Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-MS-2010-08

M.Sc. Thesis

Design of a Crypto Core for Securing Intra System-on-Chip Communication

Martijn Verschoor

Abstract

Interconnect centric security in multi core System-on-Chip (SoC) is an area of increasing concern. Monitoring and manipulation of the SoC interconnect yields great potential to bypass higher level security mechanisms.

This thesis proposes SoC-TLS: a cryptographic hardware solution aimed to protect intra SoC communication against malicious IP and side channel attacks by providing secure - confidential, integer and authenticated - communication channels at the transport layer level of the SoC interconnect.

SoC-TLS enhances interconnect (bus/Network-on-Chip) interfaces with a hard crypto core containing enciphering/deciphering and keyed hash functions. This thesis particularly focusses on the design and implementation of an appropriate crypto core to mitigate specific threats concerning intra SoC communication.

A lightweight stream cipher and keyed hash function are proposed after a survey of existing stream ciphers and pseudo random number generators. The proposed functions are analyzed to determine their cryptographic strength.

Two reference designs are implemented; a single channel design for point-to-point communication and a multichannel design that supports secure connections between 15 secure cores. Implementation results for area and throughput are related to results for Æthereal NoC NI to determine the effects of hypothetical integration.

Design of a Crypto Core for Securing Intra System-on-Chip Communication

Thesis

submitted in partial fulfillment (40 ECTS) of the
requirements for the degree of

Master of Science

in

Embedded Systems

by

Martijn Verschoor
born in Den Helder, The Netherlands

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Design of a Crypto Core for Securing Intra System-on-Chip Communication"** by **Martijn Verschoor** in partial fulfillment (40 ECTS) of the requirements for the degree of **Master of Science** in **Embedded Systems**.

Dated: October 28, 2010

Chairman: _____
prof.dr.ir. A.J. van der Veen

Advisor: _____
dr.ir. T.G.R. van Leuken

Committee Member: _____
prof.dr. K.G.W. Goossens

Abstract

Interconnect centric security in multi core System-on-Chip (SoC) is an area of increasing concern. Monitoring and manipulation of the SoC interconnect yields great potential to bypass higher level security mechanisms.

This thesis proposes SoC-TLS: a cryptographic hardware solution aimed to protect intra SoC communication against malicious IP and side channel attacks by providing secure - confidential, integer and authenticated - communication channels at the transport layer level of the SoC interconnect.

SoC-TLS enhances interconnect (bus/Network-on-Chip) interfaces with a hard crypto core containing enciphering/deciphering and keyed hash functions. This thesis particularly focusses on the design and implementation of an appropriate crypto core to mitigate specific threats concerning intra SoC communication.

A lightweight stream cipher and keyed hash function are proposed after a survey of existing stream ciphers and pseudo random number generators. The proposed functions are analyzed to determine their cryptographic strength.

Two reference designs are implemented; a single channel design for point-to-point communication and a multichannel design that supports secure connections between 15 secure cores. Implementation results for area and throughput are related to results for Æthereal NoC NI to determine the effects of hypothetical integration.

Contents

List of Figures

List of Tables

1 Introduction

Security in systems-on-chip (SoC) is an area of increasing concern. Electronic devices process, store and communicate an increasing amount of confidential data. Take a modern smart phone as example, it can be used for financial transactions and online banking, storing digital signatures, certificates and private key material. Furthermore, the phone can store private data like pictures, personal videos or corporate confidential text-messages. In this era it has become evident that confidential data needs protection to avoid exposure. What if a modified SoC succeeds to covertly communicate extracted confidential information over a wireless channel?

Digital rights managements (DRM) - enforcing copyright protection of digital media - is another example that may concern the same mobile phone. The odd thing in DRM is that the subjected media requires protection against the user, the owner of the mobile phone. Music or video is sent to the device in encrypted form. A decryption key is required to allow playback of the media, so this too is sent to the same mobile phone. A few concerns arise: the decryption key could be extracted, the decrypted media could be copied or the mechanism that enforces the number of allowed playbacks could be manipulated.

Maliciously modifying SoCs can be quite lucrative when the SoC application is mass produced, like smart phones, or when it handles top secret information as is the case with military communication equipment. A thrilling example is reported in the article 'The hunt for the kill switch' [1] about the mysterious failure of a Syrian radar installation at the very moment of an Israeli attack on a Syrian nuclear installation. The radar was allegedly turned off by sending a secret code to a modified custom of-the-shelf microprocessor with a hidden 'back door'.

A SoC comprises many cores like processors, RAM, memory controllers, network controllers, audio codecs, video processors, peripherals etc. all on a single chip. Cores communicate over the on-chip interconnect that is traditionally point-to-point, cross-bar or bus-based - much like the conventional parallel communication buses that connect components at board level. Frequently used protocols are AMBA [2], IBM CoreConnect [45] or the OpenCores Wishbone [42].

Network-on-Chip (NoC) is an alternative paradigm first proposed by Dally [9]. Cores communicate by routing packets over a network, much like the Internet. NoC is promising for large scale SoC comprising tens to hundreds of cores. NoC have been subject to much research in the last decade, well known proposals are Æthereal [22] and Xpipes [4].

System complexity, shorter time to market and lower NRE cost are drivers for incorporating third party intellectual property (IP) into custom designs. Trust in third party IP is hard to establish but when IP is malicious it can severely affect the security

functions of a SoC. Malicious IP is a threat within the SoC, much like the concept of a Trojan horse in software. Integrally trusting the SoC will no longer suffice for future designs.

Existing on-chip interconnects are insufficiently protected against malicious IP and side channel attacks. Malicious IP is especially harmful since it slumbers in every system instance and can potentially be remotely operated. The on-chip interconnect yields great potential for hostile intents since it serves as the communication backbone of the SoC. Research that considers malicious IP exists [28, 25] but does not treat the vulnerability of the interconnect. Other research considers the interconnect but mainly focuses on detecting malicious software (this is treated in greater detail in section 3.1). The area of protecting the interconnect against malicious IP remains largely unexplored. SoC-TLS provides a new, orthogonal and complementing approach for interconnect security.

1.1 SoC-TLS

This thesis envisions SoC-TLS; a cryptographic mechanism for transport layer security (TLS[1]) over existing on-chip interconnects with the aim to approach the bandwidth of the underling interconnect. SoC-TLS provides confidential, integer and authenticated end-to-end communication through secure channels between cores. A secure channel can be seen as a virtual tunnel that connects two cores, much like a VPN connection. Other cores on the same interconnect can not eavesdrop, covertly alter or inject messages.

The bus or network interface (NI[2]) of cores that require secure communication are augmented with the SoC-TLS mechanism, see figure 1.1 for a NoC example. Secure transaction data is encapsulated in packets that are encrypted and appended with a Message Authentication Code (MAC). The receiver will verify the MAC code and decrypt the packet to retrieve the original data.

