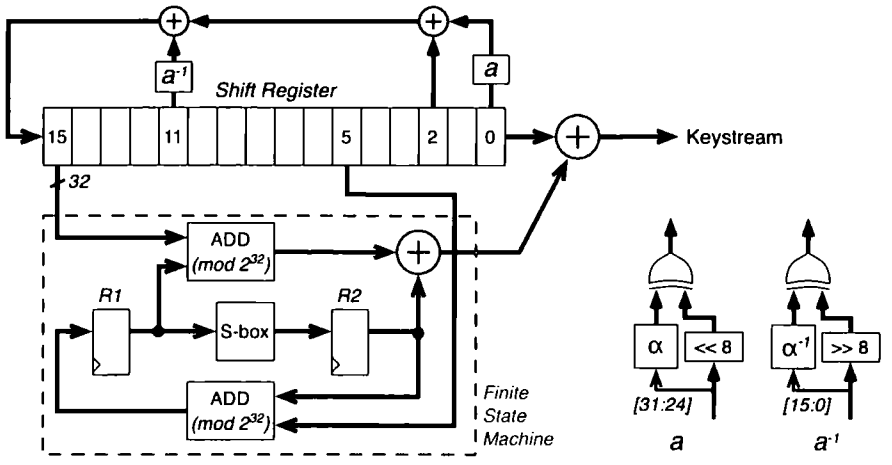


Figure 4.8: A schematic diagram of SNOW 2.0

in a word orientated manner to maintain software efficiency. Initialisation involves filling the shift register with the key and IV material pre-combined through a specified XOR operation and then clocking the cipher 32 times before the first word of key stream is outputted. The maximum number of keystream words produced before re-keying is set at 2^{50}.

The main change from the original version is in regards to the feedback loop, where efforts were made to improve the spreading of bits and leave the cipher more resistant to attack. The cipher has an altered feedback polynomial and also uses two constants instead of just one, with their move inside the feedback loop noted to actually improve the cipher's speed. The other significant changes are in the non-linearity providing finite state machine which takes a second input from the shift registers to prevent the guess and determine attack of [107]. SNOW's authors, Ekdahl and Johansson, would also change the S-box to that of the AES to provide improved diffusion.

The subsequent five year period has proved SNOW 2.0 to be one of few stream cipher algorithms which can reasonably be described as secure and has acceptable efficiency in hardware and speed in software. It has undergone much cryptanalysis in this time [108, 109, 110], but the attacks even against the 256-bit key version have produced little in the way of results and require too much data to be practical. Indeed this confidence has led to SNOW 2.0 becoming part of the ISO/IEC 18033-4 standard [111] as a recommended stream cipher primitive. Its high performance in software has also led its inclusion as a reference stream cipher for software performance [112] in the eStream Project described in the remaining sections of this chapter. As can be seen in [113] and Section 4.7.5 the software performance of SNOW 2.0 still compares

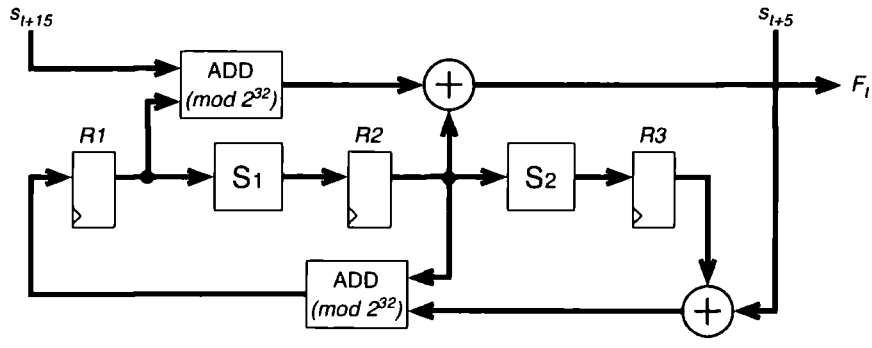


Figure 4.9: A schematic diagram showing the modified finite state machine of SNOW 3G. In comparison to SNOW 2.0 an extra S-box (S2), register (R3) and bit-wise XOR operation have been introduced

favourably with many more recently proposed designs.

SNOW 3G [53], shown in Figure 4.9 is based directly on SNOW 2.0 with only an extra S-box added by the 3rd Generation Partnership Project (3GPP) to improve its resistance against algebraic attacks. The algorithm finds widespread use in third generation mobile phones where it is used for the confidentiality and integrity functions f8 and f9 of the 3GPP UTMS (Universal Mobile Telecommunications Standard) [53]. However, after the mobile phone industry's previous experience experience with GSM and the weaknesses in A5/1, a very cautious approach has been taken and the standard also accommodates a variation on the MISTY1 block cipher [114], KASUMI, as an alternative to providing confidentiality and message authentication of user data.

As demonstrated by the implementations in [53] SNOW 3G is about 50% larger than SNOW 2.0 and operates with very similar performance, although both have a throughput 4 times that of the specified block cipher KASUMI. These figures are less impressive when compared to hardware focused algorithms which tend to have the advantage of a much smaller state size. Nevertheless, as seen in Table 4.2(c) later in this chapter, the performance of SNOW 2.0, even in hardware, is comparable to all but the most efficient stream cipher designs.

4.5 The eStream Project

Beginning in 2004 and predicted to finish in 2008, the eStream Project is a multi-year attempt by the ECRYPT network of excellence to generate new stream cipher primitives and to comprehensively examine the field of cryptography surrounding stream

ciphers. It is hoped that the resulting designs will be suitable for widespread adoption and will provide benefits to a large number of communication-based applications.

4.5.1 The Failure of NESSIE

In 2000 it was decided to review the primitives and protocols used in data security systems in a project called NESSIE (New European Schemes for Signatures, Integrity and Encryption), with the aim of recommending methods for seven different areas of cryptography: block ciphers, stream ciphers, message authentication codes, asymmetric encryption, digital signatures and digital identification. Seven stream ciphers in all were considered BMGL, SNOW, SOBER-t16, SOBER-t32, LEVIATHAN, LILI-128 and RC4 — however, in the final report [115] the NESSIE consortium found that although it could recommend algorithms in six of the seven areas, a recommendation in the remaining area of stream ciphers was not possible [116]. A comment was made by one of the reports authors that "it is quite remarkable that none of the six submitted stream ciphers meets the rather stringent security requirements put forward by NESSIE." [117].

4.5.2 A Way Forward

In response to this failure to find an efficient and secure stream cipher, the ECRYPT Network of Excellence set up a pivotal workshop in 2004 called SASC (The State of the Art of Stream Ciphers) with the remit to foster a greater understanding of what makes a good stream cipher and assess whether the stream cipher had a viable future in the face of the widespread acceptance of block ciphers.

When the workshop took place in October 2004, this viability was openly questioned on a number of levels, with the exhaustive examination presented by Adi Shamir in "Stream Ciphers: Dead or Alive?" [118] arguing that the time of stream ciphers may have passed as potential size and speed advantages become irrelevant in the face of advancing technology. Despite this, industry contributors such as Steve Babbage of Vodafone UK indicated [119] there were indeed a number of commercial areas such as those of resource and power constrained communications and high speed software based communications where the attractions of the stream cipher were still compelling.

Subsequent discussions centred on the woeful lack of understanding surrounding

the design and analysis of stream ciphers when compared to that of block ciphers. As a proper assessment of the true viability of stream ciphers could not occur until such an understanding had been obtained the workshop concluded that a co-ordinated effort would be needed in the field. It was proposed that the greatest opportunity for achieving this lay in instigating a design project along the lines of the National Institute of Standards and Technology's block cipher competition [120] that had, a few years previously, led to the selection of the popular Advance Encryption Standard.

4.5.3 Call for Primitives

In November 2004 a call for stream cipher primitives [74] was made by the ECRYPT Network of Excellence as part of a competitive project to find a new stream cipher suitable for widespread adoption. The project surrounding this call would later be entitled, 'eStream.'

The call proposed that stream cipher entries should be aimed at two key areas given the names Profile 1 and Profile 2. Profile 1 would encompass stream cipher proposals which targeted the software usage extreme of maximal throughput while Profile 2 would focus on stream cipher proposals which aimed to minimise the resources needed for hardware implementations. A number of criteria accompanied these profiles and Profile 1 ciphers were required to accommodate 128-bit keys and a IV of either 64 or 128-bits, with Profile 2 proposals requiring the lesser criteria of 80-bit keys and an IV size of either 32 or 64-bits. Alongside this, three security criteria were detailed [74]:

- "Any key-recovery attack (including time-memory-data tradeoff attacks) should be at least as difficult as exhaustive search."
- "Distinguishing attacks are likely to be of interest ... however the relative importance of high complexity distinguishing attacks may become an issue for wider discussion."
- "Clarity of design is likely to be an important consideration."

Further subcategories, Profile 1A and Profile 2A, were specified for cipher proposals that included an authentication method. The message authentication code

length was set at 32, 64, 96 or 128-bits for the software focused Profile 1A designs and 32 or 64-bits for those designs aimed at the hardware Profile 2A.

The aim of the initiative was not just to be able to recommend primitives suitable for use in commercial systems but also to "derive good stream ciphers" [74]. With the failures of the past it was seen as critical to increase the understanding around algorithm design within the field and as such the first phase of the project was to "concentrate on accumulating information" [74], using this to maximise the strength of stream cipher proposals. In light of this it was proposed that changes to submitted designs would be readily permitted after the first phase ended in March 2006.

4.5.4 Comparison to the AES

Unlike stream ciphers, there is a good level of security confidence in block ciphers, with their structure well understood and algorithms tested extensively over a period of many years. As hinted at by the original call for primitives, to achieve similar levels of acceptance stream ciphers must realistically provide at least one significant performance benefit over the de-facto standard in block cipher symmetric encryption, the Advanced Encryption Standard (AES) [42]. Hardware implementations of AES-128 at the start of eStream would use as low as 5.4k NAND gate equivalents (GE) [43], a Profile 2 stream cipher which could achieve a smaller footprint or could significantly exceed the 311Mbps throughput for a similar area would certainly prove attractive to a wide variety of applications. For software, resource usage was seen as less important with it essential that Profile 1 stream cipher proposals exceed just the AES's fully optimised throughput which at around the time of the standard's agreement stood at 16 cycles/byte on a Intel Pentium 4 processor [121]. The remaining area where a stream cipher may find favour is the inclusion of an integrated authentication facility — this is a feature absent from the AES, making Profile 1A and 2A algorithms with even comparable speeds and resource usage of potential interest.

4.6 eStream Candidates

The 2004 call for stream cipher primitives would eventually lead to 34 proposals being submitted to ECRYPT ahead of the April 2005 deadline, with 25 of these aimed wholly or in part at the hardware focused Profile 2 criteria. Of these a further 5 would include some form of authentication functionality allowing them to meet

the Profile 2A criteria.

Due to the sheer number of candidates it was considered prudent to focus research towards a small number of candidates with the potential performance levels to make them suitable for implementation on a secure display type device. In selecting these ciphers, aspects of design complexity, documentation quality would be of importance, but the main criteria was that a proposed cipher should provide a significant advantage over incumbent block ciphers such as the AES. Although eStream would allow ciphers to be tweaked in later phases of the project, it was felt that with the security standing of self-synchronising stream ciphers at an even more embryonic stage than synchronous stream ciphers, only the 23 non-self-synchronous design proposals should be considered.

The first of five stream cipher proposals selected for further analysis, Grain [122], has the potential for a throughput per area performance many times in excess of AES and is specified flexibly to allow very low resource implementations. Similarly the speed, flexibility and low resource usage of Trivium's [123] cross-linked feedback structure appears impressive. The third stream cipher, MICKEY [124] has a more complex structure and a lack of accelerated feedback leads to less flexible, lower throughput implementations. Nonetheless, the use of variable clocking and the potential of higher security of the co-defined cipher, MICKEY-128, provides a suitable contrast from a security perspective to the other selected hardware primitives.

The final two cipher proposals considered both include authentication functionality in their specifications while still potentially operating a greater hardware efficiency than AES. The footprint in hardware of these ciphers is likely to be higher than than the three ultra low resource ciphers but the potential resource sharing of integrated authentication in a final device for resource sharing was believed worthy of consideration. The hardware focused SFINKS [125] provides this authentication potential with the minimum of resources while the large block cipher structure of Phelix [126] potentially provides greater security, good software performance and another contrast in stream cipher design principles.

4.6.1 Grain

Designed aggressively for high-speed low-area implementation in hardware in its original submission, Grain [122] was specified to use an 80-bit key and 64-bit IV, meeting the criteria for a Profile 2 stream cipher. The cipher's heritage partially lies

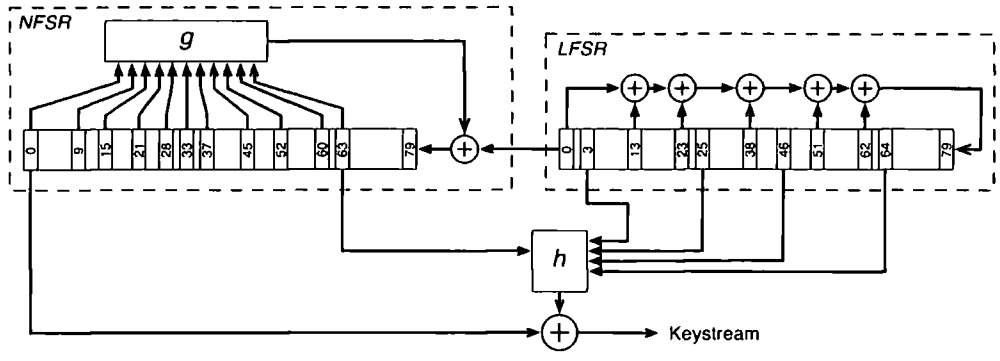


Figure 4.10: Schematic description of the Grain keystream generator as original specified in eStream: latterly known as Grain v0

in that of SNOW 2.0, via its second author Thomas Johansson — however, the move to a bit orientated structure and the use of less hardware intensive operations leads to a much more hardware efficient design.

The cipher, as shown in Figure 4.10, is primarily based around two shift registers, a conventional 80-bit LFSR which helps provide a long minimum period and an 80-bit NFSR (non-linear feedback shift register) which through the 11-variable filtering function g provides the cipher's non-linear complexity. These registers form a 160-bit state which is processed by the 5 variable filter function h and masked with a NFSR bit to generate the keystream. The functions g and h can be found in [122] and simply consist of a combination of AND and XOR operations which in themselves pose little restriction to hardware size or operating speed.

For Grain's initialisation the registers, NFSR $(b_{79}, ..., b_0)$ and LFSR $(s_{79}, ..., s_0)$, are loaded directly with the key and IV material as follows:

$$\text{NFSR } (b_{79}, ..., b_0) \leftarrow (K_{79}, ..., K_0)$$
$$\text{LFSR } (s_{79}, ..., s_0) \leftarrow (1, ..., 1, IV_{63}, ..., IV_0)$$

To fully mix this data the cipher is cycled 160 times with the keystream output disabled and the output material instead fed back into the two shift registers, as shown in Figure 4.11. Following this the cipher returns to the conventional configuration of Figure 4.10 and 1-bit of keystream material is generated every cycle.

An interesting property of Grain is that the choice of tap points for the feedback and filter functions allows the use of additional feedback loops to accelerate the keystream generation process. Figure 4.12 illustrates the use of additional hardware logic to double output generation to 2-bits per cycle, which not only speeds keystream production but halves the required initialisation mixing period. The concept can be

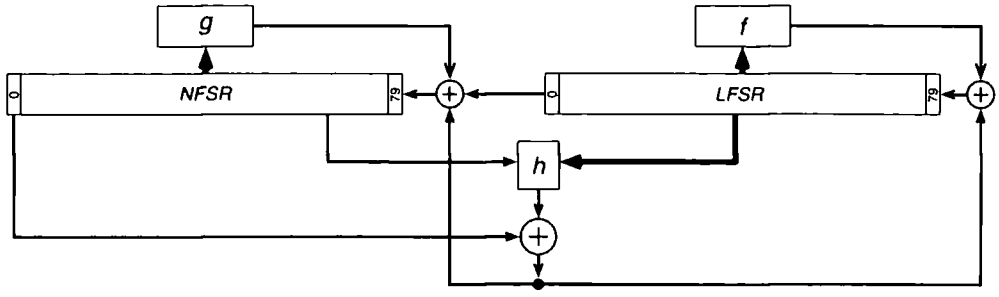


Figure 4.11: Schematic diagram of Grain v0 at initialisation

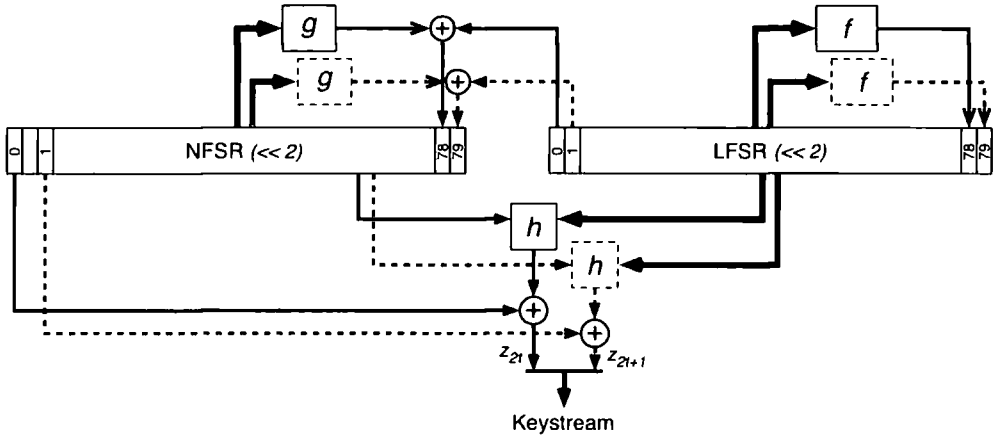


Figure 4.12: Schematic diagram showing Grain v0 with the throughput doubled, this acceleration of Grain v0 can further extended through use of additional feedback loops

expanded further with the highest throughput configuration specified achieving a sixteen times increase in speed with 2-bytes of output per cycle. In this configuration, once key and IV material are loaded, Grain needs only 10 cycles to initialise, giving the cipher a potentially impressive key agility, although the optimal usage point may be less clear-cut with routing congestion from the additional feedback polynomials potentially adversely affecting achievable clock speeds. Software implementations should benefit from the possibilities of using broad 16-bit manipulations, and taking a wider context, the accelerated feedback options of Grain provide the cipher with a strong level of flexibility in trading data throughput against area usage, a factor which should prove useful in its application to resource restricted hardware environments.

4.6.2 Trivium

Alongside Grain, the other high performance and ultra-low resource stream cipher in the original eStream submissions was Trivium [123]. The cipher is specified to meet

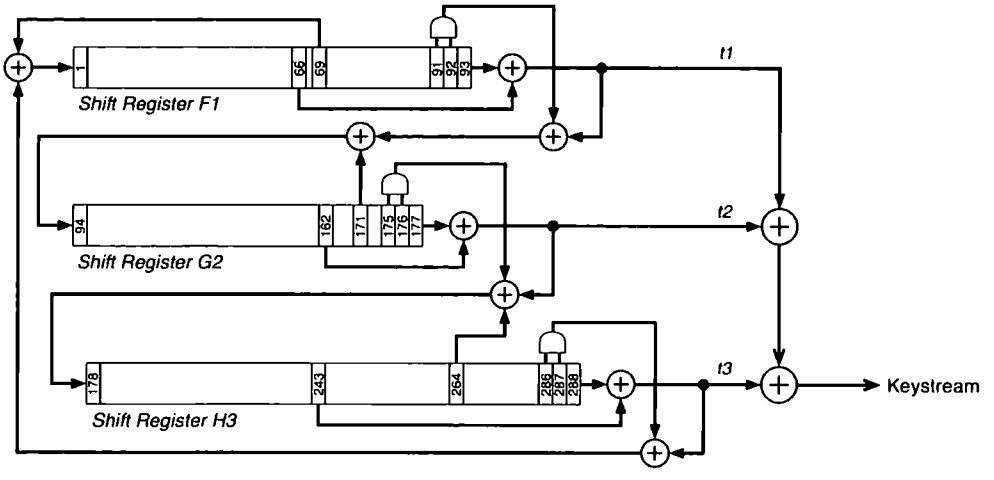


Figure 4.13: Schematic description of the Trivium keystream generator

the criteria for a Profile 2 stream cipher using an 80-bit key and 80-bit IV for its generation of keystream material. As shown in Figure 4.13 Trivium is fairly novel in its design, being based around three registers of 93-bits , 84-bits and 111-bits with five element cross linked feedback functions providing both the long period and the computational complexity against analysis. As the authors Christophe De Cannière and Bart Preneel indicate in their original submission, Trivium "was designed as an exercise in exploring how far a stream cipher can be simplified without sacrificing its security" and in its use of just a handful of XOR and AND operations the design would certainly appear to aggressively test the limits at how simple a stream cipher can be whilst remaining secure. The background design philosophy of Trivium is further described in [127].

Although its update functions are minimal, and well suited to compact hardware implementation, its state size at 288-bits is 80% larger than that of Grain, likely leaving it at an overall disadvantage when a minimum area cipher is desired. However the feedback can, like Grain, be accelerated through parallelisation with the minimalist nature of the update function providing greater efficiency to the process. Furthermore, the large state allows Trivium to be specified for a 64-times increase in output giving the cipher improved software efficiency and a very high throughput potential of 8-bytes a cycle. This high level of flexibility provided by Trivium in trading area usage against throughput again improves its suitability to restricted hardware environments. As with Grain the usual caveats on routing complexity would apply with regard to maximum clock speeds.

The initialisation of Trivium is slightly more complex than that of Grain and in its non-accelerated form after key loading it requires 1152 cycles of mixing before the

output is enabled and keystream bits generated. In the 64-bit parallelised version this figure falls to just 18 cycles giving the cipher a still potentially excellent key agility. For loading the registers $F1$, $G2$, $H3$ that form the state 's' are fed directly the key and IV material as follows:

$$(s_1, ..., s_{93}) \leftarrow (K_0, ..., K_{79}, 0, ..., 0)$$
$$(s_{94}, ..., s_{177}) \leftarrow (IV_0, ..., IV_{79}, 0, ..., 0)$$
$$(s_{178}, ..., s_{288}) \leftarrow (0, ..., 0, 1, 1, 1)$$

4.6.3 MICKEY

MICKEY [124] is aimed at resource-constrained hardware environments but places its emphasis on minimising area usage and proving security with a lower priority given to data throughput performance. The cipher is specified to meet the Profile 2 criteria using an 80-bit key and 80-bit IV in its generation of keystream material.

As shown in Figure 4.14 MICKEY's keystream generator is based around two 80 stage, 1-bit wide registers, a variable feedback LFSR 'R' and a variable lookup NFSR 'S', with MICKEY's 1-bit per cycle keystream output generated by combining a single bit from each through an XOR operation. The two control bits that regulate the cipher's variability are also generated in this manner and these help link the structure so that the 160-bits of state material provides the long period necessary and good statistical qualities for high quality keystream production. The 'control bit R' effectively jumps the LFSR forward $(2^{40} - 23)$ times and compliments the non-linearity of MICKEY's state update function provided by 'S'.

Unlike many other stream ciphers, the shift registers of MICKEY are of a Galois style construction, with XOR operators interspersing the individual registers — the detailed designs of 'R' and 'S' can be found in Figures 4.15 and 4.16 respectively. The non-linear 'S' register is especially complex, requiring a large number of NOT and AND operations in its feedback structure. Although this will provide non-linear complexity, it will also increase its area usage, especially for ASIC designs. Nevertheless, the fact that this feedback is from nearby registers should minimise routing complexities, making FPGA implementations efficient and the fixed values defined in [124] for the lookup table 'FB' should, when implemented, lead to almost as much logic redundancy as additional AND gates. For the register 'R' 80 AND gates would be required to implement its variable feedback.

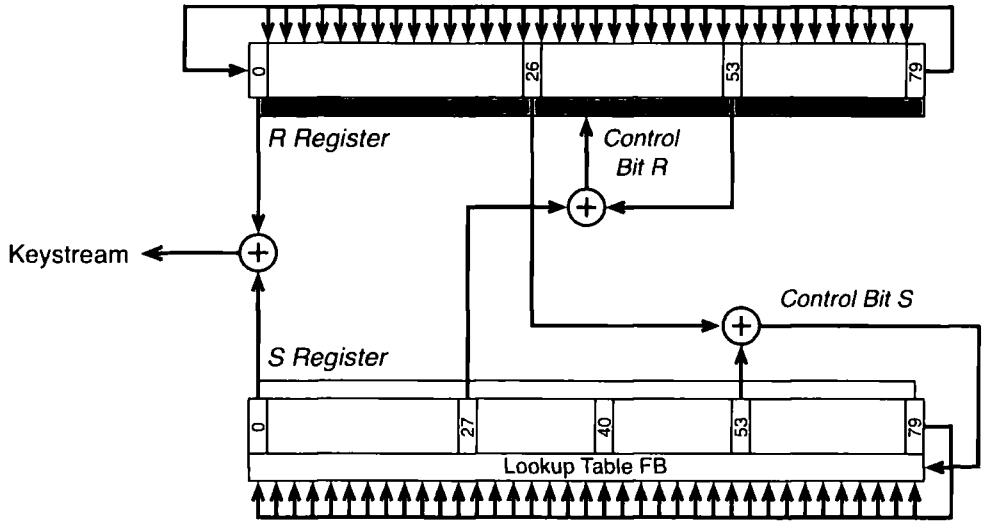


Figure 4.14: Schematic description of the MICKEY keystream generator

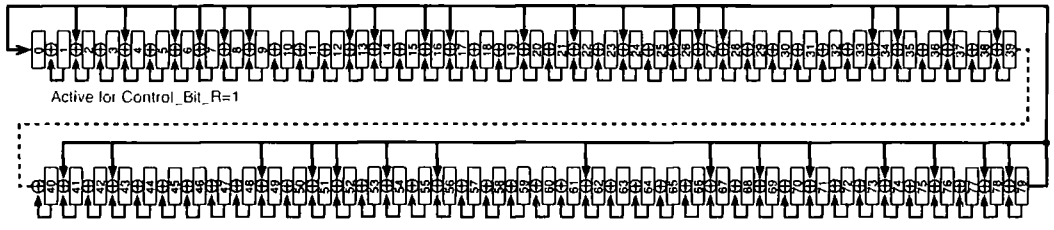


Figure 4.15: R register layout for the stream cipher MICKEY

Although the simple operations in MICKEY should not significantly affect achievable hardware clock speeds, the additional logic not only increases resource usage but also prevents the efficient feedback acceleration achievable in Grain and Trivium. This forces the cipher to be operated in bit-wise manner, restricting throughput performance and leading to inefficient implementations in software.

In operation MICKEY is initialised by setting the registers to zero and then, with the cipher configured as in Figure 4.17, the IV is fed in one bit per cycle to the input (IV_0 being loaded first.) Following this, the 80-bit key is similarly clocked in. Then with the input set to zero the cipher is clocked for a further 80 cycles in the same initialisation configuration before the keystream is generated as in Figure 4.14.

Although Profile 2 proposals were specified to accommodate 80-bit key lengths the original MICKEY submission would include a second cipher proposal accommodating larger 128-bit keys — this cipher MICKEY-128 is shown in Figure 4.18. Apart from larger 128-bit register sizes and increased initialisation mixing period which aim to maintain a target security level greater than exhaustive search, MICKEY-128 operates in largely identical way to MICKEY. The extra resources needed will,

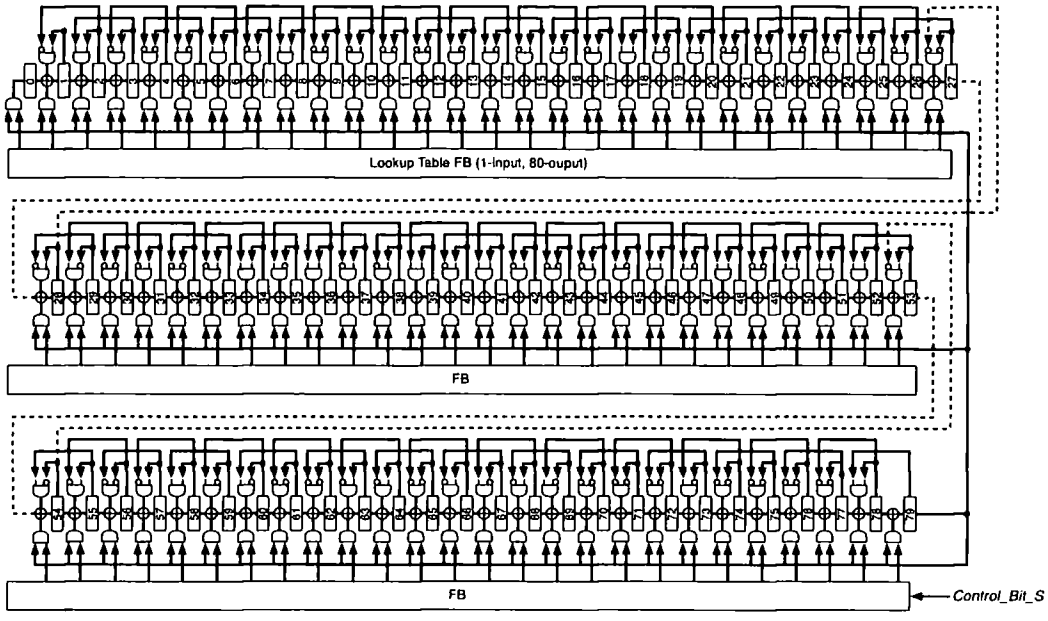


Figure 4.16: S register layout for the stream cipher MICKEY

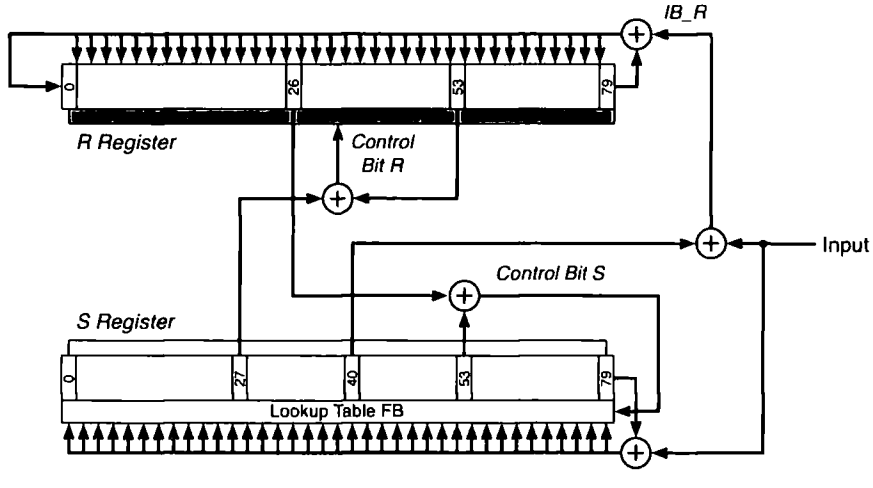


Figure 4.17: Schematic description of MICKEY at initialisation

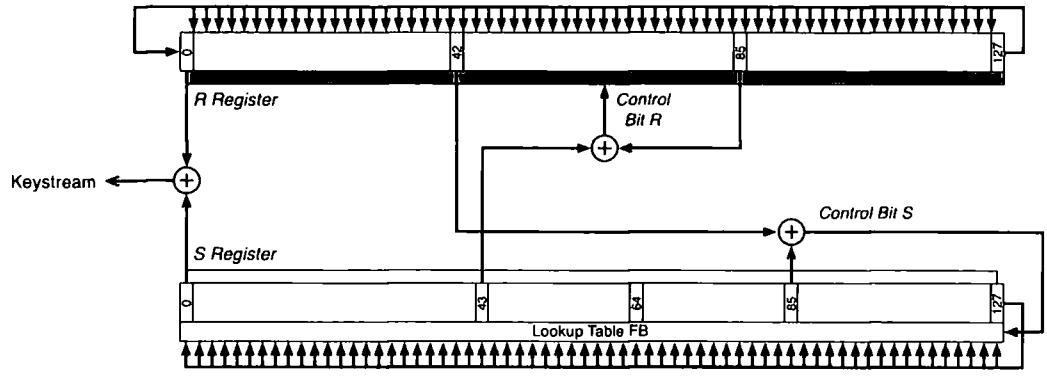


Figure 4.18: Schematic description of the 128-bit key version of the MICKEY stream cipher, MICKEY-128

however, adversely affect the cipher's software performance and hardware area usage making it attractive only if it provides extra security over that of an 80-bit key.

4.6.4 SFINKS

Still aimed at resource constrained environment SFINKS [125] has a relatively aggressive design approach and is specified with an 80-bit key and 80-bit IV to meet the basic Profile 2 criteria. In addition to this, the proposal describes a simple built-in authentication mechanism capable of generating a 64-bit MAC which allowed SFINKS to meet the further requirements for categorisation as a Profile 2A cipher.

The stream cipher without authentication is shown in Figure 4.19 and follows a structure based around the non-linear filtering of state material. This material is provided by a relatively large 256-bit LFSR which is designed to give the cipher a long underlying period. From this 16-bits are taken and fed through a substitution-box (S-box) based on inversion in the field $GF(2^{16})$ modulo $X^{16} + X^5 + X^3 + X^2 + 1$. In normal operation only the least significant bit from this operation is used in the keystream generation process with others simply discarded. A single register bit is used to mask this bit and form the keystream output.

The main complexity in SFINKS comes from the design of the S-box inversion. If it were to be implemented in its raw form it would utilise 64Kb of memory, an impractical figure for resource efficient hardware. However, as described in the original paper [125] the chosen function can, like the AES S-box [128], be implemented in hardware logic using field decomposition to allow a much greater level of resource efficiency. Good [129] suggests that further efficiency savings can be made through

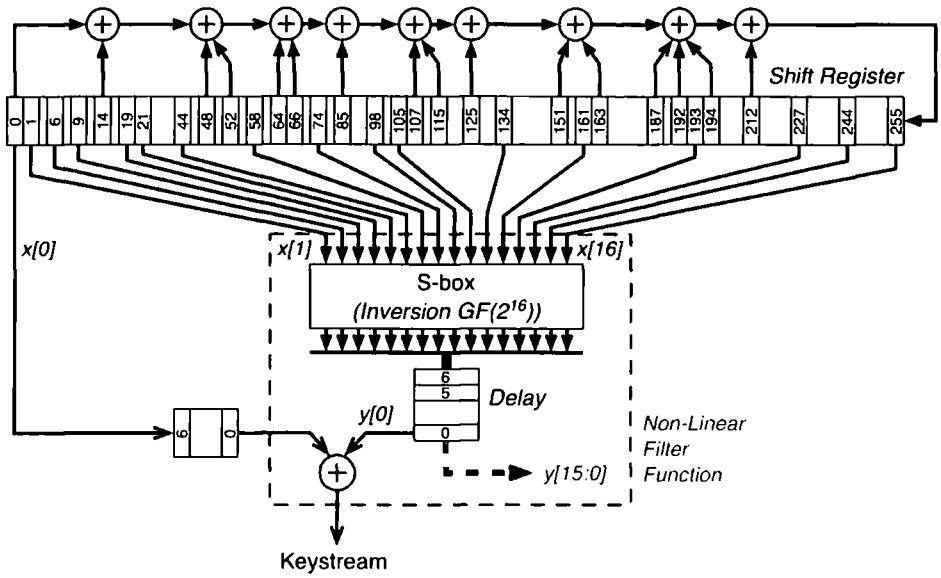


Figure 4.19: Schematic description of the SFINKS keystream generator

resource sharing, but the inversion process will still prove costly, requiring among other operations an 8-bit multiplier, a table for inversion in the field $GF(2^2)$ and the implementation of transformation matrices to move between fields $GF(2^{16})$ and $GF((((2^2)^2)^2)^2)^2)$.

In operation the initialisation of SFINKS requires the output of the S-box inversion to be fed back into LFSR as shown in Figure 4.20 with the LFSR being initial loaded with the Key and IV material as follows:

$$(s_{255}, ..., s_0) \leftarrow (IV_{79}, ..., IV_0, K_{79}, ..., K_0, 1, 0, ..., 0)$$

Due to the authors' pipelined implementation of the inversion function, the cipher specifies a 7-cycle delay on feeding back the output. This proves quite expensive in hardware terms, necessitating 112-bits of registers: these are reset to zero on initial loading. The whole initialisation process takes 128 cycles and, unlike Trivium and Grain, this cannot efficiently be reduced by accelerated feedback. Afterwards keystream is produced at a rate of a bit per cycle.

For generating the MAC, SFINKS takes one bit of the S-box output $y[0]$ and uses it as an input to a 64-bit shift register. The contents of this register are combined with the plainstream through an AND operation and then further combined with the previous contents of the MAC register through an XOR operation to produce a short 64-bit MAC. Whether this MAC size would be sufficient to provide meaningful

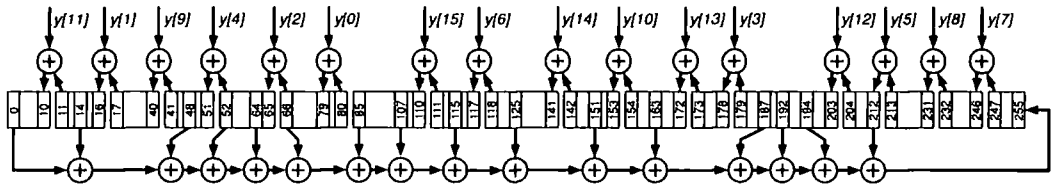


Figure 4.20: SFINKS' LFSR during initialisation

security is open to question — however the authors do suggest that it is easy to "adapt these parameters in a straightforward way" [125].

4.6.5 Phelix

Like SFINKS Phelix [126] is different to the other stream ciphers under consideration in the way it combines the properties of encryption with that of message authentication. This has the potential to eliminate the need for additional electronics to provide authentication functionality, and as well as encouraging authentication use, may lead to a reduction in overall hardware logic. Phelix is based on the previously broken stream cipher Helix [130] and shares much of its conservative design approach.

The cipher itself uses a 128-bit IV and a 256-bit key size and claims to provide 128-bit security. The paper further describes a key mixing process for expanding small keys allowing Phelix to meet both the software Profile 1 and hardware Profile 2 criteria, while the fully integrated 128-bit MAC function means the cipher proposal also meets the criteria for a Profile 1A and Profile 2A stream cipher.

Unlike many other stream ciphers, Phelix is based around a rotating block function which is fed sub-keys from a key expander in a structure more akin to that of a block cipher. As shown in Figure 4.21 each round consists of a 160-bit state register and two identical half blocks which operate as an update function. A further 128-bits of state material is provided in a 32-bit wide delay block, the output of which is added to 32-bits processed from a half-block to produce a 32-bit block of keystream.

The half block which forms the core processing element of the cipher consists of a number of add, XOR and cyclic rotations and is shown in Figure 4.22. While in hardware cyclic rotations are effectively free and 32-bit XOR operations not overly burdensome, the half block's six 32-bit additions will be very costly, creating difficulties for low area, high throughput implementations. Moreover, as shown in Figure 4.21 the key expander liberally uses both subtractions and additions meaning

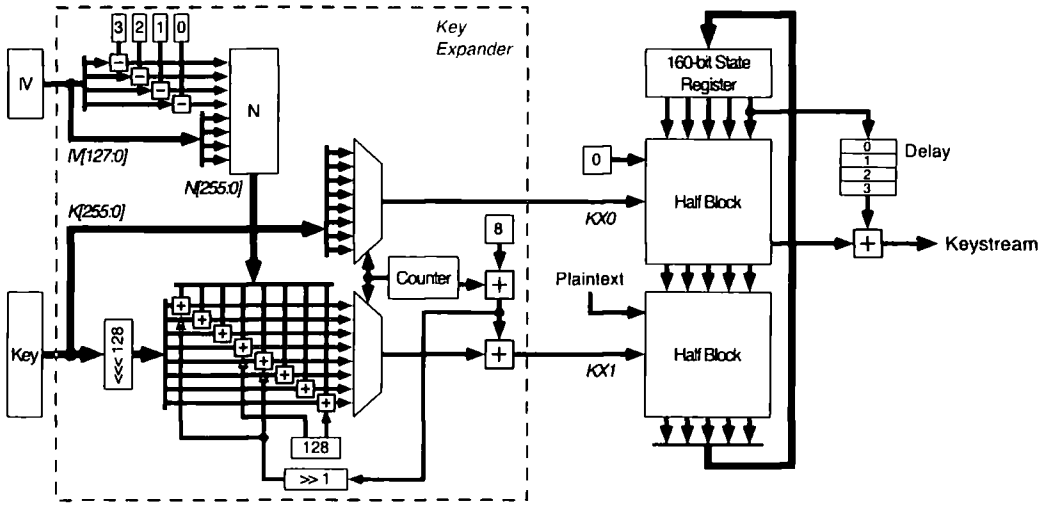


Figure 4.21: Schematic description of the Phelix keystream generator

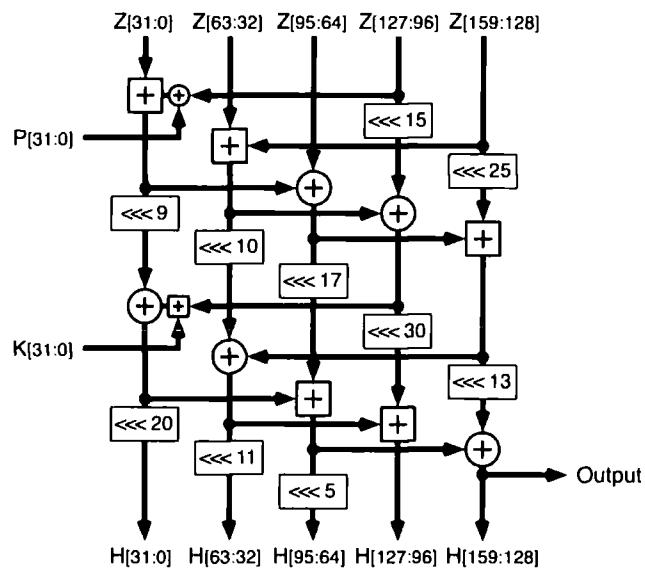


Figure 4.22: Schematic description of the encryption half block in Phelix

that unless a processor based structure is chosen for hardware implementation area usage may end up prohibitively high.

In operation the initialisation procedure involves resetting the delay registers to zero then generating eight blocks with no plainstream inputted and no keystream outputted. The state register is initially set as follows:

$$(Z_{159}, \ldots, Z_0) \leftarrow (K_{255}, \ldots, K_{224}, \{K_{223} \oplus IV_{127}\}, \ldots, \{K_{96} \oplus IV_0\})$$

Following plaintext encryption a MAC from the cipher can be generated by the

XOR of $Z_{31:0}$ with 0x912d94f1 and then cycling the cipher a further eight times with the plainstream input the length in bytes of the plainstream modulo-4. The keystream generated by a subsequent four cycles forms the 128-bit MAC. A method for generating a MAC for unencrypted data, such as header information, is also specified in the Phelix proposal [126].

As the initialisation process requires a 256-bit key size, a pre-mixing process must be used if a shorter length of input key is chosen. This involves first appending zeros to reach a size of 256-bits and then loading the lower 128-bits into state registers $Z_{127:0}$. The remaining 32 state bits are filled with a variable based on the key length in bytes plus 64 and the block function is then processed with the plaintext and key input disabled. By combining (through an XOR operation) the lower 128-bits and the upper 128-bits of the appended key, new key bits $K_{255:224}$ are generated. Following this, further key bits are produced by placing the newly generated key bits in registers $Z_{127:0}$ and using the previous contents for the post-output XOR operation. Generating all 256-bits takes eight block calculations but as key loading is a generally infrequent operation this should not greatly affect the ciphers performance. The provision for key mixing will however add to hardware implementation complexity.

Although there is no official acceleration method to increase the throughput per area performance of Phelix, there are a number of possibilities available to the implementation of its encryption block. A few of these are outlined by Good in [129] where in addition to the full round structure of Figure 4.21 architectures based on a half round structure and 32-bit datapath architecture are described, offering options for reduced area usage. The half round architecture splits the encryption into a 2-cycle process instead of duplicating the implementation of the half block which provides a reduction in resource usage with an accompanying drop in throughput. However the complexity of the key expander means that a one-to-one trade-off is unlikely to be realised and the percentage area gain would be small. As mentioned previously, moving to a processor-based design style offers more promise in terms of area usage reduction and a datapath architecture would allow resource sharing of the SHIFT, ADD and XOR operations. Nonetheless, the complexity of the key expander operation, the number of iterations needed for encryption block implementation and the required delay prior to keystream output would require the architecture to implement a large state machine controller and comparatively large memory while having the disadvantage of a very low throughput performance. It is all but certain that the flexibility benefits of the architecture variations described are much less significant than those accrued in Trivium and Grain.

4.7 Notes on Subsequent eStream Progression

Following the initial submission of proposals a period of community based analysis was proposed by the eStream Project. This culminated a year later in May 2006 with a further workshop [131] where the results of this cryptanalysis were presented and reviewed.

In August 2006 Phase 2 of the project began with most of the submitted ciphers being advanced through. However, the eStream committee decided that three of the original Profile 2 submissions should instead be archived for performance or security failings. Most significantly this would see the archiving of SFINKS which had by this stage been shown to be vulnerable to algebraic attack [132] — an area indeed which the authors had noted might prove to be a vulnerability: "we are aware of the fact that the security against the (fast) algebraic attack is really on the edge" [125]. Despite the same paper making some suggestions on countermeasures no alterations to the algorithm were officially made before the start of Phase 2, necessitating its withdrawal from the eStream process.

Alongside the decisions on archiving, in this second phase eStream would introduce the concept of giving focus status to algorithms to encourage analysis in areas with the most promise. Of the four selected primitives that made it through the archiving process all would became focus ciphers for this second phase. In accordance with the original principles of the project a number of ciphers were allowed to make alterations to improve performance and security, both Grain and Mickey would take advantage of this. The progression of these ciphers and that of Trivium and Phelix are outlined below.

4.7.1 Phelix

No modifications were made to Phelix for Phase 2. However, the advantage of packaging encryption and authentication functions in one algorithm proved to be the primitive's undoing, with the IV reuse-based attack published by Wu and Preneel [133] considered a real-world threat due to the possibility for MAC forgeries. The algorithm was archived in March 2007 and did not make it into the third phase of the eStream project.

4.7.2 Trivium

Trivium would also remain unaltered for Phase 2, but findings on its security were more positive, with the keystream generator surviving cryptanalysis despite significant attention from the research community. The best proposed algorithmic attack would be by Maximov [134], with a claimed complexity of $2^{83.5}$ — a figure very close to the intended 2^{80} security level. Nonetheless it is unclear whether this attack can be further improved and the high $2^{61.5}$ keystream requirement may adversely affect the attack's practicality. A sister toy cipher Bivium[4] described in [127] would however seem to be very weak with Maximov proposing a 2^{14} complexity state recovery attack and McDonald proposing a practical attack using a satisfiability solver [135]. Satisfiability solvers would appear however not to threaten Trivium itself [136].

Although Trivium's security margin is small, it is present, and so the cipher progressed into the final phase of the eStream project (see Section 4.7.9).

4.7.3 MICKEY 2.0 and MICKEY-128 2.0

MICKEY was revised in preparation for Phase 2 after Hong [137] noted various possible areas presenting vulnerabilities, including TMTO, state entropy loss, keystream convergence and weak keys. These would be addressed in MICKEY 2.0 [138] and MICKEY-128 2.0 [138] by increasing the size of registers in the cipher from 80-bits to 100-bits. The resulting revised designs can be seen in Figures 4.23 and 4.24. The increased area usage, although significant, may not be detrimental to the appeal of the cipher family as its main attraction remains its relatively conservative design approach.

Evidence would suggest that use of irregular clocking is a risky strategy for symmetric encryption as far as stream cipher security is concerned, with previous ciphers based on the strategy, A5/1, LILI-128 [139], and the Shrinking Generator [140], all having suffered major attacks [99, 141, 142]. However the complexity of MICKEY's clocking scheme would appear to be sufficiently secure and successful algorithmic attacks against MICKEY 2.0 and MICKEY-128 2.0 were not realised during Phase 2. In view of this information the eStream committee progressed both these revised versions.

[4] Developed to aid cryptanalysis Bivium is a reduced version of Trivium that removes register $H3$ and uses the combination of output tap points $t1$ and $t2$ (see Figure 4.13) to generate the output keystream.

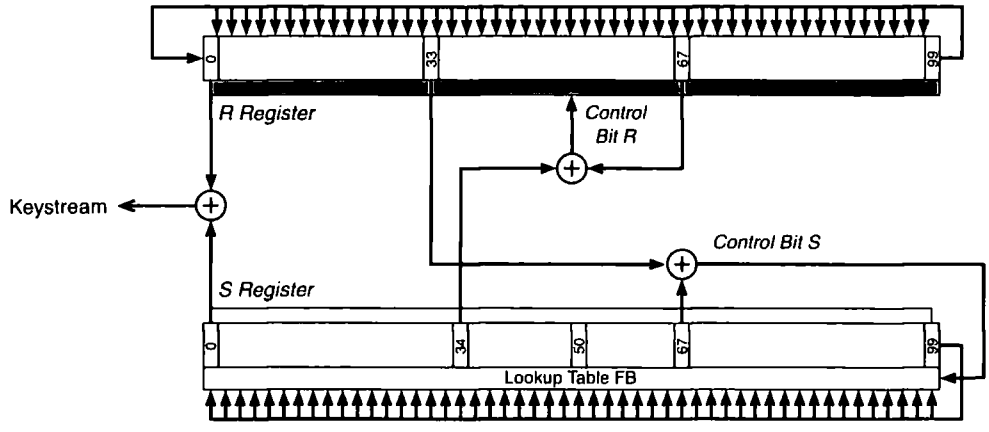


Figure 4.23: The revised MICKEY keystream generator, MICKEY 2.0

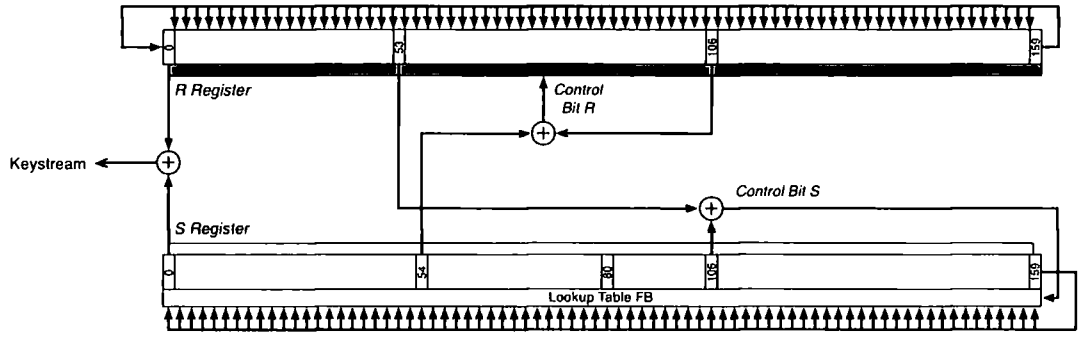


Figure 4.24: The revised MICKEY-128 keystream generator, MICKEY-128 2.0

4.7.4 Grain v1 and Grain-128

The aggressive nature of the original Grain design meant that its security margin was minimal and flaws were found in original keystream generator shortly after conception with first a distinguishing attack by Khazaei [143] of 2^{61} complexity, 2^{40} memory and 2^{61} data, followed by a further much more practical attack by Berbain [144] requiring 2^{43} operations and 2^{42} memory to perform key recovery and leaving Grain effectively broken. In light of the attacks a revised version of the cipher was proposed in [145] which modified the polynomials used in the feedback and output functions. This new primitive was assigned the title Grain v1 and is shown in Figure 4.25.