As implied by its name, SoC-TLS is embedded in the interconnect's transport layer. Interconnect protocols are subdivided into independent layers corresponding to the OSI model, see table 1.1. Each layer provides a service to the layer on top and can be implemented or changed independently. Positioning the cryptographic interconnect security mechanism in this layer yields end-to-end security transparent to the application layer and independent of the underlying interconnect.

1.2 Objective and motivation

High throughput transport layer security in on-chip communication depends on suitable crypto algorithms for encryption and message authentication. Cryptography provides a thorough measure against many interconnect centric threats, as described in chapter 2, but is often believed to be too bulky for large scale integration in SoC. Existing algorithms for encryption and message authentication consider a far more violent threat

[1]Not to be confused with the conceptually similar but functionally and cryptographically unrelated TLS - the successor of SSL - for end-to-end security over Internet (IP).

[2]This thesis will refer to both as network interface (NI)

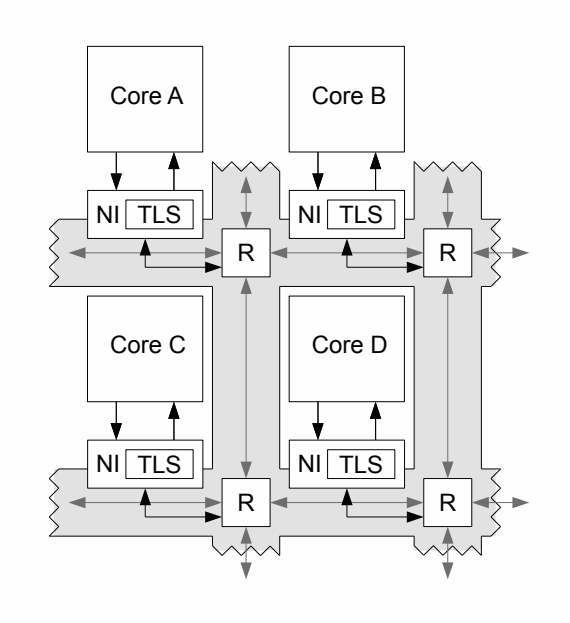


Figure 1.1: NoC with SoC-TLS embedded in each NI

5	Application	Message passing, shared memory
4	Transport	End-to-end connections and reliability, flow control, retransmission
3	Network	Path determination, routing/broadcasting, multi-access arbitration
2	Link	Link properties, e.g. width, synchronous/asynchronous
1	Physical	Physical properties, e.g. single/dual rail, precharge/conventional

Table 1.1: Five layer OSI model

model than is applicable for on-chip communication. Appendix A summarizes resource utilization and performance results of optimized implementations of well known algorithms for network security; they are indeed to bulky for large scale integration. It seems imperative that the design for SoC-TLS starts with a quest for algorithms that are tailored for the threat model of on-chip communication. The main objective for this thesis is to design a suitable crypto core that will jump start the envisioned SoC-TLS. Work like this has, to our knowledge, not been done before.

1.3 Cryptography

Cryptography (or cryptology) is the practice and study of hiding information. It has a rich and interesting history dating back to ancient times. A good introduction can be found in [11, 44]. Note that the glossary contains a short description of common crypto terminology. This section will briefly introduce the two most important crypto concepts for this thesis work; stream ciphers and message authentication codes.

1.3.1 Stream cipher

The stream cipher class of symmetrical crypto algorithms is based on the proven cryptographically secure algorithm known as 'one-time-pad' [44]. Both sender and receiver XOR subsequent characters of a message with a symmetric stream of random numbers. The one-time-pad uses a non-repeating sequence of true random numbers transported via a different channel, e.g. on a tape or DVD. An alternative is to synthesize a random number sequence using a pseudo random number generator (PRNG) that is seeded with a key, see figure 1.2. Sender and receiver can generate the same sequence only when they share the same key, thus only the key needs to be exchanged securely. Stream ciphers are good candidates for the crypto algorithm in SoC-TLS [18].

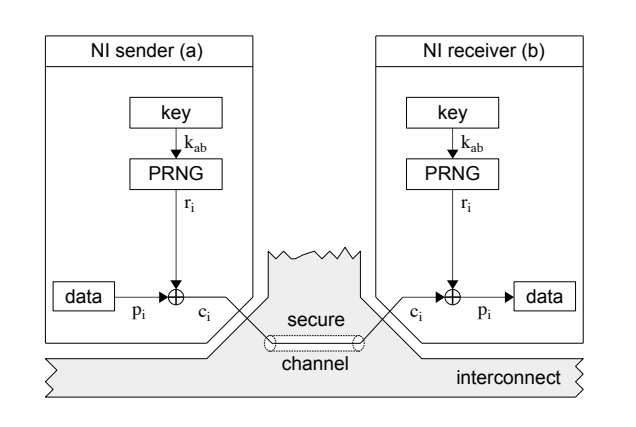


Figure 1.2: Stream cipher concept

The random sequence used for the transformation is required to be cryptographically secure, i.e. highly unpredictable. Unfortunately, most candidate PRNGs have a linear recurrence relation and are thus not cryptographically secure. One common solution is to add a nonlinear filter function that decorrelates the output from the internal linear PRNG sequence. This and other background information is discussed in greater detail in chapter 3.

1.3.2 Message Authentication Code

The purpose of a Message Authentication Code (MAC) is to determine the integrity of a message and authenticate the sender, i.e. the MAC provides a guarantee that the message did not change and was send by the indicated sender. It's aim should not be confused with that of a digital signature which additionally serves to counter a denial that the message was send (non-repudiation).

The MAC is computed with a key (dependent) hash function. The key is shared only by sender and receiver. Only they can compute the MAC corresponding to a message by making the same symmetric computations[3].

[3]Note that non-repudiation would require asymmetry by definition.

1.4 High level requirements

The design context of SoC-TLS is shaped by the desired level of performance, cost and security. These three dimensions are interrelated. For example, increasing performance will come at more cost or less security. The following design drivers express the desired balance between performance, cost and security.

High throughput and low latency Maximizing throughput supports transparent and minimally restricted use of secure channels. For the same reasons, latency should be kept to a minimum. The aim is to provide a secure channel at nearly the natural bandwidth of the underlying interconnect. Typical interconnects are 32 to 64 bits wide operating at frequencies between 200 MHz and 800 MHz yielding throughputs of 6 - 52 Gbps.

Low cost The required resources should be kept to a minimum for practical reasons. SoC-TLS requires two crypto cores in every secure NI. Only a minimal increase in overhead is acceptable.

Reasonable cryptographic security Cryptographic security should be strong enough to mitigate applicable threats concerning the on-chip interconnect. Stronger cryptography then necessary would harm the performance and cost objective.