Although using more register terms, the modified g function actually uses one fewer operation in its generation, leaving only the slightly increased complexity of the output mask potentially affecting performance. Nevertheless, Grain v1 should perform essentially the same in hardware as its predecessor. Moreover, the modifications made would appear to significantly improve security, as by the end of Phase 2 no notable attacks against the primitive had been found despite its status as a focus

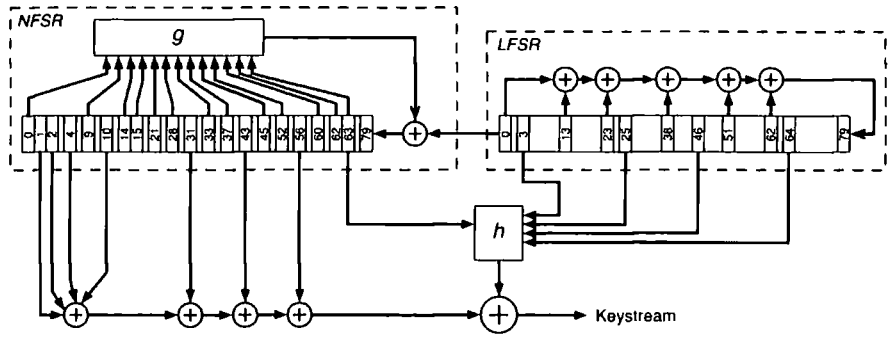


Figure 4.25: Schematic description of the updated Grain keystream generator, Grain v1

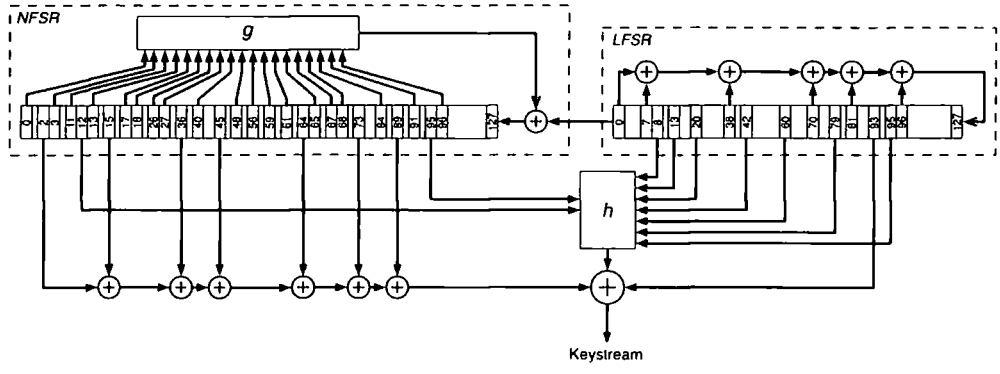


Figure 4.26: Schematic description of the Grain-128 keystream generator

cipher.

As was allowed by the competition, a further version of Grain called Grain-128 [146] was proposed in Phase 2 of eStream — this used a larger 128-bit key length and was designed to provide a greater level of security. The underlying structure of the stream cipher is shown in Figure 4.26 and remains the same as its smaller cousin, although both register and function sizes are increased while slight modifications were made by the authors to improve software performance and allow a 32-times increase in speed.

With this larger structure, the cipher's initialisation period was increased to 256 cycles to ensure full mixing of the key and IV. The initial loading of the key and IV material is as follows:

$$(b_{127}, ..., b_0) \leftarrow (K_{127}, ..., K_0)$$
$$(s_{127}, ..., s_0) \leftarrow (1, ..., 1, IV_{95}, ..., IV_0)$$

The simple structure of Grain-128 should mean the cipher retains excellent

Table 4.1: Performance of Stream Cipher Keystream Generation Schemes on a AMD Athlon 64 X2 System (figures quoted from [147])

Keystream Generator	Key Size	Throughput *cycles/byte*	IV Setup *cycles*
TRIVIUM	80	4.14	583.80
SNOW 2.0	128	4.83	528.04
Phelix	128	5.66	745.04
AES-128$_{CTR}$	128	13.39	15.58
Grain-128	128	14.02	467.23
Grain v1	80	42.13	865.52
MICKEY 2.0	80	627.28	21877.69
MICKEY-128 2.0	128	730.82	40783.00

performance in hardware despite its increased size. As of the end of Phase 2 no vulnerabilities had been found for Grain-128, allowing it join Grain v1 in the next phase of the eStream project.

4.7.5 Software Performance

The software performance of the eStream cipher proposals were comprehensively assessed in the eStream testing framework [112] and selected results for the four Phase 2 candidates under study, from the AMD Athlon 64 X2 system tests [147], are presented in Table 4.1. From these it can be seen that only Trivium and Phelix outperform AES in counter mode, although the throughput performance of Grain v1 and Grain-128 are close enough that server-side concerns should not unduly influence hardware decisions. The initialisation and throughput performance of MICKEY 2.0 and MICKEY-128 2.0, at over an order of magnitude slower may be more problematic to application use. One other figure of note is the very low IV set-up time needed for AES in counter mode, this advantage over stream ciphers, typical to such modes of operation, is beneficial to the processing of small blocks of data but is largely negated once packet sizes reach hundreds of bytes.

Although the absolute results differ for Intel based software test systems [113, 148] the performance ordering is similar, with only Trivium and SNOW 2.0 trading places.

4.7.6 Hardware Performance

However impressive software performance is, it is hardware decryption performance that is of key importance for implementing a secure display system, and also the standard against which the the selected Profile 2 ciphers are assessed. A number of evaluation results have been presented over the course of eStream [129, 149, 150, 151, 152]; the collated results from these assessments of the candidates under study are summarised in the Tables 4.2 and 4.3. Further results for Grain and Trivium implemented in the course of this research can be found in Chapter 7.

Both the results for ASIC (application specific integrated circuit) and FPGA (field programmable gate array) implementations show a high degree of consistency in their ordering of the Phase 2 ciphers. Where the minimisation of area usage is of primary importance Grain v1 would appear to be the best option, closely followed by Grain-128 and Trivium, while for maximising throughput efficiency this order is reversed with Trivium appearing to be the most capable. Tables 4.3(b) and 4.2(a) indicate that although these three cipher designs are slightly larger than the insecure A5/1 stream cipher they are capable of exceeding its throughput performance. Moreover the figures show Grain and Trivium are each capable of outperforming the AES-128 in all three aspects of area, power and throughput efficiency, sometimes by a very high degree: the throughput efficiency of Trivium on FPGAs is over two orders of magnitude higher. The attractions of such performance advantages would certainly seem to be sufficient to encourage the introduction of stream ciphers back into the hardware environment.

The results for the other Phase 2 candidates under consideration, MICKEY and Phelix, have less attractions. Phelix's minimum area usage is particularly high and significantly exceeds that of the AES, while MICKEY's throughput efficiency lags behind that of the AES in ASIC environments. Only in FPGA throughput performance would either of the designs be seen to have a particular advantage over the incumbent AES block cipher standard.

[a] Figures derived from quoted physical area. TSMC 90nm process density $420kgates/mm^2$ [155].
[b] Including CBC support in these implementations would allow a fairer comparison to the usage model of stream ciphers and is likely to add a further 1.6k gates [156].

Table 4.2: Hardware Performance of Stream Cipher Keystream Generation Schemes for Application Specific Integrated Circuits

(a) ASIC designs optimised for minimum area

Design	Bits/ Cycle	CMOS Process	Area (gate equiv.)	f_{max} (MHz)	Setup $(cycles)$
Grain v1 [150]	1	0.13 μm	1.294k	725	321
Grain-128 [150]	1	0.13 μm	1.857k	926	513
Trivium [150]	1	0.13 μm	2.599k	358	1333
Phelix [150]	16	0.13 μm	13.159k	88	51
Grain v1 [153]	1	90 nm	(2.063k)[a]	565	304
Trivium [153]	1	90 nm	(3.120k)[a]	840	1312
MICKEY-128 [153]	1	90 nm	(6.817k)[a]	457	416
Phelix [153]	16	90 nm	(22.357k)[a]	316	44
A5/1 [153]	1	90 nm	(0.834k)[a]	685	186
AES-128[b] [154]	0.124	0.35 μm	3.400k	80	1032

(b) ASIC designs optimised for low-power

Design	Bits/ Cycle	CMOS Process	Area (gate equiv.)	Setup $(cycles)$	I_{mean} @100kHz,1.5V
Trivium [152]	0.727	0.35 μm	3.090k	1603	0.68μA
Grain v1 [152]	1.231	0.35 μm	3.360k	130	0.80μA
AES-128[b] [154]	0.124	0.35 μm	3.400k	1032	3.0μA

(c) ASIC designs optimised for maximum throughput to area ratio

Design	Bits/ Cycle	CMOS Process	Area (gate equiv.)	Setup $(cycles)$	Throughput $(Mb/s.GE)$
Trivium [153]	64	90 nm	(5.645k)[a]	21	(9.071)[a]
Grain v1 [153]	16	90 nm	(4.302k)[a]	19	(1.788)[a]
Phelix [153]	16	90 nm	(22.357k)[a]	44	(0.226)[a]
MICKEY-128 [153]	1	90 nm	(6.817k)[a]	416	(0.067)[a]
Trivium [150]	64	0.13 μm	4.921k	24	4.531
Grain-128 [150]	32	0.13 μm	4.617k	17	3.140
Grain v1 [150]	16	0.13 μm	3.239k	21	3.048
Phelix [150]	32	0.13 μm	15.032k	34	0.134
Trivium [149]	64	0.25 μm	6.382k	-	3.736
Grain v1 [149]	16	0.25 μm	5.454k	-	0.918
MICKEY [149]	1	0.25 μm	3.709k	-	0.079
SFINKS [149]	1	0.25 μm	6.953k	-	0.025
AES-128[b] [43]	11.6	110 nm	12.454k	11	0.136
AES-128[b] [43]	2.4	110 nm	5.398k	54	0.058
AES-128$_{OFB}$ [149]	3.12	0.25 μm	12.975k	-	0.047
A5/1 [153]	4	90 nm	(1.508k)[a]	186	(1.064)[a]
SNOW 2.0 [53]	32	0.18 μm	7.128k	~1000	0.234

Table 4.3: Hardware Performance of Stream Cipher Keystream Generation Schemes for Field Programmable Gate Array Designs

(a) FPGA designs optimised for minimum area

Design	Bits/ Cycle	Device Family	Area	f_{max} (MHz)	Setup $(cycles)$
Grain v0 [129]	1	EP1C	191 LEs	235	-
Grain v1 [157]	1	EP1C	219 LEs	242	322
Trivium [129]	1	EP1C	327 LEs	249	-
Trivium [157]	1	EP1C	393 LEs	295	1578
MICKEY-128 [157]	1	EP1C	537 LEs	220	603
SFINKS [129]	1	EP1C	556 LEs	207	-
Phelix [129]	16	EP1C	1455 LEs	82	-
Trivium [129]	1	XC2S	40 $slices$	102	-
Trivium [151]	1	XC2V	41 $slices$	207	-
Grain-128 [151]	1	XC2V	48 $slices$	181	-
Grain v0 [129]	1	XC2S	48 $slices$	105	-
Grain v1 [153]	1	XC3S	122 $slices$	193	304
MICKEY-128 [158]	1	XCVE	167 $slices$	170	-
MICKEY-128 2.0 [151]	1	XC2V	190 $slices$	200	-
Phelix (datapath) [129]	0.11	XC2S	264 $slices$	82	-
SFINKS [129]	1	XC2S	334 $slices$	118	-
Phelix [153]	16	XC3S	1197 $slices$	52	44
Phelix [151]	16	XC2V	1213 $slices$	63	-
A5/1 [153]	1	XC3S	57 $slices$	174	186
AES-128 [159]	0.03	XC2S	264 $eq.slices$	67	-

(b) FPGA designs optimised for maximum throughput to area ratio

Design	Bits/ Cycle	Device Family	Area	Setup $(cycles)$	Throughput $(Mb/s.LUT_{eqv.})$
Trivium [157]	16	EP1C	700 LEs	38	23.31
Grain v1 [157]	16	EP1C	508 LEs	19	6.77
Grain v0 [122]	16	EP1C	553 LEs	-	5.67
Phelix [129]	32	EP1C	1772 LEs	-	0.81
MICKEY-128 [157]	1	EP1C	537 LEs	603	0.41
Trivium [153]	64	XC3S	388 $slices$	21	15.67
Grain v1 [153]	16	XC3S	356 $slices$	19	3.49
Phelix [153]	32	XC3S	1402 $slices$	28	0.53
MICKEY-128 2.0 [151]	1	XC2V	190 $slices$	-	0.53
MICKEY-128 [158]	1	XCVE	167 $slices$	-	0.51
Phelix [129]	32	XC2S	1198 $slices$	-	0.40
A5/1 [153]	1	XC3S	57 $slices$	186	1.53
AES-128 [157]	5.8	EP1C	5058 LEs	22	0.12
AES-128 [160]	1.2	XC2S	522 $eq.slices$	-	0.07

4.7.7 Authentication Performance

During Phase 2 of eStream there was also an attempt to assess the value of integrated authentication, with Bernstein presenting a number of software results in [161]. These figures indicate that Phelix's built-in authenticator may provide some benefit for verified decryption in software when compared to separately combined stream cipher and message authentication codes, but whether this translates to advantages in the hardware environment remains an open question. Moreover, the benefits of integration may indeed be misplaced, as the system model of decryption proceeding authentication leads to inefficiencies in the quick rejection of incorrect packets.

4.7.8 Side-Channel Resistance

One notable area relating to stream cipher design absent from the second analysis phase would be that of side-channel attacks and the resource cost of masking the emanations from implementations. The importance of side-channel attack counter-measures in smartcard design has seen the field become a very active research area for block ciphers and demonstrated that in hardware the resource cost implications can be significantly.

At the logic level, techniques are generally based around masking or equalising power usage and avoiding state information leakage through glitches — typically this will involve the replacement of standard logic cells with more complex cells based around a form of dual-rail design. Tiri [162] and Popp [163] both describe implementations of the AES using such counter-measures with the former requiring 3 times the area and 4 times the power, and operating at a third the speed of the unprotected equivalent; while the later uses 4 times the area and operates at half the speed. Even the more basic dual-spaced, dual-rail design described by Sokolov [164] uses more than double the area and power of conventional AES implementations and Shang's fully balanced, asynchronous design [165] would appear similarly costly occupying $1.897\,\mathrm{mm}^2$ on a $0.35\,\mu\mathrm{m}$ CMOS process. Further methods to improve protection include Guilley's place and route technique described in [166] which, when combined with dual-rail logic cell design, results in 15 times the area usage and a halving of operation speed. These extreme lengths can be justifiable from an application perspective as system security will remain the aspect of primary importance when making design considerations.

It is likely that these logic cell based techniques would be just as applicable for protecting stream ciphers and would result in similar multiplying factors during implementation.

4.7.9 eStream Phase 3

After Phase 2 ended in March 2007 a much reduced list of stream cipher primitives [167] were advanced for a final phase of analysis in phase 3 of eStream. On both the hardware and software sides eight underlying cipher designs found themselves under consideration with those Profile II ciphers: DECIM v2 [168], Edon80 [169], F-FCSR-H v2 [170], Grain v1 [145], MICKEY 2.0 [138], Moustique [171], Pomaranch v3 [172] and Trivium [123], being complimented by a number of 128-bit key companion versions: DECIM-128 [173], F-FCSR-16 v2 [170], Grain-128 [146], MICKEY-128 2.0 [174]. Of the original five selected primitives it can be seen that three made it through as far as this final stage.

Analysis in phase 3 is due to end in early 2008 with the final portfolio of recommended stream ciphers being announced in May 2008 at the eStream Project's completion.

4.8 Discussion

From the depths of the high profile insecurities of RC4, A5/1 and E0 and the failure of the NESSIE project, the recent eStream project has seen a number of stream cipher designs that would seem to provide a great deal of promise that this area of cryptography will return to its former prominence in hardware systems.

The project has instigated the development of some very efficient stream ciphers, a number of which would be suitable for restricted hardware environments such as that expected for a secure display device. From a performance perspective the primitives Grain and Trivium would seem particularly attractive candidates, with a combination of low area, high throughput and low power that is unrivalled by the block cipher AES-128. A 6kGE implementation of these ciphers even on a relatively large 0.25 μm CMOS process results in a raw keystream throughput of up to 24Gb/s [149], indicating that the 48Mb/s data rate needed for a 320x240 colour LCD should be achievable on processes with much larger feature sizes and possibly constant grain

silicon. Furthermore the flexibility offered by the options of accelerated feedback in Grain and Trivium allows any excess throughput overhead to be utilised to reduce the area usage.

What must be remembered, however, is that while such aspects of performance are the principle reasons for a stream cipher to be chosen over a block cipher, it is data security that is the primary purpose of symmetric encryption systems. Making an insecure but efficient cipher is easy — what is more difficult is to come up with an algorithm that is both secure and efficient, as eStream set out to do. While understanding of what makes a stream cipher secure has undoubtedly been increased by the project, confidence that the candidates under consideration are secure enough to justify their key size will likely not be ascertained even by the end of the final project phase. This is especially true for the minimalist designs of Grain and Trivium where the margin for error has been pushed to the limits in a search for exceptional efficiency and means that only the further passage of time will help confer confidence.

Even more fundamental to the security equation is the question of whether justifying the key size is sufficient in itself to provide viable security to applications such as secure display. While the 80-bit key size specified in the original Profile 2 criteria for eStream is longer than that used on broken block cipher DES it is substantially shorter than keys of other widely used modern symmetric ciphers, such as SNOW 3G and the AES. Assessing this aspect of the Profile 2 eStream criteria has been the focus of the research effort for much of the research period and will be discussed at length in subsequent chapters.

Chapter 5

Key Length

In 1996 an ad-hoc group of prominent cryptographers wrote "Bearing in mind that the additional computational costs of stronger encryption are modest, we strongly recommend a minimum key-length of 90 bits for symmetric cryptosystems." [175]. Now over a decade later symmetric ciphers are being proposed with 80-bit key lengths, posing the question of whether this is sensible? The remaining chapters of the thesis aim to work towards answering this question and determining what key length should be considered secure.

5.1 Keys and Security

Stream cipher designs, like many other cryptographical primitives, place their security in a small, fixed length secret, called a key. The concept of a cryptographical key is similar to that of a key in the physical world in that while securely transporting a house may be impractical, securely transporting a front-door key is a task easily achieved by millions, if not billions, of people every day. By condensing the problem of securing plaintext data into a problem of securing a small, fixed length value, symmetric encryption schemes enable greater practicality in protecting the secret during storage and transmission.

The length chosen for this new secret value is, however, a matter of some complexity. Securely exchanging even short values over a public network typically requires a key exchange protocol based around asymmetric encryption and is expensive and slow for resource constrained environments. As such, even before aspects of

protected memory are considered it is seen as desirable to have a key size that is of the minimum length possible while remaining secure. However, to do this a determination of the effort an adversary must expend to guess a key must be made.

Moreover, much as a burglar can break into a house via a window, weaknesses in the structure of the cipher can be exploited to bypass exhaustively trying each key combination. These forms of algorithm specific cryptographic attack, collectively described as algorithmic attacks (see Section 8.1.1), put further strains on the choice of key length as a cipher with a longer key must justify it through added resistance at the algorithmic level. This leads to a trading of performance considerations against security and puts further pressure on ciphers designed for resource constrained situations to use small keys. Nonetheless, any key length security trade-off is only reasonable if the cost an adversary faces in retrieving protected data remains above the value they would place upon it.

After some debate at SASC 2004 the target key length chosen in the eStream Profile 2 call for stream cipher candidates would be set at 80-bits. Later chapters will attempt to assess the economic value of this key size, however in this chapter general aspects surrounding key size selection will be addressed, including those inspired by Bernstien's 2004 paper [176] which suggests that the complexity of exhaustive search is overestimated when compared to algorithmic attacks.

5.2 eStream Specification

The original eStream call for stream ciphers primitives made in 2004 specified two key lengths as outlined Section 4.5.3: one of 128-bits for algorithms aimed at software, and one of 80-bits for those aimed at hardware. 128-bit key lengths have been widely deployed in symmetric encryption schemes before, including the AES, and a good deal of confidence exists that such a key length is secure against a computationally bound attacker. However the 80-bit key size has only had one notable deployment prior to 2004, in the NSA block cipher SKIPJACK [177], and the value of this key length is much less well understood.

Theoretical assessments made of 80-bit key lengths were described in Section 7.2 of the 2004 ECRYPT key size report [178] as providing a security protection level in symmetric ciphers suitable for "very short-term protection against agencies, long-term protection against small organzations" and as the "smallest general-purpose

level" which "protects against the most reasonable and threatening attack scenarios". These comments would seem to indicate that the chosen security level is right on the edge of current feasibility.

Following this a similarly focused report [179] by the NIST (National Institute of Standards and Technology, U.S. Department of Commerce) recommended in section 5.6.2 that the use of 80-bit key lengths would be insufficient for "Government unclassified applications" by 2011 and that a minimum 112-bit security level would be needed after this. With the eStream project not due to finish till May 2008 this would seem to give any recommended algorithm from Profile 2 less than three years before it would essentially become obsolete; a possibly terminal indictment for the commercial acceptance of new stream ciphers.

The eStream committee have nonetheless maintained the 80-bit key length target for Profile 2 candidates throughout the competition process, commenting in 2006 that they "still see value in this key length" [180] and again in 2007 that their "focus is on ciphers with 80-bit keys" [167].

5.3 Brute Force Attack

If the key length of a cipher is sufficiently small an attacker can feasibly and successfully try every combination possible in a methodical manner until they succeed in decoding the ciphertext message into its original plaintext. This exhaustive search of the key space for the correct key is generally known as a brute force attack and is a generic attack against all ciphers except the one time pad.[1] As keys may be tried in any order, brute force attacks need not operate in sequential fashion and can be accelerated through the introduction of parallelism.

These parallel brute force attacks on key space offer many advantages when all practicalities are considered in that their methodology of operation is simple leading to efficient and low cost implementations, a fact emphasised by Bernstein in [176]. In this same paper Bernstein would go on to suggest that in comparison to algorithmic attacks brute force is an underestimated attack form and that the direct link between key length and a symmetric cipher's security level should be reconsidered; these are ideas that will be returned to in Chapter 8.

[1] In a one time pad the correct plaintext will be indistinguishable as every possible plaintext output will be produced by the brute force process.

As already stated, brute force attacks are generic in nature and relevant to all ciphers; this does not, however, imply that an attack will operate with a universal equality of efficiency. The time taken to check a key will vary for different primitives due to the attack's dependancy on key set-up time and the specifics of the design implementation. As indicated in Chapter 4, well designed stream ciphers can significantly outperform ciphers such as AES-128 in the hardware environment since modern block ciphers must process a full block of data alongside the key, while the operation of a synchronous stream cipher is almost data independent. Furthermore, block cipher implementations often require speed to be traded for low area implementation whereas a stream cipher can usually provide both properties without the need for trade-offs. Consequently, brute force attacks on a stream cipher may potentially be many 'bits' easier than raw differences between block and stream cipher key length would suggest.

Traditionally the area of brute force has focused on block ciphers and although much research has gone into brute force key search machines aimed at block ciphers (see Section 6.2), the area of brute force key search machines for stream ciphers has until now received little attention.

5.4 Stream Cipher Brute Force

When the aim is a brute force key search, not only must cipher throughput be considered, but also the time required for set-up. Indeed for stream ciphers this is the factor that dominates the key search process: even if the stream cipher produces a single bit of keystream per clock cycle, after just one iteration half of the searches can be discontinued (e.g. the keystream bit is a 1 but the search target starts with 0) and after gathering a byte's worth of material 99.6% of searches may be stopped. Set-up time is still required in block ciphers for key expansion but this takes up proportionately less of the key search effort simply on account of the fact that as a full block of output bits must be generated with every search, their resource usage is fundamentally greater.

5.4.1 Data Enhanced Brute Force

Brute force attacks do not have to be restricted to using a single data sample; as briefly touched upon by Hellman [181] additional data points may be utilised to

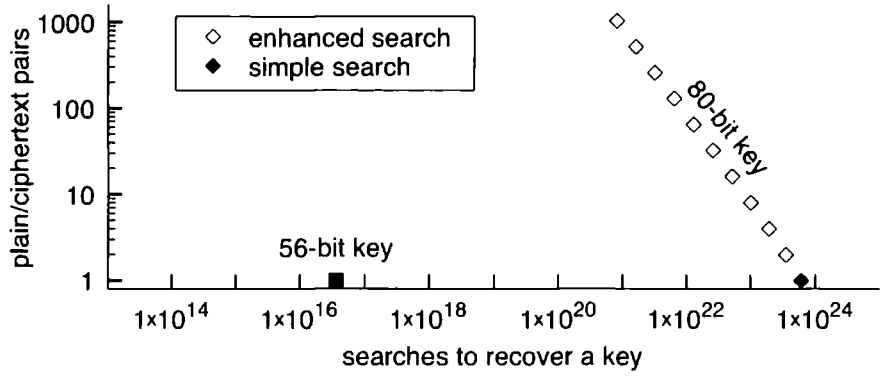


Figure 5.1: Impact of plaintext/ciphertext data pairs on key search. The 80-bit key is brought closer with more data pairs to the vulnerable 56-bit key length of the Data Encryption Standard; in linear terms, considerably closer.

increase the probability of an attack finding a successful match due to the birthday paradox [182]. This form of data enhanced attack has the potential to make searching for a key in an implausibly large key space, feasible.

The reality and efficiency of such enhanced 'meet-in-the-middle' attacks has previously been demonstrated for block ciphers by Clayton in [183] where a single FPGA device was used to attack the security protecting the ATM infrastructure. Although the attack relied upon a flaw in a version of the Common Cryptographic Architecture (CCA), much of the attack's extreme practicality came from 2^{16} pre-calculated output values stored in memory to which a comparison could instantly be made.

The potential for data enhanced brute force attacks when applied to synchronous stream ciphers is even greater. The data independent keystream generation and the widespread use of simple XOR combining functions means that pre-calculating original keystream material from collected plaintext-ciphertext pair data is trivial. In a situation where multiple plaintext-ciphertext sections encoded under different keys are available the search effort required to recover one of these keys rapidly falls (Figure 5.1). Moreover, as multiple derived keystreams can be simultaneously checked at effectively zero time cost, due to keystream independence from data, computational requirements will be minimised.

The sheer effectiveness of the data enhanced attack variation to brute force can be shown by the example of eight plaintext-ciphertext pairs being available to an adversary, reducing the problem of finding a key match abruptly from $\frac{1}{2^{80}}$ to $\frac{1}{2^{77}}$ as there are now 2^3 as many outcomes considered positive. This translates to over

5×10^{23} fewer searches. It should be noted that in the context of stream ciphers the attack variation does require a consistent IV between plaintext-ciphertext pairs for the advantage to be realised; the practicality of this assumption is outlined in Section 6.3.

5.4.2 Probability of Success

As the observation of data-enhanced search would motivate the design of the DELUGE keysearch engine, it is important to quantify the probability of a match being found so that an economic valuation may eventually be made.

Clayton suggests [183] that the probabilistic reduction in search effort can be described by a Poisson distribution with the "probability that the first r attempts will all fail is $e^{-\lambda r}$ where λ is the probability any given attempt matches". If again 8 plainstreams are looked at λ will be 2^{-77} and so with average luck ($p = 0.5$) this will equate to a match being found after $-ln(0.5)/2^{-77} = 2^{76.471}$ attempts.

However when the number of positive outcomes are small, as in the given example, and the key search engines are as described in Chapters 7 and 6, calculation using such a Poisson distribution will tend to overestimate the complexity of the search process. This is due to the probability of a successful match increasing with time as possible keys are systematically eliminated and so not being a constant. The true probability, P, of a match after r attempts will instead satisfy the equation:

$$P = 1 - \frac{q!(n-r)!}{n!(q-r)!} \tag{5.1}$$

where n is the total number keys and q is the number of incorrect keys. The equation can be made practical to calculate using a Stirling approximation for the factorisations:

$$\ln x! \approx x \ln x - x + \frac{\ln x}{2} + \frac{\ln 2\pi}{2}$$

As n and q will generally be large, the shortened Stirling approximation, $\ln x! \approx x \ln x - x$, is appropriate. Thus after taking logarithms of both sides, equation (5.1) becomes:

$$\ln(1-P) \approx q \ln q - n \ln n + (n-r) \ln(n-r) - (q-r) \ln(q-r)$$

which by rearranging gives an approximation to the probability of a successful match:

$$P \approx 1 - e^{[q \ln q - n \ln n + (n-r) \ln (n-r) - (q-r) \ln (q-r)]} \tag{5.2}$$

For very large numbers (i.e. $n, q > 2^{40}$) accurate calculation of this probability (5.2) becomes computationally difficult due to precision limitations, making use of this formula impractical. However, since the number of incorrect keys q will tend to approximate the total number of keys n, in such situations equation (5.2) can be further approximated to:

$$P \approx 1 - e^{\left[(n-q) \ln \left(\frac{n-r}{n}\right)\right]} \tag{5.3}$$

Letting p be the number of correct keys so $p = n - q$ and rearranging (5.3) then gives a revised equation for the probability of a successful match after r attempts:

$$P \approx 1 - \left(\frac{n-r}{n}\right)^p \qquad n \gg p \tag{5.4}$$

Where p is small, equation (5.4) will provide improved accuracy over that of a Poisson distribution and is therefore used to calculate the search effort required in the presented key search engines of Chapters 6 and 7. Calculations in the related papers [1] and [2] use the much simpler division calculation which has a tendency to underestimate the search effort required: the level of this underestimate will, given average luck, be below 39% — a figure that translates into less than 9 months of computational advancement assuming a continuation of Moore's prediction [184].

5.5 Discussion

The suggestion by ECRYPT that a key length of 80-bits may be teetering on the brink of feasibility, combined with the potential for stream cipher brute force to be potentially less expensive than targeting block ciphers, raises the hypothesis that Profile 2 stream ciphers such as Grain and Trivium may not be suitable for secure display purely from the security context surrounding their key length. The further gains offered by the described data enhanced brute force attack only serve to raise the priority of resolving this question.

To find out if an 80-bit key length would be appropriate it is sensible to consider the construction of a key search machine that specifically targets stream ciphers in

this manner and in the following two chapters such development is outlined.

If such a machine is to provide confidence over the security of the 80-bit key length, in this context it must operate on the best case scenario for a adversary. As of 2005 Grain v0 was among the highest performing and lowest resource entries to eStream (see Chapter 4) and with the addition of a parallel loading key interface it was found to load, initialise and produce two bytes of keystream in just 11 clock cycles. This high efficiency in key agility and resource usage that was so beneficial to the cipher's appeal would ironically identify it as the most susceptible to brute force key search. Consequently, with the aim to establish an upper limit on the potential of 80-bit key search, Grain v0 was used as the principle keystream generator in the key search machine developed between 2005–2006 and described in Chapter 6.

Chapter 6

Key Search Machines

ALGORITHM security is a significant issue of concern in the field of stream cipher design and is something that the eStream project has set out to address. Yet the comparatively short target key length for Profile 2 proposals raises another more fundamental question: if an algorithm is indeed ideal, what is the economic value of the protection being provided? In making such an assessment, all forms of generic attack must be considered including that of brute force; making it prudent to consider the design of a key search machine.

Brute force key search has already proved itself to be a practical attack form against some block cipher designs. However, its potential in the field of stream ciphers has not received similar public attention. As set out in the previous chapter exhaustive search attacks aimed at a steam cipher may find great levels of efficiency due to their method of operation. Thus a resolution of this gap in understanding is of paramount importance if a reasonable security valuation is to be made.

This chapter sets out to address the design considerations needed in creating a stream cipher key search engine based around the data enhanced form of brute force outlined in Section 5.4.1. The valuation of security and the recent history of key search machines is first examined before the more specific design considerations involved in stream cipher brute force are addressed. Later in the chapter, a FPGA based architecture capable of performing successful key search will be described and analysed. This design, for a key search architecture targeting stream ciphers, would originally be presented in early 2006 [1] and as such, focuses on ciphers of the first eStream evaluation phase.

In should be noted that the hardware key search engine described in this chapter should be considered only as a theoretical design, as although it is fully expanded it has a number of shortcomings that would prevent deployment. Resolving these shortcomings would lead to the development of the fully practical hardware key search engine which is described in Chapter 7.

6.1 Data Value

As mentioned in the Chapter 5, trading security for a shorter key length is only reasonable if the cost an adversary faces in retrieving protected data, remains above its value to them. This value is determined by a number of factors in the commercial world, but will typically have a time dependancy: in some situations, such as a financial transaction, data will have a high initial value but its value will then decay rapidly within fractions of a second; others, such as proprietary multimedia transmission, will have a lower value to an attacker but a relatively gentle value decay curve that may extend over weeks, months or even years.

Value is also determined by the availability of other methods for the attacker to receive the data, and for example, Eve, who wants some data from Alice Corp., would probably find it much cheaper to bribe Andy who works there than analyse the algorithms used and construct a multi-million pound key search machine. Nevertheless, there is a tipping point where advantages to an adversary of a key search machine will exceed other possibilities, making their analysis crucial to allowing a well-informed decision to be made.

6.2 History of Key Search

As already mentioned in Section 4.3.1 the Lorenz cipher used during World War II was attacked by U.K. cryptographers through use of the first digital electronic computer, Colossus. The computing resource, made available by this machine, aided efforts to decrypt intercepted messages through testing possible initial rotor settings. As such it operated effectively as the first electronic key search machine. An account of the Colossus development is given in the declassified document 'General Report on Tunny' [82, 83] an electronic version of which can be found at [185].

While Colossus was a key search machine, or more correctly an IV search machine, it did not try to test every initial combination in a brute force manner but instead exploited weaknesses in the underlying algorithm of the Lorenz machines. It would take another 30 years for a key search machine based on exhaustive search to be publicly proposed. Diffie and Hellman in their 1977 paper [186] conceive of a parallel architecture key search machine made from millions of ASIC chips, which they optimistically estimated to be capable of recovering a key in a day at a cost of $20 million. Further theoretical key search machine designs, attacking the DES block cipher, were later published by Wiener [187] and Goldberg [46]: Wiener's design suggesting a $1 million machine in 1994 would be capable of recovering a key in under four hours and Goldberg introducing the idea of low cost FPGA usage in key search.

Around this time the RSA Secret Key challenge was launched [188] challenging cryptographers to break the DES for a 24-character known-plaintext. This would lead to a number of distributed network attempts at breaking the cipher, as well as the first implemented hardware design for DES key search — 'DES Cracker'. Developed by the Eletronic Freedom Foundation in 1998 [46], DES Cracker was able to recover a 56-bit DES key in 56 hours for an estimated cost of $210,000. Its importance as an implemented hardware key search system means we return to consider its design details in Section 6.2.2.

The DES has remained a popular target of more recent key search machine designs, and a number of suggestions have been made to improve search efficiency further. As metioned previously in Section 5.4.1, Clayton [183] introduced a data element into his 2003 DES key search system to improve online search efficiency, and create a very low cost practical attack. Time-memory trade-off [189] may also improve such attacks.

One of the most potent demonstration as to ease with which the DES's 56-bit key space can now be searched is Kumar's [190] FPGA architecture implementation. This flexible, parallel search design capable of recovering a key in 9 days, is based around the generic computation engine 'COPACOBANA' and cost less than €9,000 to build. COPACOBANA is further discussed in Section 6.2.3.

6.2.1 Key Search in Asymmetric Encryption

Key search machines are not limited to symmetric encryption; they may also target the field of public key cryptography. Public key general number field sieves (GNFS) have already successfully targeted 663-bit numbers [191] of the RSA cipher via a software implementation and development of hardware based architectures has also seen a great deal of interest. One of the early proposed machines TWIRL [192] was instrumental in demonstrating in 2003 the vulnerability of using key lengths shorter than 1024-bits in factorisation based algorithms.

A number of architectures have been proposed to further advance the key search in the field of number factorisation: SHARK [193], indicated that even a 1024-bit factorisation may be possible given the right level of resources while CAIRN 3 [194] aimed to show that targeting a 768-bit key size is possible and relatively low cost. Other ideas for how to conduct the sieving process have also been proposed with Pelzl's design [195] implementing an elliptic curve method.

A relatively recent concern has been that quantum computers based on qubits potentially turned some hard problems such as factorisation into easy ones by making use of Shors Algorithm [196], potentially destroying the security of the current public-key cryptography structures such as RSA. However, first implemented at the 2-qubit level in 1998 [197, 198] it has taken till 2004 for quantum computers to reach the 7-qubit level [199] that would allow the factorisation of the 2-digit number 15. Even the 12-qubit computer described in 2006 [200] is a long way short of factorising the large numbers needed for these computers to be useful in cryptanalysis.

More recently an attempt [201] has been made at attacking ECC public key based systems through use of parallel version of Pollard's rho algorithm [202]. Pelzl [203] estimates the cost of attacking an 112-bit ECC key using COPACOBANA at 262 days.

6.2.2 DES Cracker

The Electronic Freedom Foundation 1998 DES Cracker [46] (often referred to as 'Deep Crack') would be the first implemented hardware architecture to target the DES. By its 56 hour brute force attack on this cipher it demonstrated the high effectiveness of hardware parallelism in key search systems. Moreover, its non-theoretical design makes it of particular interest when considering key search machines.

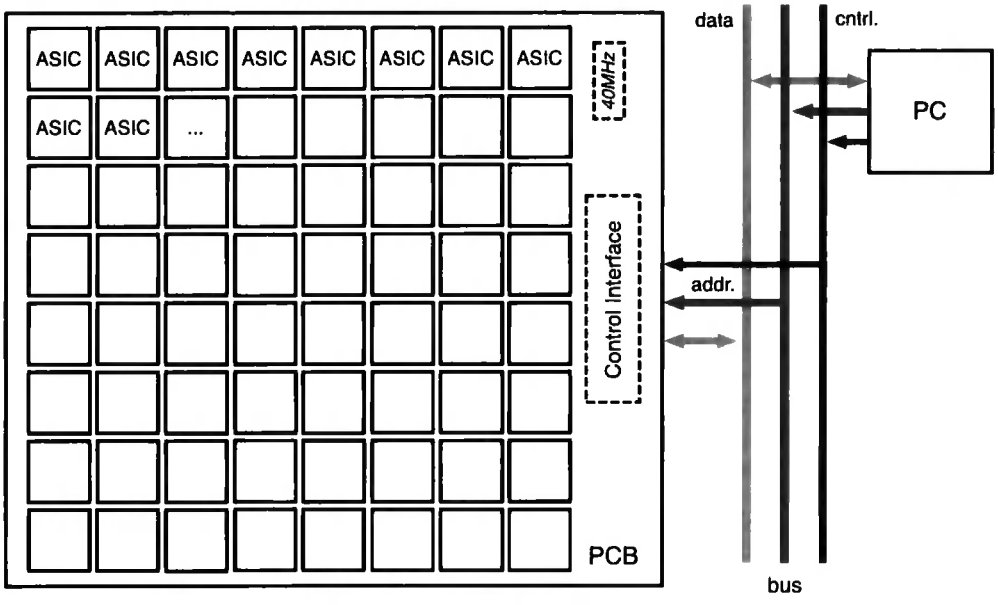


Figure 6.1: System layout of EFF's DES Cracker

The structure of the DES Cracker (Figure 6.1) is based around custom ASIC (application specific integrated circuit) chips, implementing 24 search units and running at 40MHz. Each search unit performs a 16-cycle DES decryption and compares the result to see if its a known-plaintext or a series of recognised characters. A unit is loaded with 24-bits fixed key material by the PC (personal computer) and operates on its own small 32-bit area of key space until it is finished. If it identifies an interesting result the search unit passes the key value to the controlling PC for full analysis in software.

In the 1998 DES Cracker, these custom ASICs were organised into boards of 64 chips, with the full system containing 29 boards. The full key search machine averaged just under 89 billion searches a second when attempting the RSA DES II-2 Challenge in which it was successful. The whole machine cost $210,000 of which $130,000 was for materials, such as the ASIC chips and boards.

6.2.3 COPACOBANA

COPACOBANA [190] was first demonstrated in SHARCS'06 alongside the stream key search engine presented later in this chapter. It is not specifically a key search engine but its computing structure is flexible enough to enable highly effective search engines to be programmed on to it.

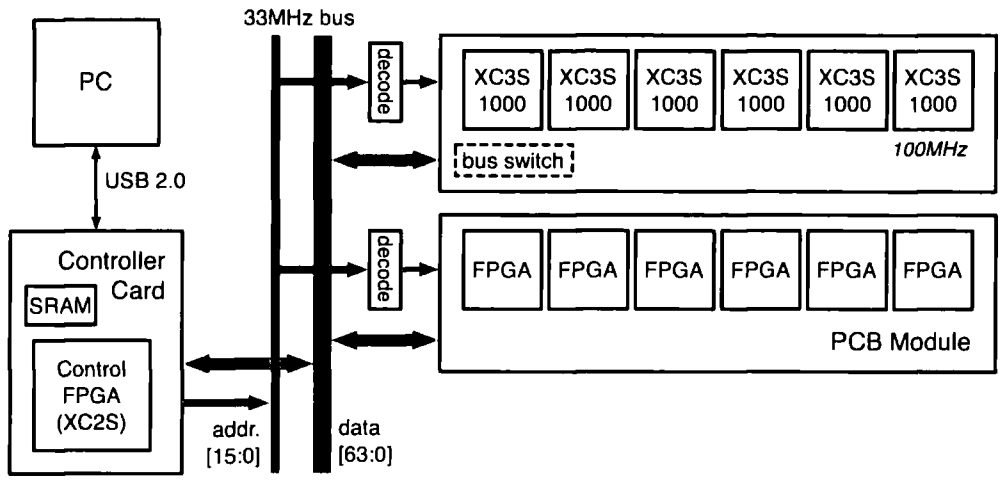


Figure 6.2: Inter-unit buses and chip layout of COPACOBANA

The design of COPACOBANA (Figure 6.2) is based around a rack of 1–20 PCB (printed circuit board) modules, each containing six low cost Xilinx Spartan III FPGAs. An address decoder and data switch that accompany each board, allow a PC, through the USB controller card, to communicate with individual FPGAs.

When configured to implement DES key search each FPGA is programmed with four fully-pipelined DES implementations operating at a 100MHz. Each DES core has a 64-bit comparison unit and a 2-bit ID associated with it. This ID number uniquely identifies the core and forms the 39th and 40th bits of the key with the other key bits coming from a counter and a fixed register programmed, on a flexible basis, by the PC. Positive matches are, like in DES Cracker, reported back to the PC for software analysis.

The full COPACOBANA based DES key search machine averages 48 billion searches a second and obtains a key in an average of 8.7 days. The cost is detailed at €8,980, of which close to half is from the cost of purchasing the FPGA devices.

6.2.4 Discussion

The level of analysis on the DES means that block cipher key search is well understood and predictions can confidently be made over the security provided by a particular key length. In asymmetric encryption schemes, the level of knowledge and understanding of key length security is also well developed. However, there is a complete absence of public designs based around stream cipher primitives, leaving

estimates as to the security of 80-bit stream cipher key lengths uncertain. To resolve the hypothesis of Chapter 5 it was therefore proposed in 2005 to construct a key search machine targeting the efficient Grain v0 stream cipher. The design considerations and implementation aspects of the proposed machine form the subject of discussion in the remaining sections to this chapter.

6.3 Attack Scenario

Server to user content transmission is a common scenario in which the efficiency and speed of hardware stream ciphers are ideally suited. In this situation it is likely that an adversary may be able to collect a number of these transmissions and use known header information to gain the plaintext/ciphertext data pairs necessary to recover the initial bytes of multiple keystreams, allowing an efficient enhanced search. It is reasonably assumed in the design of DELUGE and the key search machine in this chapter, that an adversary would be unconcerned over which user's data is retrieved and that differing IVs could be filtered out. The fact that counters are a common and easy way of producing unique variables such as an IV and that IVs typically are transmitted on open unsecured channels tends to support this later supposition.

In recent times there has been some debate over the role of IVs [204], however an IV and a key are crucially different in that the later is deemed critical to the security of data and former is not — if the key is no longer a secret the message secret is lost: the condensing of secret information is the aim behind a stream cipher and differentiates it from the open time pad. If the criteria for IV is made stricter to require randomisation and possibly secure transmission, there is little advantage to the user over defining the cipher to have a larger key size in the first place (the use of keys larger than the supposed security level is in fact proposed in Chapter 8 but for differing reasons).

6.4 Platform Choice

Before the design of a system could begin a considered choice of platform has to be made between ASICs (application-specific integrated circuits), as seen in DES Cracker, and FPGAs (field programmable gate arrays), as later seen in COPACOBANA.

The ASIC allows system design in rawest form with control of individual transistors possible. It provides the maximum performance per unit of substrate area and offers the potential of a very efficient and high-speed key search engine. However, ASIC design is currently a complex, high cost process with shrinking feature sizes constantly introducing new challenges to digital design due to changing physical device characteristics. These problems are only likely to escalate.

FPGAs offer an alternative approach and add a extra layer of abstraction in the design process, which allows the designer to concentrate on system design and not physical layout. Their ever growing popularity has led to FPGA progress being incredibly rapid over the past two decades (see Section 6.6.1) with falling prices and increased logic element counts. This is a trend which looks set to continue as the high overheads of nanometer processes encourage migration away from ASIC. The FPGA has no overhead costs associated with purchases and offers a flexible, inexpensive environment for digital designs. The programmable nature of the devices has a further effect especially relevant to attack architectures in that this allows an adversary to buy blank devices — there is no need to explain to a large multinational why a hundred thousand chips with ciphers on them are needed.

The rapid turn-around, low design costs, covert purchasing and low unit cost are very attractive features to a commercially based adversary, and combined with the reasonable circuit speeds of modern FPGAs a decision was made that the design work should focus on FPGA based systems.

Having chosen a platform a second design choice is whether the design should follow a asynchronous or synchronous approach. FPGAs are largely designed for the latter approach, with a register in every logic cell and specialist routes for global clocking to ensure synchronous operation. However, another reason is design simplicity. On the micro-module scale the design effort of creating data acknowledgement channels adds an extra level of complexity to design which can outweigh any possible speed advantages of a logic reduction. This leaves the main advantage of asynchronous design as that of power reduction, a factor not typically of concern in key search system design.

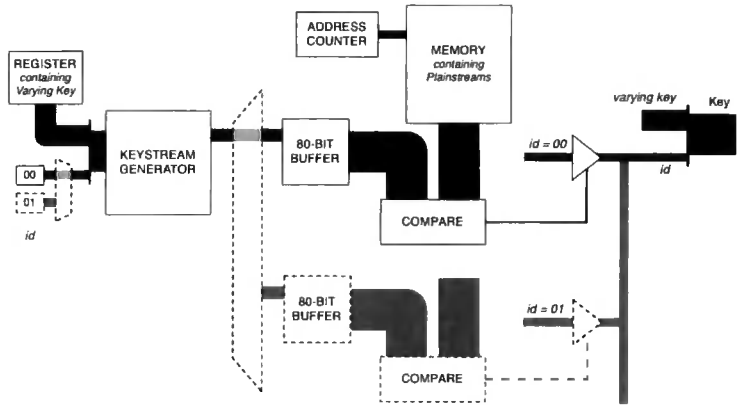


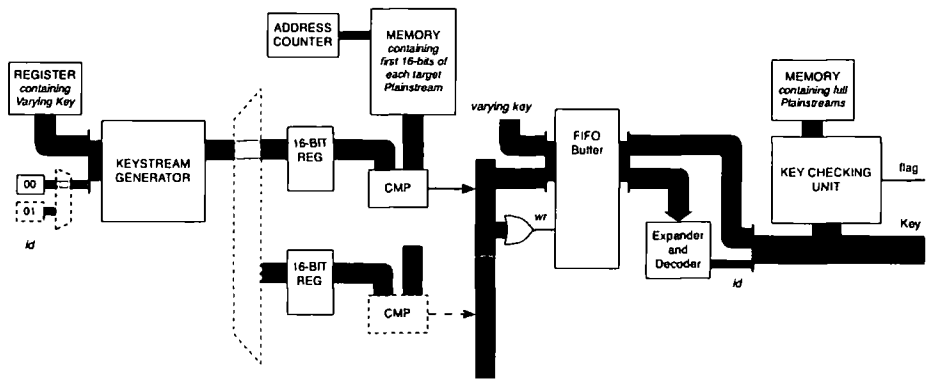
Figure 6.3: System diagram of a simple key search system

6.5 Key Search System Design

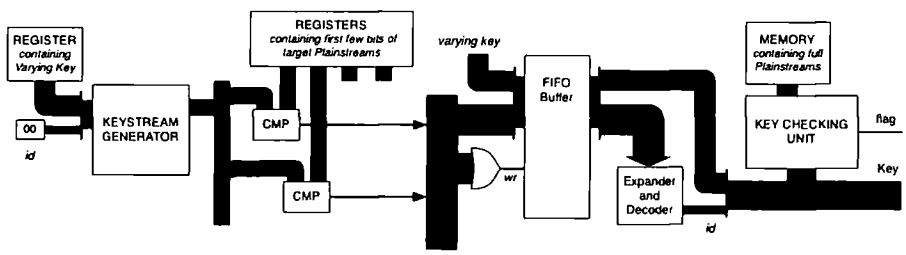
Possibly the simplest brute force key search system is one that takes a single stream cipher and varies the key to produce unique keystreams for comparison with a 'plainstream' (the keystream retrieved from a plaintext-stream/ciphertext-stream data pair). Assuming the cipher uses an 80-bit key, then checking 80-bits of keystream material with a known plainstream on average produces one positive match over the entire key space. When eventually a match is detected it is likely that the key in question has also been used to produce the original keystream.

As it has been assumed that multiple data pairs with matching IV are available, it is reasonable to extend the system so that it stores the keystream and checks each plainstream in turn while the cipher is reinitialising. This leads to the design shown in Figure 6.3. If the number of plainstreams is great enough the memory comparison may take much longer than key loading and initialisation making the use of additional memory comparison units desirable. Such units have then to be identified, and as in Figure 6.3 it is convenient for this identifier to form part of the key.

Although simple, this architecture presents a number of disadvantages to high speed, resource efficient key search with the majority of these founded on the need for 80-bit comparison of plainstream and keystream. Reliable comparison is desirable to minimise the need for external computation but by introducing a two stage process efficiency can be greatly improved: the first stage comparing a small amount of keystream material, eliminating the majority of keys; and the second stage performing a full comparison on those that are left. For large systems the additional resource required for a back-end element is far outweighed by the reduced memory



(a) Serial Search - memory-based search system



(b) Parallel Search - data enhanced brute force

Figure 6.4: System diagram of two approaches to key search

requirements, logic and routing complexity seen at the front-end, so long as the key elimination rate is high.

The 16-bit comparisons shown in the design outlined in Figure 6.4(a), are sufficient to eliminate 99.9985% of keys and combined with the FIFO buffer, allow a small key checking unit in the back-end to function in unison with a front-end that is orders of magnitude higher in performance. When the number of plainstreams available is large this serial search system becomes highly efficient as each sequential memory search can continue without performance penalty while other search units are being loaded with new keystreams.

The other theoretical key search system design considered (Figure 6.4(b)) is less memory-based and involves a stream cipher producing a small amount of keystream which is immediately checked against multiple plainstreams. When the number of plainstream comparisons is small this leads to a very low resource, high performance key search system without the control complexities required for memory-based approaches.

It is noticeable that the described memory-based architecture (Figure 6.4(a)) can

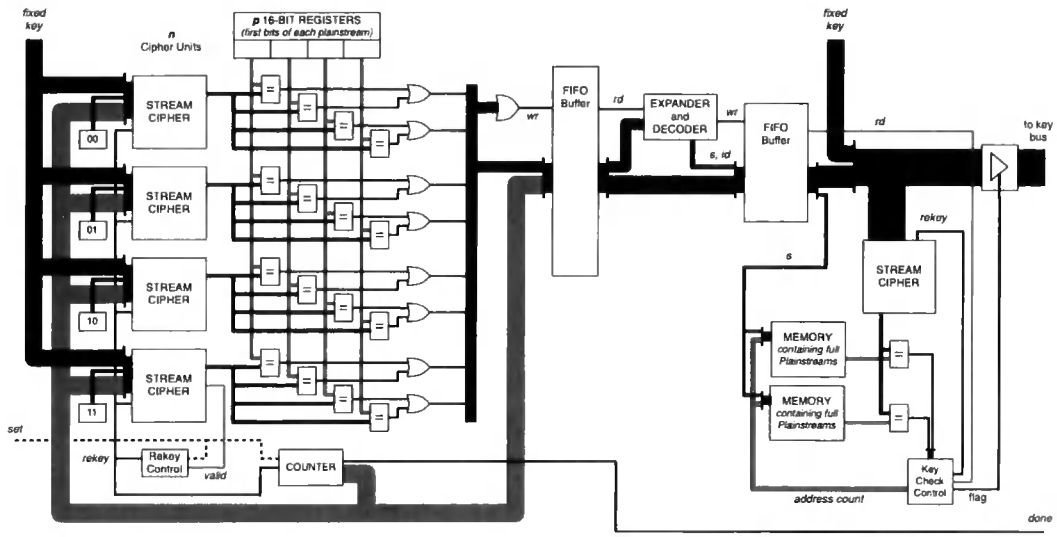


Figure 6.5: System diagram of DELUGE-0 chip module with n=4, p=4

utilise a much higher data element than the parallel search system (Figure 6.4(b)). Thus if an application generates a large number of useful plainstreams for a potential attacker then this serial search design is likely have a lower resource overhead and so be the optimal low-cost option to key search. Nevertheless, in designing a key search machine it is not only desirable to minimise the brute force computation required and the addition of a large memory element precludes a system from use in many otherwise applicable situations. Parallel search has its optimal data requirement low enough that a machine based on this system would be applicable to a wide range of applications and thus is a good compromise for the initial evaluation system, DELUGE-0.