1.5 Scope and research questions

This thesis will focus on the design and implementation of a crypto core that enables SoC-TLS. The cryptographic strength of the design is analysed and the synthesis and simulation results are reported. Integration of the crypto core into the NI is left for future work.

The goals for this thesis are expressed as the following research questions. The conclusion in chapter 7 states the answer to these questions.

RQ-1 What is a suitable threat model for attacks on the interconnect?

- **a** Which threats are mitigated by SoC-TLS?
- **b** Which threats remain or are newly introduced?

RQ-2 The design of the stream cipher and keyed hash function.

- **a** Which PRNG is most suitable for implementing the multi channel PRNG in terms of cost (resources), performance (speed, latency) and statistical properties?
- **b** Which nonlinear filter function is most suitable?
- **c** What would be an appropriate key management protocol and message encryption protocol?

RQ-3 What is the mathematical strength of the keyed hash function, e.g. what is the change of a collision?

RQ-4 What is the mathematical strength of the crypto function, e.g. what is the order of the key space (2^n)?

RQ-5 What is the cost (resources) and performance (speed, latency) of the total solution? And how does this relate to other work?

1.6 Thesis organization

This thesis is organized as follows. Chapter 2 states the interconnect threat model that is considered for the design of SoC-TLS (RQ-1). A summary of related work on interconnect security aspects and background information on stream ciphers is given in chapter 3. The design proposal for SoC-TLS and choice for cryptographic algorithms is presented in chapter 4 (RQ-2). Then, chapter 5 analyses the strength of the proposed cryptographic algorithms (RQ-3,4). Hardware design of core functions and accompanying synthesis and verification results are presented in chapter 6 (RQ-5). Finally, chapter 7 reflects on the accomplishments of this thesis and contemplates about future research.

1.6.1 Confidential supplement

This thesis is completed with a confidential supplement. Chapters 1, 2, 3 and appendix A are public and can be found in this text. Chapters 4, 5, 6 and appendixes B and C are completely confidential. The conclusion in chapter 7 is partly confidential; it is printed in both the confidential supplement and this thesis. The latter is censored for classified information.

Both this thesis and the confidential supplement are fitted with a table of contents, list of figures, list of tables, bibliography, nomenclature and glossary. Those found in the confidential supplement are complete, those found in this text are censored.

Threat model for SoC interconnect

2

The threat model presented in this section provides a clear delineation of treats that are mitigated by SoC-TLS and those that are not. It serves to formally justify the objective and scope of this thesis work. Since the objective of this thesis is to secure interconnect communications, this model will focus only on security threats that involve the interconnect.

The SoC interconnect is considered to be either bus or NoC. Most threats on bus interconnects have analog threats in NoC and vice versa. The resulting threat model has a generic nature and is augmented with specific bus/NoC examples when applicable. A threat model for a complete system would normally include a valuation for every asset and an estimation for the level of each threat. This final step is left for system designers that use SoC-TLS since the valuation of assets and level of threat are system-specific.

The first four sections of this chapter classify assets, threats, vulnerabilities and attacks. Section 2.5 maps the identified threats to the objectives of SoC-TLS. Section 2.6 reports residual vulnerabilities that are not treated by the objectives of SoC-TLS or that are newly introduced.

2.1 Assets

The prime asset - for the scope and objective of this thesis - is sensitive data communicated over the interconnect. What type of information this is depends on the system characteristics and architecture. Examples are cryptographic key material, classified plaintext, digital certificates, signatures, PIN codes from keypads, sensitive execution code or register settings. A SoC relies on the interconnect to transport sensitive data between IP cores. There are four general security concerns related to that transport, these are described below. (Italic definitions are cited from ISO/IEC-17799 [26])

Confidentiality *Ensuring that information is accessible only to those authorized to have access.* Keeping sensitive data private between communicating parties. A sensitive message is intended for the receiver, other cores should not be able to gain knowledge about the message.

Integrity *Safeguarding the accuracy and completeness of information and processing methods.* A message should not covertly be altered or lost during transmission. The message received is identical to the message send.

Availability *Ensuring that authorized users have access to information and associated assets when required.* Concerns the availability of the interconnect function as required.

Authenticity The identity of the sender is truly who the receiver beliefs him to be.

These security concerns have a general nature and are the pillars for information security. They are most suitable for generically identifying security concerns. We state these concerns as the valuable assets that should hold to guarantee secure communication over the interconnect.

2.2 Threats

A threat is an event that - when it occurs - leads to security breach of assets. This section identifies threats that involve the interconnect directly. Gains from security breach in the interconnect can be used as a stepping stone in other - system wide - attacks. Threats can be identified in the same abstract manner as the assets. The clearly distinguished assets identified above each have a related threat:

Extraction of secret information Threat to confidentiality. An attacker succeeds to extract secret information communicated over the interconnect, either explicit or implicit. In the latter case the attacker gains knowledge that decreases the uncertainty about a secret. An example of implicit information extraction is learning the hamming weight of a cryptographic key that can reduce the number of trials in an exhaustive, brute force, key search. An example of explicit information extraction is a malicious IP core that snoops the interconnect.

Modifying messages Threat to integrity. An attacker succeeds to covertly change a message that is send between two parties over the interconnect. The aim of the attack is changing the system state specifically or randomly, in the latter case just to cause some undetermined harm. The change of state may be the desired end result of an attack or can be used as a stepping stone for further attacks.

Spoofing messages Threat to authenticity. An attacker succeeds to inject a message into the interconnect that appears to be from a trusted core. The message is treated with the privileges of the spoofed sender yielding opportunities to read or modify assets protected by memory access rights.

Degrading performance Threat to availability. An adversary succeeds to attack the availability of the interconnect. The result is a degradation of performance, possibly including critical functions. Attacks aimed to degrade system performance are known as Denial of Service (DoS) attacks.

2.3 Vulnerabilities

A vulnerability can be exploited to perform an attack. Threats become reality through vulnerabilities in the system.

One observation about existing interconnects in relation to the threats identified above is readily made; all interconnects transmit data plain, most interconnects do not provide integrity checks and most interconnects can not authenticate a message, see related work in section 3.1. Not providing these assertions makes the interconnect vulnerable for the identified threats.

At this point we have arrived at the core justification for this thesis. What remains is to show how an attacker can gain access to the interconnect, this is reported in the next section.

2.4 Attacks

The development of attacks and corresponding countermeasures is an arms race where hackers continually challenge novel security mechanisms. It is almost infeasible to exhaustively identify concrete attacks. Instead we will classify attacks according to the way an attacker accesses the interconnect.

Malicious IP and hardware Trojans An IP core connected to the interconnect potentially has full read and write access to the interconnect. In bus based interconnects it is near trivial to monitor transactions on the bus giving ample opportunity for malicious IP to snoop for sensitive data [25]. Messages can be modified by causing contention on specific bus lines at the right time. An example for NoC is a malicious IP that injects packets with fake identity abusing falsely acquired memory access rights [13, 14, 17].