6.5.1 DELUGE-0

The theoretical parallel search method forms the basis of the DELUGE-0 design and it is envisaged that a large number of independent search modules, each occupying a FPGA device, would be lightly controlled by a central computer. A potential module of the DELUGE-0 key search system is shown in Figure 6.5. The 16-bit data path architecture at the front-end of the module helps minimise the necessary depth of the FIFO buffer, while still allowing the Grain v0 keystream generator to run with a minimum initialisation set-up.

Simultaneous key loading is used to simplify control of the system but this consequently requires the buffer to accept a high number of front-end search outputs

in parallel, which in turn necessitates a second smaller but deeper buffer to store outputs once expansion and decoding has taken place. The system diagram also shows outputs from comparison units being combined in pairs before being sent to the primary buffer. Although this requires memory and comparison unit duplication at the back-end of the system the speed and resource advantages of reduced buffer width more than compensate, particularly when the number of search units n and the number of plainstream registers p are large.

The key used in loading is split in three parts with first being a regularly changing variable provided by a system-wide counter, the second being a short identifier that is unique to each cipher unit and the third being an externally controlled fixed register variable that is unique to the chip. By carefully selection of the register value loaded on completion of each counter cycle,[1] multiple chips can accomplish a search of the 80-bit key space. A small control unit (not shown in Figure 6.5) that controls loading of this register is also responsible for providing the desired degree of system flexibility in IV and plainstream memory values.

At the system output a relatively simple set-up is used with the key being tristated onto an 80-bit bus when a match is found: the probability of two chips in DELUGE-0 finding a match simultaneously too low to be of concern. Further, as a match may be down to pure chance or more than one plainstream-key pair may be desired the final system caters for an acknowledgement signal to resume the paused chip. A pseudocode description of DELUGE-0's key search operation is given in Section 6.7.1.

6.5.2 DELUGE-0 System Testing

Initial simulation testing of a DELUGE-0 single chip core was carried out for the Altera Cyclone EP1C20F400C6 using Quartus II 5.0SP2. The results imply that the optimal configuration for brute force on this system is 16 search units per chip, each with a plainstream search space of 16. The configuration was estimated to run at 100MHz and use 12018 logic elements,[2] while the use of 4 back-end comparison units restrict memory requirements to just 19kbits. Sub-system resource usage is further detailed in Table 6.1.

[1] Use of a LFSR based counter in the final system necessitates that one chip in DELUGE-0 should be programmed differently, with the counter being held at zero and the fixed key being driven by a counter to mitigate the former's inability to cycle through an all zero state.

[2] Excludes resources used to tristate the key output.

Table 6.1: Resource Usage for Key Search System on a EP1C20F400 FPGA

Sub-System	Memory Bits	Registers	Logic Elements	No.
Counter	0	64	66	1
Stream Cipher	0	160-166*	496-505*	16
Comparison Unit	0	0	103-160*	16
Plainstream Register	0	16	16	16
Primary FIFO	12160	46	71	1
Front-End (n=p=16)	12160	3189	10995	-
Expander-Decoder	0	73	204	1
Secondary FIFO	4736	46	64	1
Translation Unit	4736	119	269	-
Stream Cipher	0	166	493	1
Plainstream Memory	512	0	0	4
Compare	0	2	11	4
Check Control	0	20	26	1
Back-End Test	2048	212	581	-
Key-IV Register	0	109	109	1
Loading Control	0	0	20	1
Total	18944	3681	12018	-

6.6 Results

If a well informed decision is to be made on the use of 80-bit keys, the performance figures of the DELUGE-0 system simulations must be translated into economic terms so a comparison to data value can be made. Additionally, although the potential of key search systems in the current technological situation is important, users of cryptography algorithms need to be confident that security is maintained throughout a product's time to market — as such, estimates as to their future practicality are also of interest.

6.6.1 FPGA History and Future Roadmap

After the conception of the FPGA in 1984 early chips were relatively small and novel devices: the 1985 Xilinx XC2064, designed for a 2μm process, made available just 64 logic blocks for logic configuration [205]. 21 years later, FPGAs are commonplace devices that are found throughout electronic systems with modern high performance chips such as the Xilinx Virtex-4 XC4VLX200 designed on a 90nm process and

* Sub-system resource usage varies per instance due to automatic optimisations made at compilation.

containing advanced logic blocks that number in the hundreds of thousands. Looking at this substantial progress, it is clear that the field of FPGA development is fast moving. Furthermore, the performance gains to the designer are unlikely to slow in the near future as the simple structure of FPGA's repeated cells makes it likely the full benefit of future nanometer process nodes will be realised. Moore's law [184] predicts that every 3 years the cost-performance ratio should increase by a factor of 4, a hypothesis that would appear to be adhered to by the previous three generations of low-cost Altera FPGAs [1]. That Moore's law should hold in the near future is the presumption on which the calculations of future chip cost presented in Section 6.6.2 are made.

Chapter 7 will again review these figures for predicted performance in light of more recent FPGA developments, which may indicate that the economics of search machines may in fact deviate from Moore.

6.6.2 Extrapolating Keysearch Results

The EP1C20 simulation device was a very economical chip at its time of release, however by late 2005 it was a generation behind current low cost, 90nm FPGAs: Altera's 'Cyclone II' and Xilinx's 'Spartan III'. The most economical Altera offering from this 90nm generation was the Cyclone II EP2C35 with a 250k unit volume pricing of $22 per chip [206]. Testing showed that the slowest speed-grade of this chip accommodates a n=32, p=16 DELUGE-0 search system running at 86MHz.

Altera also offers a technology called 'HardCopy' which is a form of structured ASIC. In this a generic chip, with FPGA style cell structure, is mass manufactured without the top few metallisation layers, these layers are then added at a later stage to customise individual devices. Although such a manufacturing process sacrifices the FPGA's covert nature of design and programming flexibility it does allow lower unit costs and higher performance. The HC210W chip, the most economical of the high end, 2005 HardCopy II devices, had a unit cost of $15 at volumes of 100k and a further non-recoverable engineering cost of $225k [207]. Design simulations on Quartus II 5.0SP2 indicate that such a device allows a n=64, p=32 DELUGE-0 search system to fit on the chip. A lack of timing information means that the 230MHz clock speed used in cost calculations is an estimated figure based on the fact that a n=64, p=16 DELUGE-0 search system runs at 115MHz on the 'Stratix II' EP2S60 and that this high performance FPGA family runs on average at half the speed of HardCopy II [207].

These Cyclone II and HardCopy II results allow two tables to be constructed (Table 6.2 and 6.3) which present the chip cost an adversary would face in building a DELUGE-0 system for the regular recovery of a key within the specified time-scale. It is worth noting that in constructing and running a key search machine a substantial number of other costs would likely be incurred: PCB manufacture; cost of racks to hold the PCBs; cost of a computer to control the system; wiring; power regulators; programming chips; air conditioning installation; cost of the building housing the machine; power consumption; logistical cost of handling high numbers of components. These costs add substantially to the overall expense of any brute force attempt but it is still probable that chip cost will still dominate a final bill of materials.

Table 6.2: Estimated Chip Cost of 80-bit Key Brute Force on EP2C35 and future equivalents

Recovery Time	2005	2010	2015
1 day	$61 billion	$15 billion	$3.8 billion
1 month	$2.1 billion	$520 million	$130 million
1 year	$160 million	$42 million	$10 million

Table 6.3: Estimated Chip Cost of 80-bit Key Brute Force on HC210 and future equivalents

Recovery Time	2005	2010	2015
1 hour	$88 billion	$22 billion	$5.5 billion
1 day	$3.8 billion	$920 million	$230 million
1 month	$180 million	$45 million	$12 million
1 year	$10 million	$2.9 million	$930k[a]

The figures outlined in Table 6.2 indicate that practical FPGA attacks which take under a month to obtain a key are many years away. However, the estimated cost a 1-year search in ten years time appears be low enough to be applicable in many situations. The structured ASIC results (Table 6.3) cause greater concern as even today, $10 million worth of devices makes a one year attack possible, a figure potentially viable for a large organisation [175].

6.6.3 Applicability to Other Algorithms

The Grain v0 keystream generator used for the described key search machine, was by 2006 found to be algorithmically insecure (see Section 4.7.4). This affects the

[a] Cost likely to be higher as below volume threshold.

Table 6.4: Keystream Generator Efficiency for Key Search on EP1C20

Keystream Generator	Logic Elements	Clock Rate	Clk/Key	Keys/LE.s
Grain v0	498	229MHz	11	41803
Grain v1	543	211MHz	11	35330
Trivium	633	308MHz	19	25609

validity of the DELUGE-0 results in as much as the subsequent modification to Grain v1 [145] slightly reduces the cipher's key agility as seen in Table 6.4. Table 6.4 also shows that the cost of key search machines based on the 80-bit keyed Trivium stream cipher would likely be double the cost. The effects of reducing key agility to improve system security are further discussed in Section 8.1.7.

6.7 Discussion

This chapter has presented a system for stream cipher key search that provides an initial assessment as to the value of 80-bit keys. The system is not suitable for practical implementation as the design targets the core aspects of key search and does not consider the full practicalities of an engine at system level.

Nevertheless, the key search machine described indicates that a single $15 chip is capable of searching 39 billion key/data possibilities every second. A move to a more memory based system or translation to a ASIC architecture mean that, from a data protection perspective, the concerning figures presented for 80-bit stream cipher keys may still be too optimistic. The next chapter investigates an improved key search architecture that is practical to implement and makes use of pipelining to indirectly reduce the comparison overheads.

6.7.1 Pseudocode

The following code describes the principle operations of the key search system outlined in Figure 6.5.

Part 1 - FRONT-END SEARCH

```
REPEAT UNTIL Set
SET Done to 0
SET IV to value stored externally

FOR each value of Counter
  FOR each cipher
    SET Key to {system FixedKey, unique CipherIdentifier, Counter}
    CALL rekeyCipher with Key and IV
  ENDFOR
  REPEAT UNTIL cipher initialisation period complete
  FOR each cipher
    FOR each pair of stored partial plainstreams
      SET Comparison to 0
      FOR each stored partial plainstream
        IF keystream same as plainstream THEN
          SET Comparison to 1
      ENDFOR
      SET searchFlag for pair to Comparison
    ENDFOR
  ENDFOR
  IF any searchFlag non-zero THEN
    STORE all searchFlags and Counter value in Primary FIFO
  ENDIF
ENDFOR

SET Done to 1
```

Part 2 - TRANSLATION UNIT

```
REPEAT UNTIL Primary FIFO not empty
READ searchFlags and Counter value from Primary FIFO
WHILE any searchFlag non-zero
  TRANSLATE most significant non-zero searchFlag into
            CipherIdentifier and PlainstreamSelect
  STORE CipherIdentifier, PlainstreamSelect and Counter value
        in Secondary FIFO
  SET most significant non-zero searchFlag to 0
WEND
```

Part 3 - BACK-END FULL TEST

```
REPEAT UNTIL Secondary FIFO not empty
READ CipherIdentifier, PlainstreamSelect and Counter value from
     Secondary FIFO
SET testKey to {system FixedKey, CipherIdentifier, Counter value}
CALL rekeyCipher with testKey and IV
REPEAT UNTIL cipher initialisation period complete
STORE first 80-bits of keystream
FOR each memory
   IF keystream same as value at address PlainstreamSelect THEN
     SET memFlag for memory to 1
   ELSE
     SET memFlag for memory to 0
ENDFOR
IF any memFlag is non-zero THEN
   SET Bus Output to testKey
   REPEAT UNTIL Acknowledge
ENDIF
```

Chapter 7

DELUGE - a practical stream cipher key search engine

THIS chapter introduces DELUGE, a second generation FPGA-based key search system that specifically targets stream ciphers. The design originally presented at SASC 2007 [2] should be considered the first modern public design that practically implements stream cipher brute force; its hardware code can be found in Appendix C.

The design itself offers much greater levels of efficiency than seen in the more theoretical engine of the previous chapter, and its highly modular and adjustable architecture lends itself to large scale multi-chip implementations. Although only a small demonstrator system was constructed, this was sufficient to confirm operation and allow extrapolation of simulation results to provide an economic assessment of the 80-bit key length. The resulting implications of this assessment will be addressed in Chapter 8.

7.1 Improving Key Search

Although the original key search engine described in the Chapter 6 was conceptually useful in resolving the methodology of implementing stream cipher brute force, its usefulness in making a reliable assessment of the security of key length was more limited. A number of shortcomings in the system were apparent but two principally stand out: that the failure to target structural aspects of a specific hardware platform underestimates the engine's performance potential; and that the simulation-led design

process used failed to address many system level design aspects crucial to large scale multi-chip set-ups. The creation of a second generation system, DELUGE,[1] allows these factors to be addressed and the potential of exhaustive search on synchronous stream ciphers to be fully realised.

7.1.1 Search Unit Architecture

The new search unit architecture of DELUGE still owes much to the first generation system and, as such, it targets 80-bit keys, uses the enhanced form of exhaustive search and operates towards the same scenario outlined in Section 6.3. The main search computation is performed by the search core of Figure 7.1a and this retains a buffered three tiered structure which avoids the inefficiency of performing full 80-bit comparisons for every key trial. The first stage uses a single cycle key agility stream cipher and short 16-bit partial comparisons to enable rapid elimination of the vast majority of keys, whilst the third stage, the 80-bit keystream compare unit, performs a full comparison for reliable key detection. The lack of pressure for high performance in the latter stage again allows a compact cipher implementation and serial comparison structure to be used, with the expander encoder that forms the mid stage of the design, buffering and converting parallel comparison results into the serial key and data information required.

7.1.2 Keystream Generation

One of the main areas ripe for improvement was that of keystream generation with the previous machine using just a basic implementation of Grain v0 to generate keystreams for comparison purposes. From a security stand point the Berbain, Gilbert, Maximov attack [144] meant it was chosen not to continue targeting this cipher, but instead make the almost equally efficient Grain v1 [145] the focus of search attempts.

Improvements in search speed were achieved through pipelining the stream cipher initialisation process. As Table 7.1 illustrates, this only provides small improvements in the efficiency of Grain v1 and, in the case of Trivium, none at all due to the longer initialisation procedure. Nevertheless, the consequential improvement in comparison

[1] Data Enhanced Logical Unlock-key Generating Engine. The name also reflects the machine's inundation of a target with keystreams and maintains the water based connotations of the phrase 'stream cipher'.

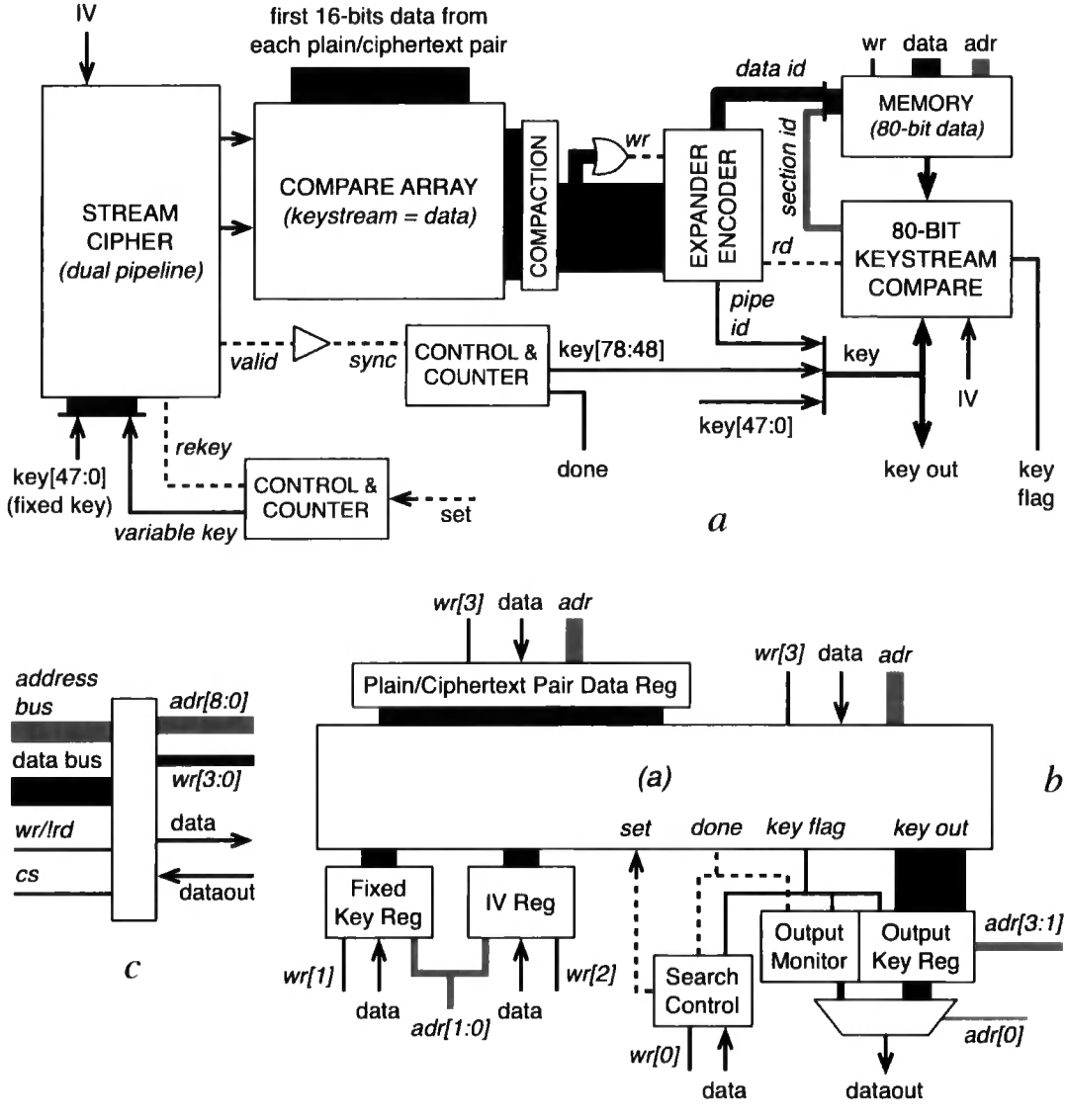


Figure 7.1: Search unit system architecture of DELUGE

a Key search core with dual keystream pipeline
b Internal search unit layout
c Bus interface of search unit

Table 7.1: Performance of Keystream Generation Schemes on Altera EP1C20

Keystream Generator	Acceleration	Logic, LE	Cycles/Key	Keys/LE.s
Grain v0	-	498	11	42k
Grain v1	-	543	11	35k
Grain v1 pipe.	pipelined	4665	1	46k
Trivium	-	633	19	26k
Trivium pipe.	pipelined	8507	1	26k
Grain v1 X2	x-factor	7647	0.5	53k
Grain v1 X4	x-factor	13554	0.25	58k

unit utilisation from the continuous feed of keystreams leads to real underlying benefit. As a consequence in each case pipelined versions are to be preferred.

7.1.3 Use of Computational Invariance in Pipelining

This technique specifically introduced for DELUGE's search core, modifies the pipelined initialisation process to take advantage of the fact that two nearly identical keys lead to similarities in the cipher state until the latter stages of initilisation. For Grain v1 (t=16) if two keys are selected so that only K_{79} differs, the calculation for the next state is identical and so the logic required for the second key may be saved. For the second stage a small calculation must be done but most of the state will be identical (Table 7.2) leading to similar logic savings. Using 5 stages of this invariance in pipes sharing key bits $K_{78..0}$ produced a 15% efficiency increase in search performance (Table 7.1). The remaining key bit, K_{79}, acts as the pipe identifier.

The technique may be extended to more pipelines and a 4 pipeline application (Grain v1 X4) with shared key bits $K_{77..0}$ produces a further 10% efficiency improvement. The invariances for this system are presented in Tables 7.3 and 7.4.

The computational invariance technique is also applicable to the ciphers Trivium and Grain-128.[2] The invariance and results for these are presented at the end of the Chapter in Section 7.4.2.

[2] These ciphers may also be used in the search machine as DELUGE itself is flexible as to the specifics of the search module so long as it produces at least one 16-bit keystream per clock cycle.

Table 7.2: Register Invariance in a Dual Grain v1 Pipeline

Stage	NFSR	LFSR
1	$b_{78..0}$	$s_{79..0}$
2	$b_{79..64}, b_{62..0}$	$s_{79..0}$
3	$b_{79..76}, b_{74..72}, b_{70..68}, b_{66}, b_{63..48}, b_{46..0}$	$s_{79..72}, s_{70..65}, s_{63..0}$
4	$b_{70}, b_{63..60}, b_{58..56}, b_{54..52}, b_{50}, b_{47..32}, b_{30..0}$	$s_{78}, s_{74}, s_{70..69}, s_{67}, s_{63..56}, s_{54..49}, s_{47..0}$
5	$b_{54}, b_{47..44}, b_{42..40}, b_{38..36}, b_{34}, b_{31..16}, b_{14..0}$	$s_{62}, s_{58}, s_{54..53}, s_{51}, s_{47..40}, s_{38..33}, s_{31..0}$
6	$b_{38}, b_{31..28}, b_{26..24}, b_{22..20}, b_{18}, b_{15..0}$	$s_{46}, s_{42}, s_{38..37}, s_{35}, s_{31..24}, s_{22..17}, s_{15..0}$
7	$b_{22}, b_{15..12}, b_{10..8}, b_{6..4}, b_2$	$s_{30}, s_{26}, s_{22..21}, s_{19}, s_{15..8}, s_{6..1}$
8	b_6	$s_{14}, s_{10}, s_{6..5}, s_3$
9	-	-
10	-	-

Table 7.3: 1st Order Invariance in Shift Registers of Grain v1: pipelines $[id = 01, 10, 11]$ relative to pipeline $[id = 00]$

(a) NFSR

Stage	Pipe $[id = 01]$	Pipe $[id = 10]$	Pipe $[id = 11]$
0	$b_{79,77:0}$	$b_{78:0}$	$b_{77:0}$
1	$b_{78:63,61:0}$	$b_{79:64,62:0}$	$b_{78:64,61:0}$
2	$b_{79:75,73:71,69:67,65,62:47,45:0}$	$b_{79:76,74:72,70:68,66,63:48,46:0}$	$b_{79:76,73:72,69:68,62:48,45:0}$
3	$b_{69,63:59,57:55,53:51,49,46:31,29:0}$	$b_{70,63:60,58:56,54:52,50,47:32,30:0}$	$b_{63:60,57:56,53:52,46:32,29:0}$
4	$b_{53,47:43,41:39,37:35,33,30:15,13:0}$	$b_{54,47:44,42:40,38:36,34,31:16,14:0}$	$b_{47:44,41:40,37:36,30:16,13:0}$
5	$b_{37,31:27,25:23,21:19,17,14:0}$	$b_{38,31:28,26:24,22:20,18,15:0}$	$b_{31:28,25:24,21:20,15:0}$
6	$b_{21,15:11,9:7,5:3,1}$	$b_{22,15:12,10:8,6:4,2}$	$b_{15:12,9:8,5:4}$
7	b_5	b_6	-
8	-	-	-
9	-	-	-

(b) LFSR

Stage	Pipe $id = 01$	Pipe $id = 10$	Pipe $id = 11$
0	$s_{79:0}$	$s_{79:0}$	$s_{79:0}$
1	$s_{78:0}$	$s_{79:0}$	$s_{78:0}$
2	$s_{79:71,69:64,62:0}$	$s_{79:72,70:65,63:0}$	$s_{79:72,69:65,62:0}$
3	$s_{77,73,69,68,66,63:55,53:48,46:0}$	$s_{78,74,70,69,67,63:56,54:49,47:0}$	$s_{69,63:56,53:49,46:0}$
4	$s_{61,57,53,52,50,47:39,37:32,30:0}$	$s_{62,58,54,53,51,47:40,38:33,31:0}$	$s_{53,47:40,37:33,30:0}$
5	$s_{45,41,37,36,34,31:23,21:16,14:0}$	$s_{46,42,38,37,35,31:24,22:17,15:0}$	$s_{37,31:24,21:17,14:0}$
6	$s_{29,25,21,20,18,15:7,5:0}$	$s_{30,26,22,21,19,15:8,6:1}$	$s_{21,19,15:8,5:1}$
7	$s_{13,9,5,4,2}$	$s_{14,10,6,5,3}$	s_3
8	-	-	-
9	-	-	-

Table 7.4: 2nd Order Invariance in Grain v1: pipeline $[id = 11]$ relative to $[id = 01, 10]$

(a) NFSR

Stage	Relative to $[id = 01]$	Relative to $[id = 10]$
0	-	-
1	b_{79}	-
2	$b_{74,70,66,63}$	$b_{75,71,67,65}$
3	$b_{70,58,54,50,47}$	$b_{69,59,55,51,49}$
4	$b_{54,42,38,34,31}$	$b_{53,43,39,35,33}$
5	$b_{38,26,22,18,15}$	$b_{37,27,23,19,17}$
6	$b_{22,10,6,2}$	$b_{21,11,7,3,1}$
7	b_6	b_5
8	-	-
9	-	-

(b) LFSR

Stage	Relative to $[id = 01]$	Relative to $[id = 10]$
0	-	-
1	s_{79}	-
2	$s_{70,63}$	$s_{71,64}$
3	$s_{78,74,70,67,54,47}$	$s_{77,73,68,66,55,48}$
4	$s_{62,58,54,51,38,31}$	$s_{61,57,52,50,39,32}$
5	$s_{46,42,38,35,22,15}$	$s_{45,41,36,34,23,16}$
6	$s_{30,26,22,19,6,}$	$s_{29,25,20,18,7,0}$
7	$s_{14,10,6,3}$	$s_{13,9,4,2}$
8	-	-
9	-	-

7.2 A Practical System

Although the large bus widths of [1] may be suitable for a single chip system the need for a large scale multi-chip design when undertaking 80-bit key space search renders such an interface impractical. DELUGE comprehensively resolves this issue with a redesigned interface structure (Figure 7.1c) based on a more conventional data and address bus architecture that would additionally be suitable for implementation in generic computational machines such as to COPACOBANA (see Section 6.2.3). The simplified interface greatly eases the set-up process, and the introduction of separated clock domains means that the search speed is no longer at the mercy of interface requirements.

DELUGE's output register (Figure 7.1b) can be polled via the data bus to check if the allocated key space has been searched or if, indeed, a possible key has been found. In this way a single computer may monitor and manage a large number of DELUGE search units (Figure 7.2). The only other requirement for DELUGE's correct multi-chip operation being that of the chip-select signal address decoders — all other interface signals are global to the system. The full outline of operating instructions implemented by the DELUGE interface can be found in Appendix B.

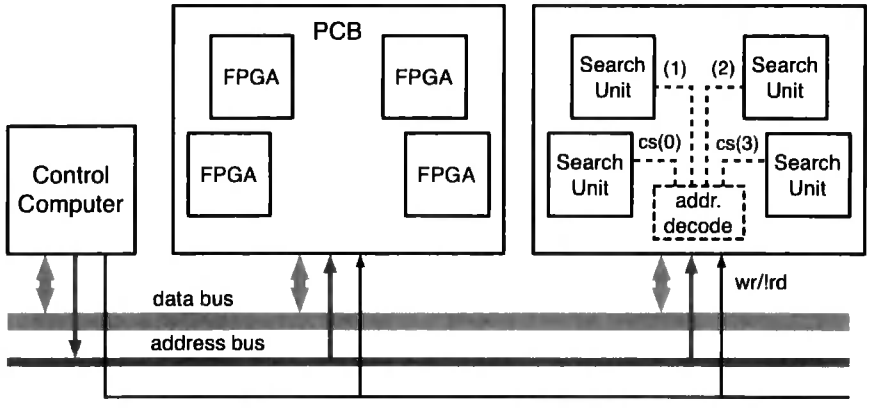


Figure 7.2: Inter-unit buses and possible chip layout for DELUGE

7.2.1 Practical Demonstrator

To be confident the system was practical in real world operation a demonstrator system was conceived (Figure 7.3). This placed a DELUGE chip module with a Grain v1 search unit on a small Altera 'Stratix' EP1S10 FPGA and the associated controller on a 'APEX' 20K200EFC FPGA with a 7-segment display feeding information to the user. The system checked 2 keystreams against 64 plainstream every cycle, covering $2^{28} - 1$ of key space every 11 seconds while flagging key matches appropriately. The 10MHz bus speed and 50MHz search speed potentially could be taken much higher but the demonstrator still was sufficient to show the feasibility of large scale DELUGE based search engines.

7.3 System Results

The chosen Altera Cyclone II target platform was again, at the time of design, one of the most economic FPGA offerings available with the EP2C35 having a 250k unit volume pricing of $22 per chip [206]. This chip could either fit a DELUGE system with 2 keystreams and 2^9 plainstreams or 4 keystreams and 2^8 plainstreams,[3] the simulation results of which are given in Table 7.5. For comparison purposes the results presented, also include the pure brute force system later implemented through DELUGE by setting its data element to '1'.

The actual full system cost of DELUGE, including construction components,

[3] Despite the two keystream set-up producing a slightly lower cost system the four keystream system may be preferred in reality due to its lower data requirement.

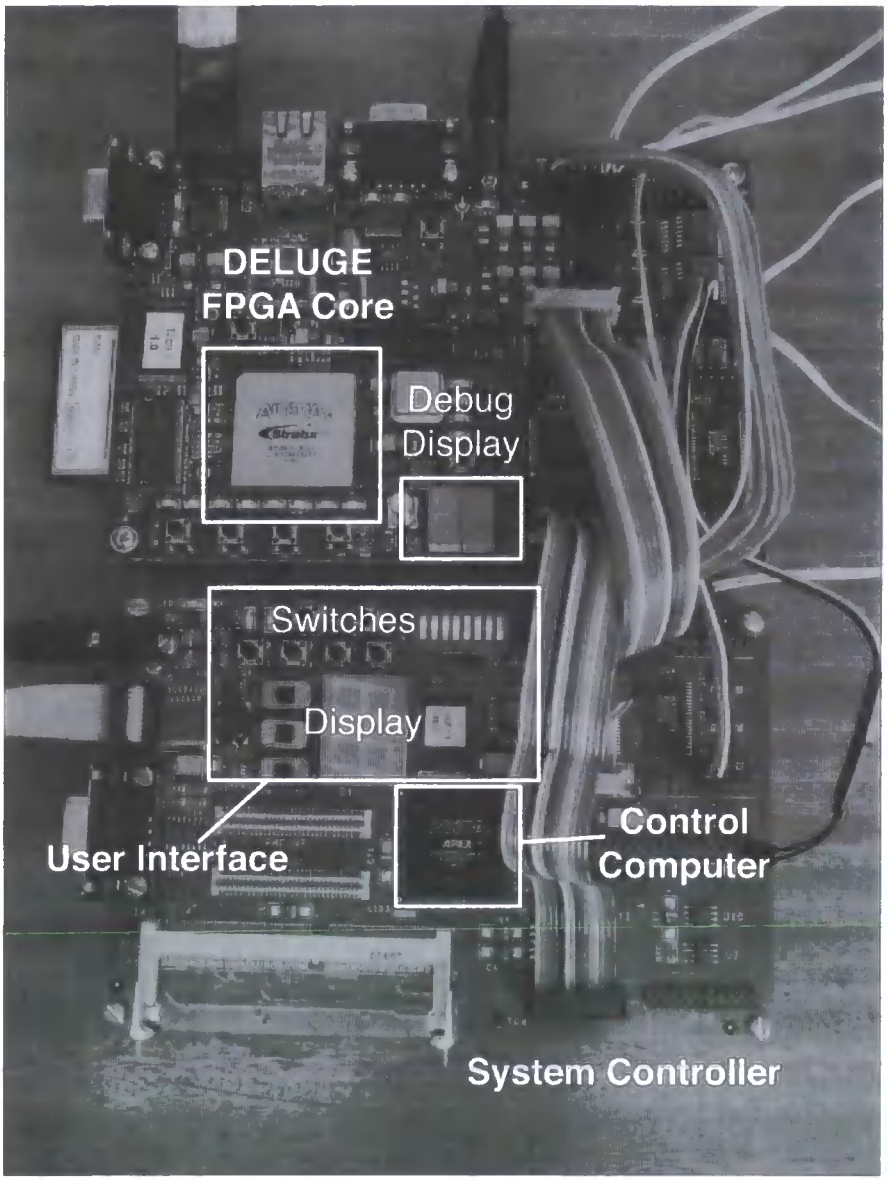


Figure 7.3: Implemented demonstrator of DELUGE key search

Table 7.5: Chip Cost of 80-bit Key Exhaustive Search using DELUGE on a EP2C35F672C8 FPGA

Core Design	Core Logic, LE	Searches/s	Recovery	Cores	Chip Cost
n=2×4X, p=1	26.4k	1.0×10^9	1wk	970m	$21.3b
n=4X, p=256	29.2k	138×10^9	1wk	10.0m	$221m
n=2X, p=512	25.5k	144×10^9	1wk	9.6m	$211m
n=2×4X, p=1	26.4k	1.0×10^9	1yr	18.6m	$409m
n=4X, p=256	29.2k	138×10^9	1yr	193k	$4.2m
n=2X, p=512	25.5k	144×10^9	1yr	184k	$4.1m

should be considered to be double the Table 7.5 figures — assuming that COPACOBANA is reflective of key search system costing.

7.4 Discussion

The performance of the DELUGE system is 40 times that described for the FPGA platform in [1] and half the cost of the described structured-ASIC machine.[4] The threat levels originally outlined in 1996 [175] indicate that these figures place DELUGE in the realms of feasibility for a large organisation and indeed with U.S. inflation over the intervening period at 35% [208, 209] the $10m level quoted may now underestimate the resources available to an attacker. Nonetheless, in making feasibility predictions the size of this inflation increase is trivial in comparison to a Moore's Law progression over the same period which resulted in an estimated 1600% increase in computing resource per dollar.

Assuming technology trends continue to follow Moore's law, it is apparent that future process nodes will bring with them the possibility of a sub-million dollar search system. As we see in Section 7.4.1 this fundamental assumption need not continue to hold but even at present it is suggested that DELUGE represents a practical threat to stream ciphers with short 80-bit key lengths.

7.4.1 Improving FPGA Technology

As outlined in Section 6.4 FPGA technology is an area of rapid development and following the presentation of DELUGE at SASC'07, Altera made an announcement in March 2007 on the release of the 'Cyclone III' FPGA [210]. This new iteration of the Cyclone family, based on a 65nm technology, may have been expected to continue down the path set by Moore, however the results in Table 7.6 indicate that this is not the case — the intervening 18-months between Cyclone II and Cyclone III leads to less than a 10% fall in costs.

Some of this shortfall can be explained by the advance status of these chips at the time of testing but much of the reason lies in a change of Altera's design

[4] It is attractive to think of moving DELUGE to an ASIC platform, bringing further cost reductions to short recovery time searches. However, as mentioned in Section 6.4, unlike FPGAs the technology forces the creation of one-offs, making it arguably less attractive to an adversary considering investment in computing resource.

philosophy — from one that is speed and cost focused to one that is orientated towards power efficiency. Indeed the large deviation from Moore on cost-performance may in part be offset by the 50% power reduction [211] of these devices which will lower the running costs of systems such as DELUGE. Nonetheless, the assumption of a continued Moore's Law extrapolation for low cost FPGAs would seem likely to be unduly pessimistic when calculating the future cost to an adversary.

Table 7.6: DELUGE Chip Cost on EP2C35F672C8 and EP3C40F484C8 FPGAs

FPGA	Core Design	Core Logic, LE	Searches/s	Recovery	Chip Cost
Cyclone II	n=4X, p=256	29.2k	138×10^9	1yr	\$4.2m
	n=2X, p=512	25.5k	144×10^9	1yr	\$4.1m
Cyclone III	n=4X, p=256	28.6k	148×10^9	1yr	\$3.9m
	n=2X, p=512	23.2k	150×10^9	1yr	\$3.9m

7.4.2 Key Search of Other Stream Ciphers

DELUGE was designed to accommodate the interchange of the cipher under test and so it is useful to consider the search machine's operating performance in the presence of stream ciphers other than Grain v1. The performance of the keystream generator with Trivium has already be observed in Table 7.1 but with a small level of modification DELUGE may also operate with a 128-bit key cipher such as Grain-128. The keystream generator module results for this stream cipher are presented in Table 7.7.[5]

The compacted dual pipeline of Grain-128 X2, perhaps surprisingly, illustrates that there is very little difference in performance terms between it and its 80-bit keyed cousin. Grain-128's dual pipeline would appear to compact with even greater efficiency than that of Grain v1, leaving it using less resources despite the larger register sizes. The added variables used in the feedback and output functions do reduce the achievable speed relative to Grain v1 but it remains fast enough that the difference would be largely negated by other routing concerns in the full key search structure. In view of this, it is thought that Grain-128 may replace Grain v1 in DELUGE without significant performance penalty except to accommodate searching

[5] It may noted that the figures quoted for the Grain v1 compacted dual pipeline are slightly improved in resource efficiency terms compared to those in Table 7.1. This is due to the use of a fully compacted pipeline instead of the partial 5-stage one seen in the SASC'07 DELUGE system. As is observed, the difference is minimal and again does not produce large reductions in key search machine costs which may affect previous findings.

Table 7.7: Updated Performance of Grain v1 and Grain-128 Keystream Generation Schemes on Altera EP1C20

Keystream Generator	Acceleration	Logic, LE	Cycles/Key	Keys/LE.s
Grain v1	-	543	11	35k
Grain-128	-	776	9	24k
Grain v1 pipe.	pipelined	4665	1	46k
Grain-128 pipe.	pipelined	4989	1	37k
Grain v1 X2	x-factor	7600	0.5	54k
Grain-128 X2	x-factor	7516	0.5	49k

the larger key space.

The table for implementing the invariance in the Grain-128 dual pipeline design is given in Table 7.8. A similar table for Trivium (Table 7.9) illustrates that large parts of Trivium's registers also display invariance to a single bit difference in the key, although with its greater set-up time the benefit will be proportionality less pronounced than that seen with the Grain ciphers.

Table 7.8: Invariance in the shift registers of Grain-128: pipeline $[id = 1]$ relative to pipeline $[id = 0]$

(a) NFSR

Stage	Invariant Registers
0	$b_{126:0}$
1	$b_{126:96,94:0}$
2	$b_{125,122:119,117:108,106:103,101,99:97,94:64,62:0}$
3	$b_{122,120,117:115,110,106,104,93,90:87,85:76,74:71,69,67:65,62:32,30:0}$
4	$b_{90,88,85:83,78,74,72,61,58:55,53:44,42:39,37,35:33,30:0}$
5	$b_{58,56,53:51,46,42,40,29,26:23,21:12,10:7,5,3:1}$
6	$b_{26,24,21:19,14,10,8}$
7	-

(b) LFSR

Stage	Invariant Registers
0	$s_{127:0}$
1	$s_{127:0}$
2	$s_{126:119,117:103,101:97,95:0}$
3	$s_{126,120,116:115,112,110,106,104,100,98,94:87,85:71,69:65,63:0}$
4	$s_{94,88,84:83,80,78,74,72,68,66,62:55,53:39,37:33,31:0}$
5	$s_{62,56,52:51,48,46,42,40,36,34,30:23,21:7,5:1}$
6	$s_{30,24,20:19,16,14,10,8,4,2}$
7	-

Table 7.9: Invariance in the shift registers of Trivium: pipeline $[id = 1]$ relative to pipeline $[id = 0]$

(a) Shift Register F1

Stage	Invariant Registers
0	$s_{2:93}$
1	$s_{1:64,66:93}$
2	$s_{1:59,61:93}$
3	$s_{1:54,56:93}$
4	$s_{1:10,19:28,32:40,44:49,51:55,57:93}$
5	$s_{5,14,21:23,30:32,40:41,43:44,46:47,53:59,65:74,83:92}$
6	$s_{9,17:18,42,69,78,85:87}$
7	$s_{37,46,55,64,73,81:82}$
8	s_{50}
9-17	-

(b) Shift Register G2

Stage	Invariant Registers
0	$s_{94:177}$
1	$s_{94:177}$
2	$s_{94:128,132:155,157:177}$
3	$s_{94:114,118:123,127:141,143:150,152:177}$
4	$s_{94:100,104:109,113:118,122:127,129:136,138:145,147:177}$
5	$s_{99:101,110:113,117:119,124:131,135:140,142:146,148:164,168:173,177}$
6	$s_{96,121:123,131:132,134:135,137:138,149:150,156,163:165,174:177}$
7	$s_{151,160}$
8	-
9-17	-

(c) Shift Register H3

Stage	Invariant Registers
0	$s_{178:288}$
1	$s_{178:288}$
2	$s_{178:288}$
3	$s_{178:192,198:207,211:219,223:234,236:288}$
4	$s_{178,184,193,200:202,209:211,213:214,118:220,222:229,231:256,262:271,}$ $_{275:283,287:288}$
5	$s_{179,188,195:197,204:206,208:209,213:215,217:224,226:233,239:242,248,257,}$ $_{264:266,273:275,277:278,282:284,286:288}$
6	$s_{190:192,203:204,208:210,216:219,225,234,243,252,259:261,268:270,272:273,}$ $_{277:279,281:288}$
7	$s_{229,254:256,267:268,272:274,280:283}$
8	-
9-17	-

Chapter 8

Discussion

THE economically feasible implementation of key search shown by DELUGE is in dramatic contrast to other attack techniques, and lends weight to the argument that brute-force attacks are underestimated in the security evaluation of cipher designs. Moreover, with exhaustive search substantially cheaper than state guessing it is suggested that the practice of designing the state to be twice key length is excessive and that by lengthening the key, security may be improved.

The final sections return to consider the key search findings from the context of implementing secure display. The suitability of the studied stream cipher proposals are reviewed, with aspects of security and performance discussed before a conclusion as to their appropriateness for the system's decryption sub-block is made.

8.1 Implications for Stream Cipher Design

The high performance of the DELUGE key search engine indicates that a 80-bit stream cipher key length is too short to maintain full security and raises some important questions over the design philosophies behind stream ciphers. In this section we will look more broadly at these issues.

8.1.1 Algorithmic Attacks

The form of attack most dangerous to any cipher is one based on a unanticipated weakness in the way an algorithm functions. These attacks have the potential not just to lead to minor compromises in security but may instead devastatingly break the cipher with an attack that is practical on discovery. In the case of stream ciphers correlation between input and output, resynchronisation, period and results of algebraic analysis are among a number of factors now relevant to their design and it is likely that as understanding matures this list will grow. The realisation of new forms and combinations of algorithm based attacks may be inevitable but their applicability to a specific stream cipher will remain something of a mathematical lottery.

8.1.2 Assessing Complexity

An attack of complexity 2^{79} would typically be considered as breaking an 80-bit key stream cipher with intended security 2^{80}. However, as alluded to by Bernstein [176], complexity is often a rather vague term when used in this context as although ideally it should relate to exhaustive search it is often impractical to do so making a true comparison difficult: attacks may have a large memory element or simply their complexity may not scale linearly. Yet, an attempt must be made as defining the true complexity of the differing attack types targeted at stream ciphers is crucial for the formation of a balanced security picture.

8.1.3 Grain Case Study - evaluation of attack complexity

Conveniently the Berbain, Gilbert, Maximov paper [144] provides an opportunity to assess attack complexity as the attack on the original version of Grain allows direct comparisons to be extrapolated from the estimated costs presented in [1]. Such analysis, first presented in [2], has, in the following section, been revised to reflect the results of the DELUGE key search engine.

Berbain's first attack uses linear approximation to obtain a supposed complexity 2^{55} attack against Grain. The impracticalities of cheaply computing the attack mean a reduced version of the cipher becomes the focus, with the same techniques producing a quoted complexity of 2^{35} and taking around an hour on an 2.5GHz Intel Xeon

workstation. Expanding this to the level of a 2^{79} complexity attack, which should reasonably be equivalent to 80-bit key exhaustive search, would require 2 billion machine years to compute and assuming, possibly optimistically, that Xeon chip cost ($198 per chip [212]) represents the entire cost of a system this would result in a total cost of $398 billion. In reality this figure should be equivalent to the $818 million of a non-enhanced exhaustive search system[1] (see Table 7.5) and a conclusion can be made that the original attack complexity may be understated by around an order of 9, making its real complexity in the region of 2^{64}. Berbain's improved second attack has a described complexity of 2^{43} and so using same methodology the figure 2^{51} can be derived for the real complexity of this attack. Both values are significantly higher than those quoted and the large memory requirements of each attack, 2^{49} and 2^{42}, should raise these figures higher still, while effectively eliminating any savings possible through translation into hardware.

The values for real complexity indicate that in the production of Grain v1, when the original Grain was corrected to provide at least 2^{80} security, the security level against algorithmic attacks may in fact have been raised to beyond 2^{88}. This is in direct contrast to security against enhanced exhaustive search attacks where Grain v1 like its predecessor remains at a level no better than 2^{73} and so a key length extension of 15-bits or more may be necessary to counteract the imbalance.

8.1.4 State Guessing

Another method for attacking stream ciphers, state guessing, involves searching for the internal state of the cipher instead of the key. As the internal state stores all of a stream cipher's entropy it is typically larger in size than the key, therefore rendering exhaustive state search less efficient than one of key-space unless a cipher's initialisation complexity is high. To overcome the extra burden, state guessing uses large amounts of data and memory to reduce computation requirements in a trade-off based arrangement [213, 33] similar to that of enhanced exhaustive search. The balance of key length, initialisation procedure and state size determining which of these two universal attacks proves more efficient against a particular cipher.

[1] It is assumed reasonably that the performance of a Grain v0 system would be comparable to the brute force Grain v1 system and that the full system would be double chip value.

8.1.5 Grain Case Study - consequences of key extension on state guessing

Looking at a 15-bit key length extension from the context of state guessing attacks, an 80-bit key, generic stream cipher can be imagined that has an initialisation period of 2^5 clock cycles and a state size of 160-bits. Thus in extending the key to 95-bits a state guessing attack with equivalent computational cost to brute-force would be capable of producing around 2^{99} keystream sections[2] suitable for comparison purposes. These would, on average, need to be compared to 2^{60} bits of collected plainstream data to achieve a successful outcome, leaving a data and memory requirement still well in the petabyte region.

Moreover, while the headline figure for memory seems plausible for a hard-disk set-up, this would simply ignore the very real time constraints on each search and compare operation. The output from each guess must be compared to all 2^{60} different values and if an impractical 2^{159} comparison operations are not to be required the output must be directly used to address memory locations with these locations returning a 'logic 1' if their position exists in the plainstream data block. Unfortunately this leads to an impracticably high-false positive rate unless the memory address space is comparable to the state size, something which would in itself lead to memory requirements becoming impractical.

By making the locations return the next part of the plainstream data block and a pointer to data location the false-positive rate can be eliminated by a subsequent round of comparison operations, with a memory cost per location of under 2^5 bytes. For the estimated $266 billion cost[3] of a DELUGE enhanced search system capable of targeting 95-bit key lengths around a zettabyte of storage could be bought, assuming a figure of $100 per half terabyte of hard-disk space. This amount of memory would provide 2^{65} storage locations, leaving a still comparatively feasible number of comparisons at 2^{94}. Nonetheless, memory access on this scale is relatively slow process with hard-disk access times typically in the millisecond range and therefore with 2^{99} memory accesses still required and only 31 billion milliseconds in a year, sextillions of separate disks would be needed to mount a successful attack within such a time-scale. Storage cost means access requirements cannot be sufficiently reduced to alleviate the issues of access time and increasing the plainstream data collected

[2] The specifics of system design will affect this figure: need for a high rate of guess elimination will tend to favour exhaustive search efficiency; use of windowing on the state guessing system's keystream output will tend to increase section generation.

[3] Calculated from Table 7.5.

to fill all storage locations only reduces accesses by a factor 32 while introducing significant collection issues due to data overlap. Therefore in the example outlined state guessing remains impractical despite the increase in the ratio of key length to state size above the conventional 1 to 2 design limit originally proposed by Babbage [213]. The gap in economic equality of the two attacks is such that exhaustive search would appear to be the substantive concern for the foreseeable future even when consideration of advances in low cost storage technologies are made.

Translating the state guessing attack to a Biryukov style attack [214] through use of the Hellman time-memory trade-off [181] does not alter this conclusion as while it potentially reduces data collection it also introduces a large precomputational requirement comparable to that of exhaustively searching the entire 160-bit state space and so any such attack would again be impractical.

8.1.6 Key Space Based Time-Memory Trade-off

If, as described by Hong [204], the Hellman time-memory trade-off is applied to search the stream cipher key space rather than the enlarged state, a much more effective attack can be realised despite the need for cipher initialisation calculations. Both online computational costs and memory requirements would see feasibility due the reduced search space, while the precomputation cost, that provides a complexity threshold to the attack's practicability, would also be reduced.

In a theoretical sense this precomputation should be of identical cost to that of an enhanced brute-force key space attack with a key length extension equally complicating both attack methods. However, the similarity in duration of the 1 week precomputation described in [215] for a time-memory trade-off attack on 40-bit DES and that taken for COPACOBANA [190] to search a full 56-bits of DES key space using exhaustive search possibly suggests otherwise. The reality is that the complexity of precomputation will generally outweigh that of enhanced brute-force due to a corresponding need for point storage and the computational cost incurred by search reduction techniques such as distinguishing points. Consequently when data availability is low, enhanced brute-force will represent the lowest entry level attack.

8.1.7 Alternatives to Key Length Extension

Key extension is not the only way to increase brute-force resistance, and one alternative is to simply increase the required length of the set-up protocol. Each doubling in length will equate to increasing the key length by one bit as extra effort must be expended in terms of hardware resource or processing time.

Although effective, lengthening the set-up protocol is not an attractive path to follow as initialisation time directly affects a cipher's performance: in the case of Grain v1, if set-up was extended to provide an equivalent of an extra 10-bits of security the set-up time would rise from 10-cycles to over 10000-cycles — a level that is likely to be impractical in many applications. Other methods that increase the cipher's implementation complexity to raise attack resistance share similar performance caveats.

8.2 Modifying Key Lengths

An inevitable consequence of enhanced brute-force attacks is that stream ciphers must fundamentally lose the direct relationship that presently exists between key length and security. This is of little consequence to software stream ciphers, with their long keys, as even an enhanced brute-force attack will simply be impractical. However for hardware focused ciphers with short 80-bit keys, such as Trivium and Grain v1, it is of great consequence, with enhanced brute-force representing a practical threat. Unless these algorithms are to follow a path of massively increased initialisation set-ups and a redesign to increase implementation complexity — something which contradicts the performance aims that underpin this class of cipher — key length must be raised. The modifications described in the final sections are one method of achieving this.

8.2.1 Grain

Grain v1's state size of 160-bits is right on the conventional limit to avoid state guessing attacks so care has to be taken in the level of key length extension used. In light of the analysis in Section 8.1.5 it is proposed that key length should be extended by 22-bits to a total of 102-bits. This leaves the state-guessing attack possibly preferential to enhanced brute-force but still in the realm of impracticality medium-

term. As such, algorithmic attacks will be re-established as the main theoretical concern.

Due to the comparatively small state size, implementing the proposed extra key bits must necessarily have an impact on the available IV range of Grain v1 as only 15-bits of the initial state serve no function and hence a compromise must be made. With often predictable values and open transmission the role of IVs in entropy provision as described by Hong [204] is negligible so their implementation should be regarded primarily as a convenience to the user. Use of a 42-bit IV provides over 4 trillion possible IVs and should be sufficient for almost all applications, so the proposed modification involves placing the extra key material in 22 bit locations where IV material would have previously resided:

$$(b_{79}, ..., b_0) \leftarrow (K_{79}, ..., K_0)$$
$$(s_{79}, ..., s_0) \leftarrow (1, ..., 1, K_{101}, ..., K_{80}, IV_{41}, ..., IV_0)$$

By swapping IV state bits directly for key material the expectation is that the algorithmic security of Grain v1 will be left unaltered by the change but it may be more prudent to simply migrate designs to its 128-bit keyed cousin, Grain-128 [146], which with its similar structure maintains the family's size and speed advantages. Crucially the introduction of the 128-bit key dispels the possibility of practical generic attacks in all likely applications and leaves system designers to only concern themselves with its maintenance of algorithmic strength.