This class of attack can be considered most dangerous because the malicious IP slumbers in every instance of the product and can (and is likely to) be able to connect to a remote location to pass snooped secret data or receive commands for attacks.

Side channel attacks/analysis The adversary tries to deduct sensitive information by externally monitoring emissions from the interconnect. Data transmissions on the interconnect leave traces in power, EM and RF spectrum due to rapid changes of voltage on the wires that comprise the interconnect. Typical interconnects are sensitive to side channel analysis especially because the emissions from the interconnect are fiercer compared to emissions from a processor [19, 3]. In a side channel attack the adversary tries to change the state of the system through manipulation of the systems environment. Examples are power glitching, EM and RF attacks. This class of attack requires proximity access.

Physical attacks This class of attack concerns an attacker that succeeds to gain physical access to the chip die of a system that remains fully functional. The attacker can theoretically do anything to the entire system by probing signals on the chip: eavesdropping, changing system state or reverse engineering. Although a physical attack can be targeted at the interconnect, attempts to prevent possible effects through logical or cryptographic measures on the interconnect provide merely a hurdle; the attacker can also target IP cores, memory and protection mechanisms directly.

2.5 SoC-TLS security objectives

The objective of any security measure is to mitigate threats by treating vulnerabilities. The objective for SoC-TLS is to treat the vulnerabilities stated in section 2.3 by taking

the following two measures:

Encryption SoC-TLS enables setup of secure channels between pairs of IP cores providing end-to-end encryption. Messages are encrypted at the NI of the sender and decrypted at the NI of the receiver. Each secure channel operates with a unique key-pair for encrypting messages in both directions. The secure channel functions like an isolating tunnel that privately connects two IP cores preventing eavesdropping by other parties on the interconnect.

Message Authentication Code Encrypted messages on secure channels are appended with a MAC that only the true sender and receiver on that specific secure channel can produce. The MAC is unique[1] for every message and proves to the receiver both the authenticity and integrity of the message.

The cryptographic strength of the required protocols should be *good enough* which means they should provide a sufficient barrier for the level of threat but remain practical for large scale integration. We assume a different intelligence context for intra SoC communication than for typical cryptographic solutions and therefore believe that simpler algorithms will suffice. Cryptanalysis always depends on gathering huge amounts of data; ciphertext only or ciphertext with corresponding plaintext. The amount of processing time and memory space required makes on-chip cryptanalysis practically infeasible, even for relatively short cryptographic key lengths. Off-chip cryptanalysis would require a covert channel that leaks sufficient - preferably - continuous ciphertext. Although off-chip cryptanalysis is possible, there are two barriers: 1) the high bandwidth of SoC-TLS makes covertly streaming continuous ciphertext infeasible and 2) such a covert channel is likely to be noticed. We therefore believe that simple crypto algorithms will suffice.

2.6 Residual vulnerabilities

The objectives of SoC-TLS treat the confidentiality, integrity and authenticity vulnerabilities of a SoC interconnect. This section identifies other vulnerabilities involving the interconnect that are not treated by SoC-TLS, or that are introduced by SoC-TLS.

Availability Attacks with the aim to degrade system availability can not be treated by SoC-TLS alone. The integrity checking function of SoC-TLS can aid in detecting availability attacks by reporting repeated integrity failure of components to some security authority on the SoC. A notable drawback of the integrity checking mechanism is that it can be used as a means to degrade system availability. Resolving corrupted messages comes at the cost of retransmissions and resynchronization, putting a higher strain on the available bandwidth (this is treated in chapter 4 in the confidential supplement). Without the mentioned reporting mechanism an attacker may succeed to degrade system availability simply by corrupting messages randomly.

[1]The MAC is not truly unique, but the probability of collision is highly unlikely, see chapter 5 in the confidential supplement

Valid transactions Attacks that use the interconnect in its valid function can not be prevented by SoC-TLS. An example is abusing access rights in a memory mapped system. Malware and viruses fall into this category. This type of attack could be prevented by enforcing memory access policy at the interconnect as suggested in [8, 14].

Extracting metadata Every transaction on the interconnect includes plain metadata that can reveal information about the execution of sensitive processes. SoC-TLS will hide many details of the transaction, see chapter 4 in the confidential supplement. The existence of a transaction, the initiator and the target NI identity and the length of the message are not hidden by SoC-TLS. The extraction of metadata can be treated by padding messages to a fixed length and/or inserting random dummy messages.

Attacks aimed at SoC-TLS Unfortunately, often security measures introduce new security vulnerabilities; this also holds for SoC-TLS. An analysis of the completed system should determine what concrete new vulnerabilities are introduced. Attacks directly aimed at SoC-TLS are:

Cryptanalysis An attacker may try to break the security function of SoC-TLS by performing cryptanalysis on the crypto algorithm. Chapter 5 is designated to the analysis of the crypto algorithm strength.

Extraction of SoC-TLS keys Secrecy of key-pairs is the prime asset for the security function provided by SoC-TLS. Keys are vulnerable during storage and transport. This vulnerability is related to the topic of key management described in chapter 4 in the confidential supplement.

Bypass A different approach of attack is to covertly bypass the SoC-TLS mechanism for specific channels causing messages to be transmitted plain over the interconnect. The SoC-TLS provides a security service transparent to the IP cores. Bypassing messages could go undetected unless some additional mechanism checks the correct functioning of the SoC-TLS mechanism. Such mechanism is desirable for SoC-TLS but a detailed description or proposal for such a mechanism lies beyond the scope of this thesis.

2.7 Conclusion

This chapter presented the formal threat model for SoC interconnect by exploring assets, threats, vulnerabilities and attacks concerning SoC interconnect and describing how threats are mitigated by SoC-TLS. The encryption and MAC algorithms provided by SoC-TLS can prevent extraction of information, modifying messages and spoofing messages. Attacks aimed at degrading performance can not be treated by SoC-TLS alone, but the integrity checking mechanism of SoC-TLS does provide a means to detect redundant unauthorized messages.

The remainder of this thesis will investigate feasibility, design and implementation of the SoC-TLS crypto core, starting with a description of related work and a quest for suitable algorithms in the next chapter.

3 Background

This chapter provides background information related to this thesis work. We start with an overview of related work in section 3.1. Background information on stream cipher design is presented in section 3.2.

3.1 Related work

This section describes related work on security aspects involving the on-chip interconnect found in literature. At the end of this section we discuss the related work and compare it to this thesis work.

3.1.1 Interconnect security

Benini et al. A method for energy efficient bus scrambling is proposed in [3]. The objective is to prevent differential power analysis (DPA) of the on-chip bus by XORing data with a random value for transmission. A scrambled transmission contains statistically random words. DPA becomes infeasible without information about the random sequence.

Scrambling has a negative effect on the power consumption compared to - usually more correlated - plain data. The authors make two proposals to increase power efficiency: 1) scrambling is made conditional; it is switched off for non-sensitive transactions, 2) individual scrambled transmission words can be inverted to minimize the hamming distance between two successive words.

The scrambling mechanism has similarities to this work, but - as will be discussed in section 3.1.2 - does not provide protection against malicious IP.

Gebotys One of the earliest works on security in NoC comes from Gebotys [19] who proposes a cryptographic framework for secure storage and transportation of application key material in NoC based SoC. The work distinguishes secure cores that implement security or cryptographic functions like AES or SHA and use or handle application key material, and non-secure cores that do not need to know about cryptographic keys. Application key material is stored in a central (secure) key keeper core and is never transmitted plain. The NI of every secure core is augmented with a security wrapper that can encrypt and decrypt messages with the network key k_n - *shared by all cores*. Upon request of a stored application key by a secure core, the key keeper will encrypt the requested key with the network key k_n and send it over the interconnect. The receiving secure core decrypts the message upon reception with the same network key k_n to derive the requested

application key. Application keys can be stored or updated in the key keeper by transporting them in the same manner.

The original work described above contains some cryptographic deficiencies that the authors solve in an update of their work [20]. The shared network key k_n is renamed master network key and is no longer used for encryption directly to avoid reuse of the same key[1]. The master network key can be updated by the key keeper core. Secure network messages are now encrypted with a deviation key k'_n computed from the master network key and a message counter using a hash function. Furthermore, the authors add a MAC and integrity algorithm and propose simple - one bit - LFSR schemes to implement the crypto algorithms.

Evain and Diquet Evain and Diquet identify specific NoC security attacks and suggest solutions in [13]. They identify four attack types. 1) Denial of Service (DoS), 2) Extraction of secret information, 3) Hijacking the system, 4) Reverse engineering and extraction of secret information by proximity access. These attacks are considered and described in more detail in the threat model in section 2.4.

DoS attacks are further classified into: a) Incorrect path; sending a packet over an erroneous path with the aim of trapping it into a dead-end thereby blocking channels for valid packets. b) Deadlock; this attack is engineered to disregard deadlock free rules of the routing technique with the intent to cause a deadlock. c) Livelock; introducing a packet that is engineered to never reach his destination but instead keeps cycling the network indefinitely.

The authors propose to split secure and non-secure communications over two virtual channels (VC) as a general measure against bandwidth attacks. Prioritizing secure VC mitigates the effects of a DoS attack from non-secure cores and thereby protects critical system functions. A similar discrimination can be made using guaranteed services (GS) for secure and best effort (BE) for non-secure communications, as offered in Æthereal [22].

The work of Evain and Diquet considers cases where not all NoC routers and NIs are trusted. If an untrusted IP is connected via a trusted NIs then a memory mapped authorization mechanism similar to the APU of SECA (described next) is applicable. In case NIs or routers are untrusted, path information can not be trusted either which raises an authorization concern. A path filter at the trust boundary can authorize transactions or filter out illegal packets, preventing illegal access and DoS attacks with constructed messages (incorrect path, deadlock, livelock).

The concept of Self Complemented Path coding (SCP) is introduced as a means for authenticating senders. In principle the traversed path of a packet should uniquely identify the sender. The proposed path coding ensures that path information in a delivered packet resembles the actual traversed path, thereby authenticating its sender.

The described solutions are unique for NoC and the authors reason in [12] that a security enhanced NoC is a good basis for secure SoC.

[1] A sin - really - especially with a stream cipher implementation.

Coburn et al. The Security-Enhanced Communication Architecture (SECA) by Coburn et al. [8] aims to prevent software attacks that abuse falsely acquired access privileges - e.g. buffer overflow exploits - by monitoring on-chip bus transactions. The security mechanism can be retrofitted on existing on-chip buses and was implemented on AMBA for a mobile telephone SoC application.

The proposed architecture comprises a central Security Enforcement Module (SEM) and distributed Security Enforcement Interfaces (SEI) implemented locally in slave bus interfaces. Together, they implement three protection schemes:

Address-based Protection Unit (APU) enforces memory access rules that specify how a bus master can access a slave device (read-only, write-only, read-write, no access) for a particular operating context. These rules are stored in ternary content addressable memory (TCAM) in the central SEM. The memory mapped address space is divided into blocks whose size is a reciprocal of the number of lines in the TCAM table. An alarm is raised when an IP wants to access a memory location in a way that exceeds his privileges for the current context.

Data-based Protection Unit (DPU) enforces valid configuration of peripheral devices for a particular operating context. The SEI, local at each peripheral, contains a table with valid settings for configuration registers per operating mode. The table is consulted for approving write operations to configuration registers.

Sequence-based Protection Unit (SPU) implements a security automata that detects anomalies in critical execution sequences. One example of such a sequence for digital rights management (DRM) is decrementing a playback counter. The security automaton is setup when the user selects a DRM controlled song and raises an alarm on a deviation in bus cycles required to decrement the counter after playback.

Fiorin et al. More recent work on NoC security comes from Fiorin et al. In [17] the authors give an overview of security aspects in NoC and propose the following solutions: 1) an Address Protection Unit (APU) for memory mapped authorization, 2) Quality of Service (QoS) techniques to mitigate bandwidth attacks, 3) Security automata to detect anomalous transactions and 4) Intrusion detection techniques to detect anomalous system behaviour.

In [15] the authors proceed with an implementation of the APU[2] and present results of two implementation models integrated in either the target NI or initiator NI. A network security manager (NSM) providing runtime reconfigurability is added in [14]. The resulting memory mapped authentication mechanism is functionally similar to the described APU in SECA and provides protection against software attacks that abuse falsely acquired access privileges such as viruses and Malware.

[2]Which - for unmentioned reasons - is renamed Data Protection Unit, but remains similar in concept to the APU of SECA. This text will remain to use the APU term to avoid confusion.

Intrusion Detection uses monitoring probes that measure some indicator of security violations. Suggested indicators are: buffer occupancy, anomalous behavior of the power manager, unauthorized access to secure memory locations or execution violation of critical routines. An example is the detection of DoS attacks by indication of buffer occupancy. An implementation proposal and results for intrusion detection are described in [16].

3.1.2 Discussion and comparison to this work

The objective of the scrambling mechanism by Benini [3] is to prevent DPA attacks. It does not provide protection against malicious IP because it is not (designed to be) cryptographically secure. The effects on power consumption and ideas for improving power efficiency equally applicable to this work.