8.2.2 Trivium

Trivium's state size of 288-bits is much larger than that of Grain v1 and so state guessing attacks are not an issue of concern when considering key length extensions. As a consequence a 32-bit key length extension was proposed in [2] which would in reality remove enhanced brute-force from the security equation. However the state recovery attack outlined by Maximov [134] has complexity similar to that of guessing the state of Grain v1 and in light of this it is proposed that the key should only be lengthened by 22-bits to a total of 102-bits. These extra bits need not impact on the possible IV size and it is proposed that they are loaded into the lower 22-bits of the register containing state bits $s_{178}, ..., s_{288}$:

$$(s_1, ..., s_{93}) \leftarrow (K_0, ..., K_{79}, 0, ..., 0)$$

$$(s_{94}, ..., s_{177}) \leftarrow (IV_0, ..., IV_{79}, 0, ..., 0)$$
$$(s_{178}, ..., s_{288}) \leftarrow (K_{80}, ..., K_{101}, 0, ..., 0, 1, 1, 1)$$

The 22-bit key extension makes current brute-force attacks impractical and should leave algorithmic attacks as the non-invasive method most likely to compromise the cipher. Indeed, many of these attacks would be made more complex by the additional key material, so although the medium term security of Trivium would clearly not be guaranteed, the algorithm could avoid the spectre of a practical attack.

8.2.3 Further Modifications

Although key length modification is proposed in the preceding paragraphs, in the implementation of stream cipher hardware modules it may be prudent to include a facility to make the set-up length selectable, as this would provide a facility to partially mitigate against future cryptanalysis advances: extending the set-up time by a multiple of 32 would potentially make attacks 32-times as costly. By invoking this flexibility the performance hit need only be taken as and when circumstances dictate.

It is also worth noting that algorithmic attacks often require high levels of data produced under a single key or IV. As such it may be prudent to implement a flexible and conservative operation protocol since changing the key and IV more regularly potentially stops an adversary observing the data quantity required for a successful attack.

8.3 Stream Cipher use in Secure Display

The hardware performance figures presented in Table 4.2 and the keystream generation performance of the stream cipher module within DELUGE, indicate that stream ciphers are capable of achieving the data-rates and efficiency required for decryption in secure display systems. However, DELUGE and future trends in FPGA cost-performance suggest that an 80-bit key is insufficient for secure, stream cipher based symmetric encryption, and places undesirable constraints on secure display system use.

The cost of using larger keys, such as moving to a 128-bit key length, need not significantly impact on decoding performance and hardware efficiency, but would significantly increase search complexity and hence attack cost, making the long term viability of stream cipher proposals much more attractive. The eStream cipher, Grain-128, gives essentially equivalent hardware performance (see Table 4.2) to the highly efficient 80-bit key ciphers, Grain v1 and Trivium, with its 128-bit key size and increased security making it a promising alternative to the cipher modifications proposed in this chapter. In view of this, Grain-128 is suggested as best option for the decryption sub-block, in a secure display system. Selection of an unmodified 80-bit keyed eStream proposal is thought imprudent at this stage.

8.4 Issues Yet to be Addressed

Finally, presented below are a number of unresolved issues surrounding secure display and stream cipher key search that may prove to be interesting areas for further investigation:

- **Chip-on-glass technology:** The less restricted environment of a bonded silicon chip potentially resolves many of the performance concerns surrounding implementing secure display, making it a good option for initial secure display designs.
- **Tamper resistant display circuits:** The use of tamper resistance in data security systems is widespread. However, while knowledge of the techniques are well resolved for silicon based applications such as smart cards, it is not known at present how to achieve these on glass.
- **ASIC key search performance:** ASIC chips offer potentially significant performance advantages to FPGA implementation. However, the size of these advantages for key search machines such as DELUGE is unresolved and may potentially affect the economic value of stream cipher key-sizes.
- **3D chip technologies:** these have the potential to create a step change in the performance of the DELUGE key search engine by resolving routing complexities through the use of interconnected multilayer chips. This could potentially have a substantial bearing on the economic value of key length.

Chapter 9

Conclusion

The thesis has examined the concept of secure display with particular reference to the usefulness that stream ciphers may play. The security benefits of protecting the display link right to glass substrate are plausible, especially in the light of Kuhn's recent work [17] that has shown flat-panel displays to be vulnerable to being read remotely. However, the challenges of designing a secure display are substantial, with the high data-rate requirements of video transmission conflicting with the restrictions of glass substrates as well as the needs of power and area efficiency in mobile devices.

The efficiency in hardware of the newly proposed stream ciphers, Trivium and Grain v1, present realistic options for meeting these requirements and both have impressive performance characteristics well suited to restricted hardware. Their use of 80-bit keys is, however, a concern with this key size in block ciphers considered to be right on the very edge of what is required for an algorithm's security.

The hypothesis that these stream ciphers may not be suitable for secure display purely from the security context surrounding their key length, has been addressed through the design of an original key search machine, DELUGE. This machine indicates that although brute force key search does not present a realistic attack vector, a data enhanced attack does, and even today places an economic value on the 80-bit key size at around £4 million. The advance of processing power means this valuation is set to rapidly fall with time and would significantly discourage stream cipher use in long-term and high value applications of data security. For secure display systems, it is therefore suggested that a larger key size may be preferred and suitable alterations to the ciphers Trivium and Grain v1 have been proposed. The 128-bit keyed stream cipher, Grain-128, offers very similar hardware performance to

both Trivium and Grain v1, and it is therefore suggested as the favoured option for implementing decryption in a secure display system.

Bibliography

[1] Iain Devlin and Alan Purvis. A fundamental evaluation of 80-bit keys employed by hardware oriented stream ciphers. In *SHARCS '06*, Special-purpose Hardware for Attacking Cryptographic Systems, Cologne, Germany, April 2006.

[2] Iain Devlin and Alan Purvis. Assessing the security of key length. In *SASC 2007*, The State of the Art of Stream Ciphers, Bochum, Germany, January 2007.

[3] Harry Walton. The System LCD – a new paradigm for display industry. In *EPPIC Display Technology - The Big Picture*, Cambridge, UK, March 2003.

[4] Hirohide Nakagawa. Current trends of flat panel displays viewed from applications. *Sharp Technical Journal*, -(4), April 2003.

[5] Graham Cairns, Catherine Dachs, Mike Brownlow, Yasushi Kubota, Hajime Washio, and Masaya Hijikigawa. Multi-format digital display with content driven display. In *Society for Information Display 2001, Digest of Technical Papers*, volume 32, pages 102–277, San Jose, USA, June 2001.

[6] Bu-Yeol Lee, Yasushi Kubota, and Shigeki Imai. CPU on a glass substrate using CG Silicon TFT. *Sharp Technical Journal*, -(4), April 2003.

[7] Sharp Corporation. Sharp develops and will mass produce new System LCD with embedded optical sensors to provide input capabilities including touch screen and scanner functions. Press Release, `http://sharp-world.com/corporate/news/`, August 31, 2007.

[8] Corning. Corning leads industry as first commercial supplier of Generation 6 Size glass substrates. Press Release, June 26, 2003.

[9] Sang-Soo Han, Kyoung-Moon Lim, Juhn S. Yoo, Young-Sik Jeong, Kyoung-Eon Lee, JoonKyu Park, Dae Hyun Nam, Seok-Woo Lee, Jin-Mo Yoon, Yun-Ho

Jung, Hyun-Sik Seo, and Chang-Dong Kim. 3.5 inch QVGA Low-Temparature Poly-Si TFT LCD with integrated driver circuits. In *Society for Information Display 2003 Digest of Technical Papers*, volume 34, Baltimore, USA, May 2003.

[10] Toshio Mizuki, Junko Matsunda, Yoshinobu Nakamura, Junkoh Takagi, and Toyonobu Yoshida. Large domains of continuous grain silicon on glass substrate for high-performance TFTs. *IEEE Transactions on Electron Devices*, 51(2), February 2004.

[11] Silicon Strategies. Sharp sets roadmap for Moore's Law of silicon-on-glass. `http://www.siliconstrategies.com/`, June 30, 2004.

[12] C. H. Gebotys, S. Ho, and C. C. Tiu. EM analysis of Rijndael and ECC on a wireless Java-based PDA. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 250–264, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[13] Kris Tiri. Prototype IC with WDDL and differential routing DPA resistance assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 354–365, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[14] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO '96*, LNCS 1109, pages 104–113. Springer-Verlag, 1996.

[15] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99*, LNCS 1666, pages 388–397. Springer-Verlag, 1999.

[16] Markus G. Kuhn. *Compromising Emanations: eavesdropping risks of computer displays.* PhD thesis, University of Cambridge, December 2003.

[17] Markus G. Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *4th Workshop on Privacy Enhancing Technologies, Toronto, Canada*, 2004.

[18] Intel Corporation. High-bandwidth Digital Content Protection system, revision 1.3, December 2006.

[19] Intel Corporation. High-bandwidth Digital Content Protection system, revision 1.0, February 2000.

[20] Niels Ferguson. Censorship in action: Silenced by the DMCA, August 2001.

[21] Keith Irwin. Four simple cryptographic attacks on HDCP, July 2001.

[22] S. Crosby, I. Goldberg, R. Johnson, D. Song, and D. Wagner. A cryptanalysis of the High-bandwidth Digital Content Protection system. In *Workshop on Security and Privacy in Digital Rights Management*, pages 192–200, 2001.

[23] J. Dhem and N. Feyt. Hardware and software symbiosis helps smart card evolution. *IEEE Micro*, 21(6), 2001.

[24] Oliver Kommerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology, Smartcard '99*, pages 9–20, Chicago, Illinois, USA, 1999. USENIX Association.

[25] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 5th printing (2001) edition, 1996.

[26] Ross Anderson. *Security Engineering*. John Wiley and Sons Inc., 2001.

[27] Macrovision Corporation. Analog Protection System. Presentation to the Analog Reconversion Discussion Group, March 5, 2003.

[28] Macrovision Corporation. Analogue Content Protection for DVD. `http://www.macrovision.com/products/macrovision_acp/`, Feb. 2005.

[29] Shiguo Lian, Zhongxuan Liu, Zhen Ren, and Haila Wang. Secure advanced video coding based on selective ecryption algorithms. *IEEE Transactions on Consumer Electronics*, 52(2):621–629, May 2006.

[30] Chengqing Li. On the security of a class of image encryption scheme. Cryptology ePrint Archive, Report 2007/339, 2007.

[31] Chengqing Li, Shujun Li, Dan Zhang, and Guanrong Chen. Cryptanalysis of a data security protection scheme for VoIP. *IEE Proceedings on Visual Image Signal Processing*, 153(1):1–10, 2006.

[32] Ross J. Anderson. A5 (Was: Hacking Digital Phones). `http://jya.com/crack-a5.htm`, 1994.

[33] J. Golic. Cryptanalysis of alleged A5 stream cipher. In *EUROCRYPT '97*, LNCS 1233, pages 239–255. Springer-Verlag, 1997.

[34] ANSI/IEEE Standard 802.11, 1999 edition, wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE Computer Society, September 1999.

[35] Bluetooth SIG. Specification of the Bluetooth system, version 1.2, November 5, 2003.

[36] Advanced Access Content System (AACS): Introduction and common cryptographic elements, revision 0.91. AACS LA LLC, February 17, 2006.

[37] Reponse to reports of attacks on AACS technology. AACS Press Release, `http://www.aacsla.com/press/`, January 24, 2007.

[38] AACS LA announces security updates. AACS Press Release, Reponse to Reports of Attacks on AACS Technology, `http://www.aacsla.com/press/`, April 16, 2007.

[39] NDS Press Room. Sky to incorporate NDS and Pace technology in new integrated personal television recorder. `http://www.nds.com/press_room/press_releases.html`, September 8, 2000.

[40] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[41] Horst Feistel. Cryptography. *Scientific American*, 228(5):15–23, May 1973.

[42] Advanced Encryption Standard, FIPS PUB 197. National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.

[43] A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A compact Rijndael hardware architecture with S-box optimization. In *ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, LNCS 2248, pages 239–253, Gold Coast, Australia, December 13, 2001. Springer.

[44] Data Encryption Standard FIPS PUB 46. National Institute of Standards and Technology, U.S. Department of Commerce, January 1977.

[45] Don Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38(3), May 1994.

[46] Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design.* O'Reilly, 1998.

[47] National Institute of Standards and Technology, U.S. Department of Commerce. Announcing proposed withdrawal of Federal Information Processing Standard (FIPS) for the Data Encryption Standard (DES) and request for comments. *Federal Register*, 69(142):44509–44510, July 26, 2004.

[48] Data Encryption Standard, FIPS PUB 46-3. National Institute of Standards and Technology, U.S. Department of Commerce, October 1999.

[49] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback. Report on the development of the Advanced Encryption Standard (AES). Technical report, Information Technology Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce, October 2000.

[50] A symmetric key encryption algorithm: MULTI-S01. Hitachi, Ltd., 2001.

[51] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001, Revised Papers*, LNCS 2259, pages 1–24, Toronto, Canada, August 2001. Springer-Verlag.

[52] Patrik Ekdahl and Thomas Johansson. Another attack on A5/1. *IEEE Transactions on Information Theory*, 49(1):284–289, January 2003.

[53] ETSI/SAGE. Specification of the 3GPP confidentiality and integrity algorithms UEA2 and UIA2. – Document 5: Design and evaluation report. version 1.1, September 2006.

[54] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[55] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology - CRYPTO '84*, pages 10–18. Springer-Verlag, 1985. (also published in IEEE Transactions on Information Theory, IT-31, no. 4, 1985, pp.469–472).

[56] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[57] D Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[58] Victor Miller. Use of elliptic curves in cryptography. In *CRYPTO '85*, pages 417–426, LNCS 218, 1985.

[59] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[60] Guerric Meurice de Dormale and Jean-Jacques Quisquater. High-speed hardware implementations of Elliptic Curve Cryptography: A survey. *Journal of Systems Architecture*, 53(2-3):72–84, 2007.

[61] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July 1948.

[62] Ian Goldberg and David Wagner. Randomness and the Netscape browser, how secure is the world wide web? Dr. Dobb's Journal, `http://www.ddj.com/windows/184409807`, January 1, 1996.

[63] Eric J. Hoffman. Random number generator. U.S. Patent #6,061,702, May 9, 2000.

[64] M. Bucci, L. Germani, R. Luzzi, P. Tommansino, A. Trifiletti, and M. Varanuovo. A high-speed oscillator-based truly random number source for cryptographic applications on a smartCard IC. *IEEE Transactions on Computers*, 52(4), April 2003.

[65] M. Ditchtl. Random number generator and method for generating a random number. U.S. Patent Application #2004/0010526, Janruary 15, 2004.

[66] C. Dike and E. Burton. Miller and noise effects in a synchronizing flip-flop. *IEEE Journal of Solid-State Circuits*, 34(6), June 1999.

[67] D. C. Ranasinghe, D. Lim, S. Devadas, D. Abbott, and P. H. Cole. Random numbers from metastability and thermal noise. *IEE Electronics Letters*, 41(16), August 4 2005.

[68] Michael A. Epstein. Method and apparatus for generating random numbers using flip-flop meta-stability. U.S. Patent #6631390, October 7, 2003.

[69] Andre Weimerskirch. Method and apparatus for preventing noise from influencing a random number generator based on flip-flop meta-stability. U.S. Patent #6963888, November 8, 2005.

[70] D. J. Kinniment and E. G. Chester. Design of on-chip random number generator using metastability. In *Proceedings of ESSCIRC 2002*, September 2002.

[71] Shinobu Fujita, Ken Uchida, and Shinichi Yasuda. Ultrasmall random number generators for high-level information security. *Toshiba Review*, 58(8), 2003.

[72] Steven E. Wells and David A. Ward. Duty cycle corrector for a random number generator. U.S. Patent #6643374, November 4, 2003.

[73] E Trichina, M Bucci, D De Seta, and R Luzzi. Supplemental cryptographic hardware for smart cards. *IEEE Micro*, 21:26–35, 2001.

[74] Preliminary call for stream cipher primitives, v1.0. ECRYPT NoE, European Network of Excellence for Cryptology, November 30, 2004.

[75] Data Encryption Standard, FIPS PUB 81. National Institute of Standards and Technology, U.S. Department of Commerce, December 2, 1980.

[76] Patrik Ekdahl. *On LFSR based Stream Ciphers.* PhD thesis, Lund University, November 2003.

[77] Blaise de Vigenère. *Traicté de Chiffres.* -, 1585.

[78] David Kahn. *The Codebreakers.* Macmillan, 1967.

[79] Giovan Batista Belaso. *La cifra del. Sig.* -, 1553.

[80] Gilbert S. Vernam. Secret signaling system. U.S. Patent #1310719, July 22, 1919.

[81] Lyman F. Morehouse. Ciphering system. U.S. Patent #1356546, October 26, 1920.

[82] I. J. Good, Donald Michie, and Geoffrey Timms. HW25/4, General Report on Tunny, Volume I. The National Archives, Kew, UK, released September 28, 2000, 1945.

[83] I. J. Good, Donald Michie, and Geoffrey Timms. HW25/5, General Report on Tunny, Volume II. The National Archives, Kew, UK, released September 28, 2000, 1945.

[84] Mellville Klein. Securing record communications, the TSEC/KW26. Center for Cryptologic History, National Security Agency, U.S.A., 2003.

[85] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[86] A. Klimov. *Applications of T-functions in Cryptography.* PhD thesis, Weizmann Institute of Science, Israel, 2004.

[87] R. Rivest. The RC4 encryption algorithm. RSA Data Security, Inc., March 12, 1992.

[88] The SSL protocol, version 3.0. Netscape Communications Corporation, November 1996.

[89] IEEE Standard 802.11i, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Computer Society, July 23 2004.

[90] Hongjun Wu. The misuse of RC4 in Microsoft Word and Excel. Cryptology ePrint Archive, Report 2005/007, 2005.

[91] Thank you bob anderson. `http://cypherpunks.venona.com/archive/1994/09/msg00304.html`, 1994.

[92] Scott Fluhrer and David McGrew. Statistical analysis of the alleged RC4 keystream generator. In *Fast Software Encryption: 7th International Workshop, FSE 2000*, LNCS 1978, pages 66–71, New York, USA, April 2000. Springer.

[93] A. Stubblefield, J. Ioannidis, and A. Rubin. Using the Fluhrer, Mantin, and Shamir attack to break WEP, TD4ZCPZZ. Technical report, AT&T Labs, August 6, 2001.

[94] R. Rivest. RSA security response to weaknesses in key scheduling algorithm of RC4, Tech Note. Technical report, RSA Data Security Inc., October 2001.

[95] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit WEP in less than 60 seconds. Cryptology ePrint Archive, Report 2007/120, 2007.

[96] Souradyuti Paul and Bart Preneel. A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In *Fast Software Encryption: 9th International Workshop, FSE 2003*, LNCS 3017, pages 245–259. Springer, 2004.

[97] Violeta Tomsevic, Slobodan Bojanic, and Octavio Nieto-Taladriz. Finding an internal state of RC4 stream cipher. *Information Sciences: an International Journal*, 177(7):1715–1727, April 2007.

[98] Marc Briceno, Ian Goldberg, and David Wagner. A pedagogical implementation of A5/1. `http://www.gsm-security.net/papers/a51.shtml`, 1999.

[99] Eli Biham and Orr Dunkelman. Cryptanalysis of the A5/1 GSM stream cipher. In *INDOCRYPT 2000*, LNCS 1977, pages 43–51, Calcutta, India, 2000. Springer-Verlag.

[100] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *Fast Software Encryption: 7th International Workshop, FSE 2000*, LNCS 1978, pages 37–44, New York, USA, 2001. Springer.

[101] Patrik Ekdahl and Thomas Johansson. Some results on correlations in the Bluetooth stream cipher. In *10th Joint Conference on Communications and Coding*, pages 210–224, Obertauern, Austria, March 2000.

[102] Yi Lu, Willi Meier, and Serge Vaudenay. The conditional correlation attack: A practical attack on Bluetooth encryption. In *Advances in Cryptology - CRYPTO 2005*, LNCS 3621, pages 97–117. Springer, 2005.

[103] Bluetooth SIG. Bluetooth Core Specification version 2.1 EDR makes intial device set-up faster, easier and increases battery life. Press Release, March 27, 2007.

[104] Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher SNOW. In *Selected Areas in Cryptography - SAC 2002*, LNCS 2595, pages 47–61. Springer-Verlag, September 2002.

[105] Patrik Ekdahl and Thomas Johansson. SNOW – a new stream cipher. In *Proceedings of 1st Open NESSIE Workshop*, 2000.

[106] Don Coppersmith, S. Halevi, and C. S. Jutla. Cryptanalysis of stream ciphers with linear masking. In *CRYPTO 2002*, LNCS 2442, pages 515–532. Springer-Verlag, 2002.

[107] Philip Hawkes and Gregory G. Rose. Guess-and-determine attacks on SNOW. In *Selected Areas in Cryptography - SAC 2002*, LNCS 2595. Springer-Verlag, 2002.

[108] Olivier Billet and Henri Gilbert. Resistance of SNOW 2.0 against algebraic attacks. In *Topics in Cryptology - CT-RSA 2005*, LNCS 3376, pages 19–28. Springer, 2005.

[109] Dai Watanabe, Alex Biryukov, and Christophe De Cannière. A distinguishing attack of SNOW 2.0 with linear masking method. In *Selected Areas in Cryptography, SAC 2004*, LNCS 3006, pages 222–233. Springer, 2004.

[110] Alexander Maximov and Thomas Johansson. Fast computation of large distributions and its cryptographic applications. In *ASIACRYPT 2005*, Madras, India, 2005.

[111] ISO/IEC 18033-4:2005, Information technology – Security techniques – Encryption algorithms – Part 4: Stream ciphers.

[112] Christophe De Cannière. eSTREAM Testing Framework. eSTREAM, ECRYPT Stream Cipher Project, `http://www.ecrypt.eu.org/stream/perf/`, November 2005.

[113] Performance Figures, Intel Pentium M, revision 193. eSTREAM, ECRYPT Stream Cipher Project, `http://www.ecrypt.eu.org/stream/phase3perf/2007a/pentium-m/`.

[114] Mitsuru Matsui. New block encryption algorithm MISTY. In *Fast Software Encryption, 4th International Workshop, FSE '97*, LNCS 1267, pages 54–68, Haifa, Israel, January 1997. Springer.

[115] NESSIE security report v2.0. Project for New European Schemes for Signature, Integrity, and Encryption, Information Society Technologies Programme, European Commission, February 2003.

[116] NESSIE consortium. Portfolio of recommended cryptographic primitives, February 2003.

[117] NESSIE project announces final selection of crypto algorithms. NESSIE Press Release, February 27, 2003.

[118] Adi Shamir. Stream ciphers: Dead or alive? The State of the Art of Stream Ciphers, SASC, Brugge, Belgium, October 15, 2004.

[119] Steve Babbage. Stream ciphers: What does industry want? The State of the Art of Stream Ciphers, SASC, Brugge, Belgium, October 15, 2004.

[120] Elaine Barker James Nechvatal, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the Advanced Encryption Standard (AES). National Institute of Standards and Technology, U.S. Department of Commerce, October 2000.

[121] Kazumaro Aoki and Helger Lipmaa. Fast implementations of AES candidates. In *Third AES Candidate Conference, New York, USA*, April 13–14 2000.

[122] Martin Hell, Thomas Johansson, and Willi Meier. Grain - a stream cipher for constrained environments. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[123] Christophe De Cannière and Bart Preneel. Trivium - specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030, 2005.

[124] Steve Babbage and Matthew Dodd. The stream cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project, April 2005.

[125] An Braeken, Joseph Lano, Nele Mentens, Bart Preneel, and Ingrid Verbauwhede. SFINKS: A synchronous stream cipher for restricted hardware environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/026, 2005.

[126] Doug Whiting, Bruce Schneier, Stefan Lucks, and Frederic Muller. Phelix: Fast encryption and authentication in a single cryptographic primitive. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[127] Christophe De Cannière and Bart Preneel. TRIVIUM - a stream cipher construction inspired by block cipher design principles. In *SASC 2006, Stream Ciphers Revisited*, pages 203–215, Leuven, Belgium, 2006.

[128] Vincent Rijmen. Efficient implementation of the Rijndael S-box. `http://www.iaik.tu-graz.ac.at/research/krypto/aes/old/~rijmen/rijndael/sbox.pdf`.

[129] Tim Good, W. Chelton, and Mohammed Benaissa. Review of stream cipher candidates from a low resource hardware perspective. In *SASC 2006 - Stream Ciphers Revisited*, The State of the Art of Stream Ciphers, Leuven, Belgium, February 2006.

[130] Niels Ferguson, Doug Whiting, Bruce Schneier, Stefan Lucks, and Tadayoshi Kohno. Helix: Fast encryption and authentication in a single cryptographic primitive. In *Fast Software Encryption - 10th International Workshop, FSE 2003*, volume 2887, pages 330–346, Lund, Sweden, February 24-26, 2003. Springer.

[131] End of phase 1. ECRYPT NoE, European Network of Excellence for Cryptology, `http://www.ecrypt.eu.org/stream/endofphase1.html`.

[132] Nicolas Courtois. Cryptanalysis of Sfinks. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/002, 2006.

[133] H. Wu and Bart Preneel. Differential-linear attacks against the stream cipher phelix. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/056, 2006.

[134] Alexander Maximov and Alex Biryukov. Two trivial attacks on Trivium. In *SASC 2007*, The State of the Art of Stream Ciphers, Bochum, Germany, January 2007.

[135] Cameron McDonald, Chris Charnes, and Josef Pieprzyk. Attacking Bivium with MiniSat. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/040, 2007.

[136] Cameron McDonald, Chris Charnes, and Josef Pieprzyk. An algebraic analysis of Trivium ciphers based on the boolean satisfiability problem. Cryptology ePrint Archive, Report 2007/129,`http://eprint.iacr.org/`, 2007.