The cryptographic framework by Gebotys [19, 20] is closely related to this work but there are some important differences. Gebotys only secures specific and rare transactions for key transportation purposes. The suggested (bitwise) hardware implementation is not sufficient to setup a secure channel at near full bandwidth speeds. Furthermore, the described derivation of network keys does not seem feasible - nor complete - since it does not allow encrypted transactions between secure cores other then the key keeper. All secure cores share the same master key, which is inadvisable. No implementation or synthesis results are presented.

The routing algorithm by Evain [13] provides sender authentication, a purpose shared with this work. Furthermore, Evain does not assume all routers and NI trusted, thereby treating some concerns raised by malicious hardware.

Coburn [8] and Fiorin [17, 15, 14] proposed similar ideas for preventing software attacks that are manifested on bus or NoC respectively. Their work is orthogonal and complementary to this work.

Table 3.1 gives an overview of the described work.

Work	Focus	Protection against			Concrete design
		Malware	Malicious IP	Side channel	
SECA	Bus	√	×	×	√
Fiorin	NoC	√	×	×	√
Evain, Gebotys	NoC	√	√	×	×
Benini	Bus	×	×	√	√
Gebotys	NoC	×	√	√	×
This work	Bus/NoC	×	√	√	√

Table 3.1: Overview of related work

3.2 Stream ciphers

The proposed crypto function is a stream cipher. The concept of a stream cipher is illustrated in figure 3.1, it operates as follows. Both sender and receiver generate the same pseudo random sequence r with their build in pseudo random number generator (PRNG). The sender XORs plaintext message p with pseudo random sequence (PRS) r and transmits the resulting ciphertext c. The receiver XORs the received ciphertext c with his identical PRS r which reveals the original plaintext message p. The state of the PRNG is the shared key that should be guarded from adversaries. A key management protocol (KMP) takes care of synchronization of key-pairs between sender and receiver.

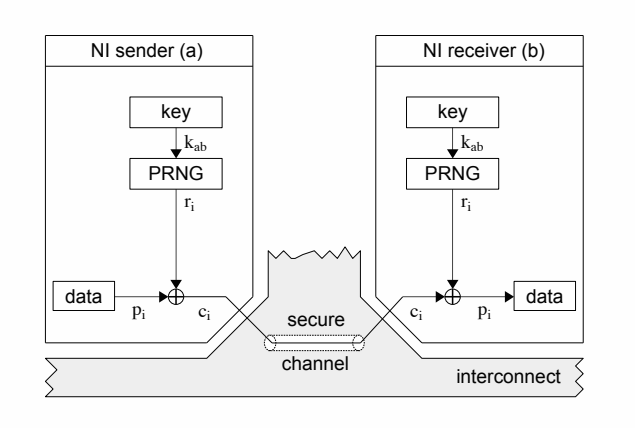


Figure 3.1: Stream cipher concept

It is essential that the generated PR sequence is unpredictable i.e. cryptographically secure. Knowledge about previous numbers in the sequence should not help to predict future numbers. Some PRNGs are cryptographically secure by nature, they are specifically designed to generate an unpredictable random sequence. However, these generators are generally slow. Another approach is to append a nonlinear filter function to a predictable generator with the aim to de-correlate the output sequence from the internal state.

Background information in this section aids in making design decisions for the proposal in chapter 4. Section 3.2.1 describes secure and non-secure candidate PRNGs. Section 3.2.2 presents some popular test suites to assess the randomness of generators. Methods to harden the security of predictable PRNGs are discussed in section 3.2.3. Some standard methods for cryptanalysis are summarized in section 3.2.4. The selection of PRNG and filter function is deferred to the design proposal in chapter 4.

3.2.1 Pseudo Random Generators

A lot of literature is written on the subject of PRNGs. This section presents a selection of background information on PRNGs that helps to understand and justify the design proposal in chapter 4. Each of the following subsections describes a candidate (class of)

PRNG and relates their properties to the design objectives. A summary of the most relevant properties of the discussed PRNGs is provided at the end of this section.

3.2.1.1 Linear Congruential Generators

Linear congruential generators (LCG) form the well known class of PRNGs introduced by D.H. Lehmer in 1949 [30]. LCGs are defined by the recurrence relation:

$$X_n = (aX_{n-1} + b) \mod m, \quad n > 0$$

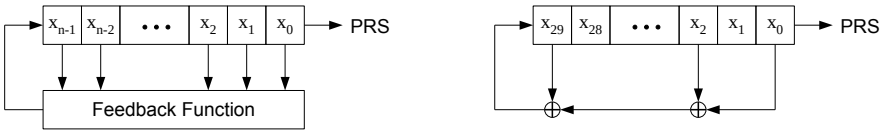
in which X_n is the sequence of pseudo random numbers, seeded by setting X_0. Variables a, b and m are constants: a is called the multiplier, b the increment and m the modulus. The generator can reach a *maximal* period of $m-1$ when the constants are well chosen. The generator is seeded by setting the first value X_0.

The LCG class of PRNG is efficiently implemented in software, requiring one addition, one multiplication, one modulus operation and little memory. Implementation in hardware is less efficient compared to other generators due to the multiplication and modulus operation that are costly in both area and speed [37]. It is tempting to choose the modulo m as a power of 2 because this would reduce the modulo operation to an AND operation with $(m-1)$ yielding significant improvement in efficiency - especially for hardware implementations. However, from the case of RANDU, a generator with $m = 2^{31}$ that was widely used since the 1960's, we can learn that such generators yield poor statistical properties [23]. The output sequence of RANDU has a correlation in each three-tuple of consecutive numbers. As a result, if a three-tuple is considered a coordinate in three-dimensional space, the three-tuples would all fall into 15 two-dimensional planes [33].

Linear congruential generators are not cryptographically secure on their own [44]. An additional nonlinear filter function is required for application in a stream cipher.

3.2.1.2 Linear Feedback Shift Registers

Linear feedback shift registers form another well known class that is researched to great extend. In fact, many generators are somehow related to this class as we will show in following sections.



(a) Abstract FSR

(b) LFSR example

Figure 3.2: Feedback Shift Registers

A feedback shift register (FSR) is comprised by two parts: a shift register and a feedback function. The output from the shift register is the pseudo random sequence. The input to the shift register is a function of (some of) its cells, see figure 3.2(a).

The feedback function of the linear feedback shift register (LFSR) is a simple XOR of certain bits in the shift register. The bits that are XORed are called taps.

The linear feedback function allows mathematical analysis to reassure good statistical properties. An n-bit LFSR can be in $2^n - 1$ states. The generator can not recover from the zero state, which is therefore not allowed. The feedback function is expressed as a polynomial mod 2 with the bits in the shift register representing the coefficients, e.g.:

$$x^{n-1} + x^{n-2} + ... + x^2 + x^1 + 1$$

The maximal period of $2^n - 1$ can be reached only if the polynomial is primitive. As an example, the feedback function in figure 3.2(b) corresponds to the primitive polynomial $x^{29} + x^2 + 1$. A table with sparse primitive polynomials can be found in [44].

It is unsafe to use the output of an LFRS directly as input to a stream cipher due to the linear behavior of the feedback function. The last n bits of the PRS reveal the entire state that the register had n bits before. An adversary with knowledge of the feedback function can reproduce the entire sequence with knowledge about only n consecutive output bits. Hiding the feedback function does not provide additional security; the entire LFSR can be synthesized after examining a sequence of only $2n$ bits using the Berlekamp-Massey algorithm [38].

3.2.1.3 Lagged Fibonacci Generators

Lagged Fibonacci generators are a class of PRNGs inspired by the famous Fibonacci sequence: $0, 1, 1, 2, 3, 5, 8, 13, ...$ The Fibonacci sequence can be produced with the recurrence relation that taps the previous two values:

$$X_n = X_{n-1} + X_{n-2}$$

A lagged Fibonacci sequence taps two delayed (lagged) values:

$$X_n = X_{n-l} \otimes X_{n-k} \mod m, \quad 0 < l < k$$

where $\otimes$ can be any binary operation and m is usually a power of two, $m = 2^w$ where w denotes the word size. Common operations are addition, multiplication and XOR. LFGs that use an XOR operation are treated in the next section.

Lagged Fibonacci generators with the addition operation are called additive lagged Fibonacci generators (ALFG). They have a maximum period of $(2^k - 1) \cdot 2^{w-1} \approx 2^{w+k-1}$. To reach the maximum period the polynomial in GF(2):

$$x^k + x^l + 1$$

should be primitive. A table with values for k and l that make the above polynomial primitive can be found in the second volume of Knuth [29]. Likewise, multiplicative lagged Fibonacci generators (MLFG) use a multiplication operation and can reach a maximum period of $(2^k - 1) \cdot 2^{w-3} \approx 2^{w+k-3}$, one fourth of the ALFG.

A *maximal* period generator has a period that is as great as the number of possible states. The sequence is then strictly periodic and every possible k-tuple of X's will

appear in the full sequence, a desirable property for uniformity [35]. An LFG has a state space of 2^{wk} which far exceeds the stated maximum period for the ALFG and MLFG which are in the order of 2^{w+k-1} and 2^{w+k-3} respectively. MLFGs have better statistical properties then ALFGs due to the nonlinearity of MLFGs [34].

In general the LFG class generators are efficiently implemented in both software and hardware. The operations are simple and require only a few instruction in software. Addition is favorable over multiplication in a true hardware implementation due to the cost of the multiplication operation. Storing the state requires a considerable amount of memory (especially compared to the LCG type of generators). The two tap mechanism can efficiently be implemented using dual port memory. A compromise between period length and required storage space can be made by choosing appropriate parameters for l and k.

3.2.1.4 Generalized Feedback Shift Registers

The two tap generalized feedback shift register (GFSR) proposed by Lewis [31] is an LFG with an XOR operation.

$$X_n = X_{n-l} \oplus X_{n-k} \tag{3.1}$$

A GFSR with word width w can be seen as w independent parallel LFSRs with two taps. Implementation in both software and hardware is similar to the ALFG. The XOR operation does not require carry logic and may require less time than the add operation in a pure hardware implementation.

The statistical properties of the GFSR are similar to an LFSR with trinomial $x^k + x^l + 1$, which are known to have poor randomness [39]. The maximum period that can be achieved when parameters are properly chosen is $2^k - 1$, far less then the number of possible states: 2^{kw}. On top: GFSR randomness is very sensitive to initialization, requiring a careful selection of initial seeds [39].

3.2.1.5 Twisted GFSR

The maximum period of GFSR generators is only a small fraction of the number of possible states due to the independence of bit positions. This disadvantage is overcome by Matsumoto et al. [39, 40] who cancel this independence with a modification to equation 3.1:

$$X_n = X_{n-l} \oplus (X_{n-k} >> 1) \oplus \begin{cases} 0 & \text{if LSB of } X_{n-k} = 0 \\ a & \text{if LSB of } X_{n-k} = 1 \end{cases} \tag{3.2}$$

In which the variable a is a constant chosen to make the recurrence based on a primitive polynomial with many terms. See [39] for details about finding suitable a, l and k and a more detailed mathematical background.

Twisted GFSR (TGFSR) is a *maximal* period generator, e.g. the maximum period equals the theoretical upper bound of 2^{kw}. It has improved randomness and initialization is carefree. The additional algorithmic step in equation 3.2 requires marginal extra cost to implement in either hardware or software compared to the original GFSR in equation 3.2.

At first the authors claimed that the sequence is k-distributed, which is best possible. Later it was discovered that there was a defect in the k-distribution to v-bit accuracy [46]. To improve on k-distribution, Matsumoto refined his original idea by tempering (hardening) the output as follows:

$$Y_n = X_n \oplus (X_n << s) \wedge b$$
$$Z_n = Y_n \oplus (Y_n << t) \wedge c$$

where the variable X_n is the output from the original TGFSR, Z_n becomes the new output, Y_n is an intermediate variable, s and t indicate the number of bits to shift left and b and c are a mask [40]. Hardware implementation of the tempering mechanism comes at the cost of only two XOR stages because shifts and masks are hardwired. The improved TGFSR is referred to as Tempered TGFSR (TTGFSR).

3.2.1.6 Mersenne Twister

The Mersenne Twister (MT) is a variant of the TGFSR by Matsumoto et al. with an astronomically large period of $2^{19937} - 1$ [41]. It is a *maximal* period generator with its period chosen to be a Mersenne prime. At time of writing it is the default PRNG in mathematical software like Maple and Matlab and scripting languages Python and Perl.

There are different version of MT; MT19937 is the original 32-bit version, MT19937-64 is a 64-bit version, SFMT is a 128-bit SIMD version. Furthermore, there is support for various periods ranging from $2^{607} - 1$ to $2^{216091} - 1$.

The internal state is updated in a similar but slightly different way as equation 3.2 for TGFSR.

$$\begin{aligned} Q_n &= X_{n-k}(32)|X_{n-(k-1)}(31:0) \\ X_n &= X_{n-l} \oplus (Q_n >> 1) \oplus \begin{cases} 0 & \text{if LSB of } Q_n = 0 \\ a & \text{if LSB of } Q_n = 1 \end{cases} \end{aligned} \tag{3.3}$$

Instead of using the variable X_{n-k} directly, MT takes the MSB of X_{n-k} and the remaining less significant bits of $X_{n-(k-1)}$. MT also has a tempering mechanism similar to the TTGFSR, but with additional stages pre- and postfixed:

$$Y_n = X_n \oplus (X_n >> u)$$
$$Y'_n = Y_n \oplus ((Y_n << s) \wedge b)$$
$$Y''_n = Y'_n \oplus ((Y'_n << t) \wedge c)$$
$$Z_n = Y''_n \oplus (Y''_n >> l)$$

Implementing MT19937 requires a considerable amount of memory: 624 32-bit words, or 2496 bytes. MT can not be implemented with a single dual port memory since the recurrence relation depends on three previous values instead of two.