[137] Jin Hong and Woo-Hwan Kim. TMD-tradeoff and state entropy loss considerations of streamcipher MICKEY. ECRYPT Stream Cipher Project, Report 2005/055, 2005.

[138] Steve Babbage and Matthew Dodd. The stream cipher MICKEY 2.0. eSTREAM, ECRYPT Stream Cipher Project, June 2006.

[139] E. Dawson, A. Clark, J. Golic, W. Millan, L. Penna, and L. Simpson. The lili-128 keystream generator. NESSIE Project Submission, November 2000.

[140] Don Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In *Advances in Cryptology - CRYPTO '93*, LNCS 773, pages 22–39. Springer-Verlag, 1993.

[141] Havard Molland and Tor Helleseth. An improved correlation attack against irregular clocked and filtered keystream generators. *Advances in Cryptology – CRYPTO 2004*, pages 373–389, LNCS 3152 2004.

[142] Patrik Ekdahl, Willi Meier, and Thomas Johansson. Predicting the shrinking generator with fixed connections. In *Advances in Cryptology - EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques*, LNCS 2656, pages 330–344, Warsaw, Poland, May 4-8, 2003. Springer.

[143] Shahram Khazaei, Mehdi Hassanzadeh, and Mohammad Kiaei. Distinguishing attack on Grain. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/071, 2005.

[144] C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of Grain. In *SASC 2006, Stream Ciphers Revisited*, 2006.

[145] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007.

[146] Martin Hell, Thomas Johansson, and Willi Meier. A stream cipher proposal: Grain-128. eSTREAM, ECRYPT Stream Cipher Project, 2006.

[147] Performance Figures, AMD Athlon 64 X2, revision 206. eSTREAM, ECRYPT Stream Cipher Project, `http://www.ecrypt.eu.org/stream/phase3perf/2007a/amd64/`, January 2007.

[148] Performance Figures, Intel Pentium 4, revision 206. eSTREAM, ECRYPT Stream Cipher Project, January 2007.

[149] Frank K. Gurkaynak, Peter Luethi, Nico Bernold, René Blattmann, Victoria Goode, Marcel Marghitola, Hubert Kaeslin, Norbert Felber, and Wolfgang Fichtner. Hardware evaluation of eSTREAM candidates. In *SASC 2006, Stream Ciphers Revisited*, The State of the Art of Stream Ciphers, Leuven, Belgium, February 2006.

[150] Tim Good and Mohammed Benaissa. Hardware results for selected stream cipher candidates. In *SASC 2007, Bochum, Germany*, The State of the Art of Stream Ciphers, February 2007.

[151] Philippe Bulens, Kassem Kalach, Francois-Xavier Standaert, and Jean-Jacques Quisquater. FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile. In *SASC 2007*, The State of the Art of Stream Ciphers, Bochum, Germany, February 2007.

[152] Martin Feldhofer. Comparison of low-power implementations of Trivium and Grain. In *SASC 2007*, The State of the Art of Stream Ciphers, Bochum, Germany, February 2007.

[153] Kris Gaj, Gabriel Southern, and Ramakrishna Bachimanchi. Comparison of hardware performance of selected phase II eSTREAM candidates. In *SASC*

2007, The State of the Art of Stream Ciphers, Bochum, Germany, February 2006.

[154] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. *IEE Proceedings in Information Security*, 152(1):13–20, October 2005.

[155] Taiwan Semiconductor Manufacturing Company Ltd. TSMC 90 nanometer libraries support NexsysSM 90nm production. New Archives, `http://www.tsmc.com/english/page_transsfer/pr_news_archive.htm`, February 1, 2005.

[156] Stefan Mangard, Manfred Aigner, and Sandra Dominikus. A highly regular and scalable AES harware architecture. *IEEE Transactions on Computers*, 52(4):483–491, April 2003.

[157] Marcin Rogawski. Hardware evaluation of eSTREAM candidates: Grain, Lex, Mickey128, Salsa20 and Trivium. In *SASC 2007*, The State of the Art of Stream Ciphers, Bochum, Germany, February 2006.

[158] Paris Kitsos. On the hardware implementation of the MICKEY-128 stream cipher. Cryptology ePrint Archive, Report 2005/301, 2005.

[159] Tim Good and Mohammed Benaissa. AES on FPGA from the fastest to the smallest. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 427–440, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[160] Pawel Chodowiec and Kris Gaj. Very compact FPGA implementation of the AES algorithm. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop*, LNCS 2779, pages 319–333, Cologne, Germany, September 2003. Springer.

[161] Daniel J. Bernstein. Cycle counts for authenticated encryption. In *SASC 2007*, The State of the Art of Stream Ciphers, January 2007.

[162] Kris Tiri, David D. Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. WDDL based AES cryptographic coprocessor. In *VLSI Symposium 2005*, 2005.

[163] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 172–186, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[164] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev. Improving the security of dual-rail circuits. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, LNCS 3156, pages 282–297. Springer-Verlag, 2004.

[165] D. Shang, F. Burns, A. Bystrov, A. Koelmans, D. Sokolov, and A. Yakovlev. A low and balanced power implementation of the AES security mechanism using self-timed circuits. In *Integrated Circuit and System Design, Integrated Circuit and System Design, PATMOS 2004*, LNCS 3254, pages 471–480, Santorini, Greece, September 2004. Springer.

[166] Sylvain Guilley. The "backend duplication" method. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 383–397, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[167] Short report on the end of the second phase. ECRYPT NoE, European Network of Excellence for Cryptology, March 26 2007.

[168] C. Berbain, O. Billet, A. Canteaut, Nicolas Courtois, B. Bebraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim v2. eSTREAM, ECRYPT Stream Cipher Project, 2007.

[169] Danilo Gligoroski, Smile Markovski, Ljupco Kocarev, and Marjan Gusev. Edon80. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[170] F. Arnault, T. P. Berger, and C. Lauradoux. Update on F-FCSR stream cipher. eSTREAM, ECRYPT Stream Cipher Project, 2007.

[171] Joan Daemen and Paris Kitsos. The self-synchronizing stream cipher moustique. eSTREAM, ECRYPT Stream Cipher Project, June 2006.

[172] Cees Jansen, Tor Helleseth, and Alexander Kolosha. Cascade jump controlled sequence generator and pomaranch stream cipher (version 3). eSTREAM, ECRYPT Stream Cipher Project, 2006.

[173] C. Berbain, O. Billet, A. Canteaut, Nicolas Courtois, B. Bebraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim-128. eSTREAM, ECRYPT Stream Cipher Project, 2007.

[174] Steve Babbage and Matthew Dodd. The stream cipher MICKEY-128 2.0. eSTREAM, ECRYPT Stream Cipher Project, August 2006.

[175] Matt Blaze, Whitfield Diffie, Ronald L. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thompson, and Michael Wiener. Minimal key lengths for

symmetric ciphers to provide adequate commercial security. Available at http://www.schneier.com/paper-keylength.html, January 1996.

[176] D. J. Bernstein. Understanding brute force. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/036, 2005.

[177] National Institute of Standards and Technology. SKIPJACK and KEA algorithms. http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm, May 29, 1998.

[178] Ecrypt yearly report on algorithms and keysizes (2004), D.SPA.10, 2005.

[179] National Institute of Standards and Technology. Recommendation for Key Management — Part 1: General, NIST Special Publication 800-57, Draft, May 2006.

[180] ECRYPT NoE, European Network of Excellence for Cryptology. estream phase 2, faq. http://www.ecrypt.eu.org/stream/, September 5 2006.

[181] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, July 1980.

[182] Z. E. Schnabel. The estimation of the total fish population of a lake. *American Mathematical Monthly*, 45(6):348–352, 1938.

[183] Richard Clayton and Mike Bond. Experience using a low-cost FPGA design to crack DES keys. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, LNCS 2523, pages 579–592, Redwood Shores, USA, August 2002. Springer.

[184] G.E. Moore. Cramming more components onto integrated circuits. *Electronics*, 30(8), April 19 1965.

[185] I. J. Good, Donald Michie, and Geoffrey Timms. General report on tunny. http://www.alanturing.net/turing_archive/archive/index/tunnyreportindex.html, 1945.

[186] Whitfield Diffie and Martin E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10(6):74–84, 1977.

[187] M. Wiener. Efficient DES key search, TR-244. Technical report, Carleton University, May 1994.

[188] RSA Laboratories. Announcements: The RSA data security secret-key challenge. *Cryptobytes*, 2(3):16, 1997.

[189] Jean-Jacques Quisquater and Francois-Xavier Standaert. Exhaustive key search of the DES: Updates and refinements. In *SHARCS '05 - Special-purpose Hardware for Attacking Cryptographic Systems*, 2005.

[190] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler. How to break DES for €8,980. In *SHARCS '06*, Special-purpose Hardware for Attacking Cryptographic Systems, Cologne, Germany, April 2006.

[191] Thorsten Kleinjung. RSA200. `http://www.loria.fr/~zimmerma/records/rsa200`, May 9, 2005.

[192] Adi Shamir and Eran Tromer. Factoring large numbers with the TWIRL device. In *CRYPTO 2003*, LNCS 2729, pages 1–26, Santa Barbara, CA, USA, August 2003. Springer.

[193] Jens Franke, Thorsten Kleinjung, Christof Paar, Jen Pelzl, Christine Priplata, and Colin Stahlke. SHARK - a realizable special hardware sieving device for factoring 1024-bit integers. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, LNCS 3659, pages 119–130, Edinburgh, UK, August 29 – September 1, 2005. Springer.

[194] Tetsuya Izu, Jun Kogure, and Takeshi Shimoyama. CAIRN 3: An FPGA implementation of the sieving step with the lattice sieving, extended abstract. In *SHARCS '07 - Special-purpose Hardware for Attacking Cryptographic Systems*, Vienna, Austria, September 2007.

[195] J. Pelzl, M. Simka, T. Kleinjung, J. Franke, C. Priplata, C. Stahlke, M. Drutarovsky, V. Fischer, and Christof Paar. Area-time efficient hardware architecture for factoring integers with the elliptic curve method. *IEE Proceedings in Information Security*, 152(1):67–78, 2005.

[196] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing*, 26(5):1484–1509, 1997.

[197] J. A. Jones and M. Mosca. Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *Journal of Chemical Physics*, 109(5):1648–1653, August 1998.

[198] Isaac L. Chuang, Lieven M. K. Vandersypen, Xinlan Zhou, Debbie W. Leung, and Seth Lloyd. Experimental realization of a quantum algorithm. *Nature*, 393:143–146, May 14, 1998.

[199] L. M. K. Vandersyoen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and Isaac L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(20/27):883–887, December 2001.

[200] C. Negrevergne, T. S. Mahesh, C. A. Ryan, M. Ditty, F. Cyr-Racine, W. Power, N. Boulant, T. Havel, D. G. Cory, and R. Laflamme. Benchmarking quantum control methods on a 12-qubit system. *Physics Review Letters*, 96(170501), May 1, 2006.

[201] T. E. Guneysu. Efficient hardware architectures for solving the discrete logarithm problem on elliptic curves. Master's thesis, Horst Gortz Institute, Ruhr University of Bochum, February 2006.

[202] P. van Oorschot and M. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.

[203] J. Pelzl, T. Guneysu, C. Paar, C. Schleiffer, G. Pfeiffer, and M. Schimmler. Cryptanalysis with a cost-optimized FPGA cluster. In *UCLA IPAM Workshop IV, Special Purpose Hardware for Cryptography: Attacks and Applications*, Special-purpose Hardware for Attacking Cryptographic Systems, December 4–8 2006.

[204] J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. Cryptology ePrint Archive, Report 2005/090, 2005.

[205] Xilinx Staff. Celebrating 20 years of innovation. *Xcell Journal*, -(48), 2004.

[206] Altera. Altera's new Cyclone II FPGAs offer 30 percent lower costs than previous generation. Altera Press Release, June 28, 2004.

[207] Altera. Hardcopy 2 backgrounder. Altera Virtual Press Kit Release, June 28, 2004.

[208] Bureau of Labor Statistics, U.S. Department of Labor. Consumer Price Index news release, January 1996. `http://inflationdata.com/Inflation/Consumer_Price_Index/HistoricalCPI.aspx`, February 28, 1996.

[209] Bureau of Labor Statistics, U.S. Department of Labor. Consumer Price Index news release, October 2007. `http://inflationdata.com/Inflation/Consumer_Price_Index/HistoricalCPI.aspx`, November 15, 2007.

[210] Altera. Altera ships industry's first 65-nm low-cost FPGA. Altera Press Release, March 19, 2007.

[211] Altera. Cyclone III FPGA family press FAQ. Altera Virtual Press Kit Release, March 20, 2007.

[212] Intel Corporation. Intel continues push down power-optimization path with Intel Xeon processor line. Intel News Release, September 26, 2005.

[213] S. Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, pages 161–166. IEE, May 16–18, 1995.

[214] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *ASIACRYPT 2000*, LNCS 1976. Springer-Verlag, 2000.

[215] J. Quisquater, F. Standaert, G. Rouvroy, J. David, and J. Legat. A cryptanalytic time-memory tradeoff: First FPGA implementation. In *FPL 2002, Field Programmable Logic and Application*, 2002.

Appendices

Appendix A

Letter on Key Search of Hardware Focused Stream Ciphers

Practical key search engine for analysing hardware focused stream ciphers

I. Devlin, A. Purvis

A FPGA based key search system that specifically targets new hardware stream ciphers is presented. The design makes use of a data element to enhance the exhaustive search process and is practical and economically feasible to implement. The results indicate that the security of 80-bit key lengths in current stream cipher proposals should be considered insufficient to protect data.

Introduction: Stream ciphers are widely used in communication systems. They use a small fixed length secret key to efficiently generate a large secret binary stream which can then be used for message encryption. The security of the binary stream, usually referred to as the keystream, depends on both the generation algorithm and the key itself. Design choices typically trade-off security against performance with the balance requiring an evaluation of attack feasibility.

Exhaustive search is an attack strategy that in the context of stream ciphers targets the key itself. It involves searching the entire key space for the single combination that decrypts to produce a match to some held plaintext information. Traditionally the cryptographical exhaustive search attack has been focused on block ciphers and recent attacks by the machine COPACOBANA [1] have shown the 56-bit keys of the Data Encryption Standard (DES) [2] to be a very susceptible and low cost target.

Recently proposed stream ciphers Grain v1 [3] and Trivium [4] that aim to avoid the algorithmic vulnerabilities typical of previous generation stream ciphers use 80-bit keys, a search problem potentially seven orders of magnitude more complex. However, as synchronous stream ciphers possess data independent keystream generation and generally use a simple combining function, collection of plaintext-ciphertext pair data by an adversary will allow the trivial recovery of original keystream material. In a situation where multiple plaintext-ciphertext sections encoded under different keys are available the search effort required to recover one of these keys falls (Fig. 1) due to the birthday paradox with the consequence that searching for a key in a larger key space may become feasible; an observation briefly touched upon by Hellman [5].

In this letter we use this observation as motivation to build a practical key search machine, named 'DELUGE,' and so draw attention to the potential false sense of security afforded by a 80-bit key length.

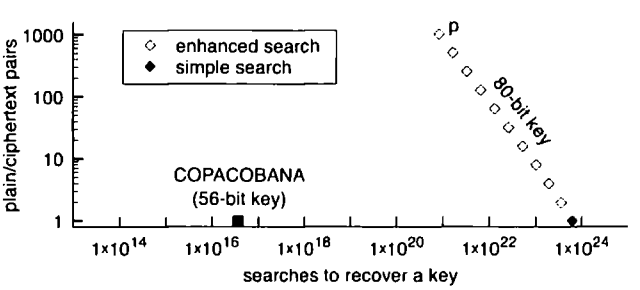


Fig. 1. *Impact of Plain/Ciphertext Data Pairs on Key Search.* *p* The 80-bit key is brought closer to vulnerable 56-bit key length with more data pairs; in linear terms, significantly closer

Attack Scenario: Server to user content transmission is a common scenario in which the efficiency and speed of hardware

stream ciphers are ideally suited. In this situation it is likely that an adversary may be able to collect a number of these transmissions and use known header information to gain the data pairs necessary to recover initial bytes of multiple keystreams and allow an efficient enhanced search. It is reasonably assumed in the design of DELUGE that an adversary would be unconcerned over which user's data is retrieved and that differing steam cipher initialisation vectors (IV) could be filtered out.

Internal Architecture: The search unit architecture of DELUGE which targets the use of 80-bit keys in the outlined scenario is detailed in Fig. 2. The main search computation is performed by the search core of Fig. 2a. In this system block the enhanced form of exhaustive search is implemented with generated keystreams compared to stored data from the plaintext-ciphertext pairs. A two stage process is used due to the inefficiency of performing full 80-bit comparisons for every key trial. The first half of Fig. 2a using a single cycle key agility stream cipher and short 16-bit partial comparisons to enable rapid elimination of the vast majority of keys. The second stage being the 80-bit keystream compare unit which performs a full comparison for reliable key detection. The lack of pressure for high performance in this later stage allows a compact cipher implementation and serial comparison structure to be used with the expander encoder of Fig. 2a buffering and converting the parallel comparison results into the serial key and data information required.

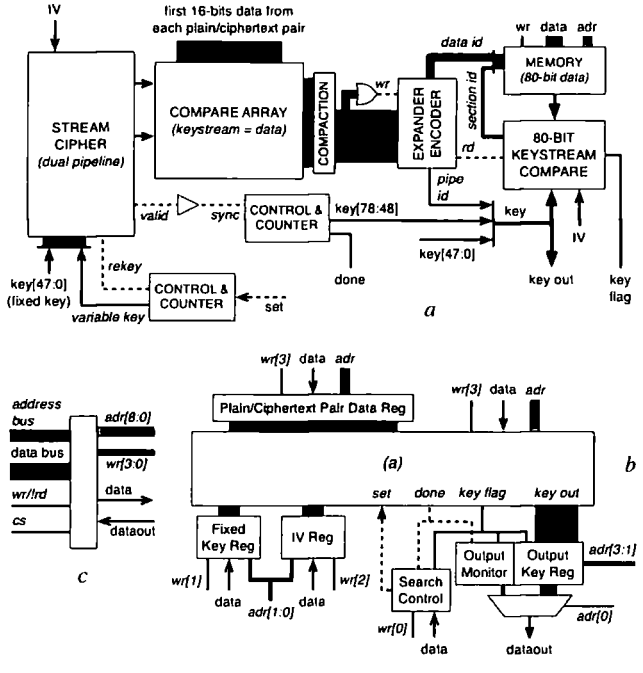


Fig. 2. *Search Unit System Architecture.*
a Key search core with dual keystream pipeline
b Internal search unit layout
c Bus interface of search unit

The search unit and indeed the key search core itself, are designed to be cipher independent. However, for testing, the proposed stream cipher Grain v1 is used due to its hardware efficient performance with pipelining of the 10-cycle initialisation process enabling key agility requirements at the search core's front-end to be met. In this pipelining, advantage can be made of observed invariance in early initialisation stages when two or more cipher pipelines share similar keys. In the case of Grain v1 the dual pipeline of Fig. 2a derives 53k $keys/LE.s$ on Cyclone FPGAs

for pipes sharing key bits $K_{78..0}$, a 15% efficiency increase in search performance. The remaining key bit, K_{79}, acting as the pipe identifier. A larger 58k *keys/LE.s* quad pipeline is also possible using shared key bits $K_{77..0}$.

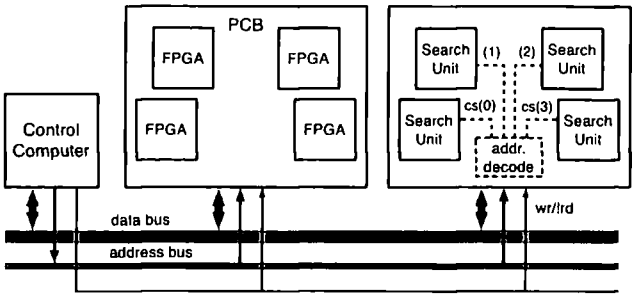


Fig. 3. *Inter-unit buses and possible chip layout for DELUGE.*

System Architecture: DELUGE's internal interface structure follows a conventional data and address bus design (Fig. 3) so that it is suitable for implementation in a generic computation machine and will allow a single computer to monitor and manage a large number of search units through the use of chip-select signal decoders. The output key register (Fig. 2b) of each unit can be polled through (Fig. 2c) to check if the allocated key space has been searched and if a possible key has been flagged. Writing to the bus interface of a search unit allows the reallocation of key space and loading of new IV and data information.

Results and discussion: A 2 keystream, 64 plainstream practical demonstrator search unit was implemented on a Stratix FPGA using 10k logic elements with an APEX 20K acting as the control computer. The search core was run at 50MHz and tested 6.4 billion key-plainstream combinations every second requiring allocation by the controller of a new section of key space every 44 seconds via the 10MHz bus. Simulation of the search unit for larger more cost effective devices such as the Cyclone II range of FPGAs provides the basis of results for a full multi-chip search machine capable of targeting 80-bit keys (Fig. 4b,c,d).

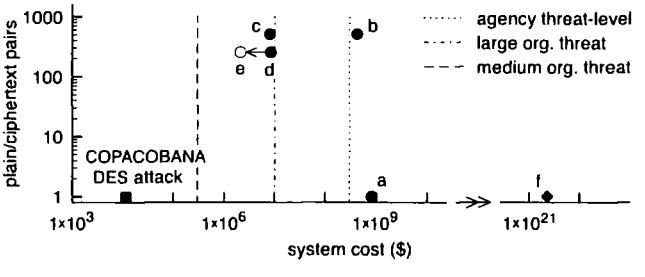


Fig. 4. *80-bit key space attack cost using DELUGE*
a Year recovery, simple search configuration
b Week recovery, dual pipeline, enhanced search
c Year recovery, dual pipeline, enhanced search
d Year recovery, quad pipeline, enhanced search
e Prediction for enhanced search in 2009, year recovery
f Estimate for year recovery of a 128-bit key

A large improvement in machine cost is gained by the move from simple exhaustive search (Fig. 4a) to the data enhanced search of DELUGE. With the use of 512 data elements (Fig. 4c) the advantage is largely maximised and in fact Fig. 4d may represent the optimal operating point due to a less demanding data requirement. The system cost values suggest that a machine

of this type may already fall into the realm of practical concern when dealing with high value, long-term data and FPGA development will continue to reduce costs (Fig. 4e) forcing the consideration of key search as a possible threat from medium sized organisations. Further performance improvements may be possible through use of ASIC or structured ASIC technology but the aspects of flexibility, cost and privacy will tend to favour reconfigurable logic making the described machine representative of the potential of key search attacks.

Conclusion: Given future trends in FPGA cost-performance the 80-bit key length of some newly proposed stream ciphers is insufficient for a wide variety of applications and would place undesirable constraints on their use. The cost of using larger keys, such as moving to a 128-bit key length, need not significantly impact on decoding performance and hardware efficiency [6] but would significantly increase search complexity and hence attack cost (Fig. 4f) making the long term viability of stream cipher proposals much more attractive.

References

[1] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, M. Schimmler,: How to Break DES for €8,980. SHARCS 2006. http://www.copacobana.org/

[2] National Institute of Standards and Technology: Data Encryption Standard, FIPS PUB 46. U.S. Dept. of Commerce, Jan 1977.

[3] M. Hell, T. Johansson, W. Meier: Grain - A Stream Cipher for Constrained Environments. Special Issue on Security of Computer Network and Mobile Systems. International Journal of Wireless and Mobile Computing, 2006.

[4] C. De Cannière, B. Preneel: TRIVIUM - Specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 (2005). http://www.ecrypt.eu.org/stream

[5] M. E. Hellman: A Cryptanalytic Time-Memory Trade-Off. IEEE Trans. on Information Theory, Vol. 26, No. 4, July 1980.

[6] T. Good, M. Benaissa: Hardware results for selected stream cipher candidates SASC 2007, Stream Ciphers Revisited.

Appendix B

System Operation Information for DELUGE

```
//-----------------------------------------------------------------
//  cs, wr  | addr[msb..lsb] |      data       | NOTES...          |
//-----------------------------------------------------------------
//   0   X  |        X       |      ZZZZ       |                   |
//-----------------------------------------------------------------
//   1   0  |      X         |   0  | {0001} =>done | read status reg   |
//          |                |      | {0002} =>flag |  (outputing data) |
//          |  X  |  C  |  1  |      | {search key}  | read o/p key reg  |
//          |     |     |     |      |               |  (C selects part) |
//-----------------------------------------------------------------
//   1   1  |  00 |     X       |                    | write ack         |
//          |     |             |      XXX1          |     (rekeys core) |
//          |     |             |      XXX2          |     (clears flag) |
//          |  01 |  X  |  A    |      rKey          | write reg key     |
//          |     |     |       |                    |       (A -> part) |
//          |  10 |  X  |  A    |       IV           | write IV          |
//          |     |     |       |                    |       (A -> part) |
//          |  11 |  J  |  C    |      plain         | write plainstream |
//          |     |     |       |                    |   (J -> position) |
//          |     |     |       |                    |   (C -> part    ) |
//-----------------------------------------------------------------

//-----------------------------------------------------------------
// **   raised flag indicates a successful key has been found
// **   note: a two cylce latency is present on outputs
//-----------------------------------------------------------------
```

```
//-----------------------------------------------------------
//  xclk    |  interface clock - optimise for compatibility    |
//  clk     |  core clock      - optimise for max search speed  |
//-----------------------------------------------------------
//  xaclr   |  reset system interface to default state          |
//          |          (clears plain, key & IV registers)       |
//  aclr    |  reset search core to default state               |
//          |          (use after loading new register key)     |
//-----------------------------------------------------------
//  gpause  |  pause core (sleep)                               |
//-----------------------------------------------------------
```

Appendix C

Code Base - DELUGE

The Verilog hardware description language code that describes the DELUGE search unit is included in the accompanying CD-ROM to this section. This code may be read by any general purpose text-editor but is primarily aimed at Altera's Quartus 7 design environment and the Cyclone family of FPGAs. The files contained in the top-level folder '`DELUGE verilog hdl files`' are as follows:

`Deluge.v`
`Del_FrontEndSearch.v`
`Del_TranslationUnit.v`
`Del_BackEndFullSearch.v`

The related stream cipher files are contained in the other top-level folder '`StreamCipher verilog hdl files`', and are as follows:

`Stream_Core.v`
`SC_Grain1.v`
`SC_Grain128.v`
`SC_Trivium.v`