3.2.1.7 Blum Blum Shub

Blum Blum Shub (BBS) [5], named after its authors, is a cryptographically secure PRNG. Its recurrence relation is:

$$X_n = X_{n-1}^2 \mod m$$

where m is the product of two large primes p and q which are congruent to 3 mod 4. In contrast to the generators described above, X_n is not directly output as random sequence. Instead, the output takes only the parity or only a few least significant bits of X_n. The difficulty of factoring m is the pillar for its cryptographic security.

BBS is slow. The operations that comprise the generator are time consuming and only one (or at most a few) bits are output every cycle. Implementation of the multiplication and modulo arithmetic in hardware requires a lot of area. Therefore BBS is not suitable for our objective.

3.2.1.8 Block cipher PRNGs

Finally, to be complete we should also mention that many stream ciphers utilize a block cipher to generate a pseudo random sequence. Although these generators are cryptographically secure, implementations of known block ciphers require too much resources for our objective, see table A.

This concludes the survey of PRNGs. A comparison and selection is made for the design proposal in the next chapter.

3.2.2 Testing randomness

It is generally accepted that testing PRNGs is the most suitable way to assess their randomness. Tests generally compare a generated pseudo random sequence with the properties of true random sequences and express the probability that the pseudo random sequence was truly random. A test fails if it is dissimilar to true random, but, remarkably, also fails if it is too similar to true random.

Many tests for PRNGs have been devised. One famous test-suite is Diehard compiled by George Marsaglia [36]. This test suite inspired Robert G. Brown to create an open source version named Dieharder [6]. The NIST has developed a test suite specifically for cryptographically secure PRNGs [43].

3.2.3 Making PRNGs cryptographically secure

The generators that are suitable for the objective of SoC-TLS are not cryptographically secure by nature. The objective of cryptographic security is to prevent that the output sequence reveals information about the internal state or key material. There are different approaches to make a generator cryptographically secure, these are threated in the following subsections.

3.2.3.1 Hash function

A non-cryptographically secure pseudo random stream can be hashed with a secure hashing algorithm like SHA or MD5 to make it secure. This method produces good quality secure random based on the cryptographic properties of the underlying hashing algorithm.

Known secure hashing algorithms that operate at the desired performance require considerable amounts of resources in a hardware implementation. This method is therefore not adequate for the objectives of SoC-TLS.

3.2.3.2 Combination generators

One way to hide the internal state of the generator is to combine the output of multiple generators with some combining function. The result is a new, larger, generator called a combination generator whose state is the combined state of the underlying generators.

Many proposals for combination generators have been made, see [44] for an overview of some proposals for LFSRs. Some have been broken, others are not. We will list some of the general principles for illustration:

Multiplexer The output of internal generators is fed into a multiplexer that is controlled by one generator. The resulting output sequence is a random selection of the internal generators.

Stop-and-go One generator controls the clock of other generators. The output sequence can be a boolean function of the internal generators.

Threshold The combined output is a logic one only if the majority of internal generators is a logic one.

Combination generators may be suitable for SoC-TLS. They do require a more complex structure due to the multiple internal generators.

3.2.3.3 Nonlinear filter function

The non-cryptographically secure random sequence is transformed with a nonlinear filter function[3]. These filters perform an algebraic nonlinear function on the original non-secure random sequence.

3.2.4 Cryptanalysis

Cryptanalysis is the study of methods for deciphering without knowledge of the key, or reducing uncertainty about the key. This subsection lists a few well known methods for cryptanalysis.

[3]In fact, the above described hash function and combination generator also add nonlinearity, but here we refer to a filter with an algebraically nonlinear function.

Time Memory Trade-off This class of attack pre-computes ciphertext for a known or chosen plaintext (a part of) all keys and stores them in a massive memory [24]. The ciphertext sequence of known plaintext is looked up in memory reducing uncertainty about the key.

Berlekamp-Massey The linear complexity (LC) of a generator defines the shortest sequence of output that needs to be analyzed to synthesise an LSFR that mimics the function of the analysed generator. For linear generators the LC is $2n$, where n is the number of bits in the state register.

3.2.5 Summary

This section elaborated on various PRNG classes with the aim to select a suitable generator for the SoC-TLS stream cipher. Methods for de-correlating the output by non-linearization are described. Selection is deferred to the design proposal in the next chapter. Furthermore, methods to analyze the randomness of PRNGs are listed and basic cryptanalysis methods are shortly described.

4 SoC-TLS proposal

This chapter is classified as confidential. It can be found in the confidential supplement.

5 Cryptanalysis

This chapter is classified as confidential. It can be found in the confidential supplement.

Hardware design and implementation

6

This chapter is classified as confidential. It can be found in the confidential supplement.

7 Conclusion

Confidential information is censored from this public thesis. Please read the full conclusion in the confidential supplement when possible.

This thesis proposes SoC-TLS, a cryptographic solution for securing intra SoC communication against malicious IP and side channel attacks. SoC-TLS enhances interconnect interfaces with a hard crypto core containing enciphering/deciphering and keyed hash functions. SoC-TLS provides secure - confidential, integer and authenticated - communication channels in which messages are encrypted and appended with a message authentication code (MAC). This thesis proposed a suitable crypto core and focussed on its design, analysis and implementation.

7.1 Answers to research questions

The thesis goals were formulated as research questions in section 1.5. We now return to these questions to conclude and reflect on this work.

RQ-1 *What is a suitable threat model for attacks on the interconnect?* This thesis started with the description of a threat model for SoC interconnects in chapter 2. The threat model helps to identify specific security threats to the interconnect and serves as a formal problem definition. The model is used to determine which threats are mitigated by SoC-TLS and which are not.

The general security concerns of confidentiality, integrity, availability and authenticity are identified as the security assets of SoC interconnect. These concerns no longer hold when (a part of) the SoC is untrusted, as is the case with possible malicious IP. Each of these assets corresponds to a threat; extraction of secret information, modification of messages and spoofing messages are mitigated by SoC-TLS. Degradation of performance or denial of service (DoS) attacks are not mitigated by SoC-TLS but can be treated with other measures.

Four new treats are introduced by SoC-TLS. First, the SoC-TLS mechanism could be used as an instrument in a DoS attack unless abundant mismatching MACs are reported to some security manager core. Second, SoC-TLS does not prevent attacks that use the interconnect in its valid function, e.g. software attacks such as malware and viruses. Third, SoC-TLS can not prevent extraction of metadata, though this could be mitigated with additional measures. Fourth, the SoC-TLS mechanism itself could be attacked. For example through cryptanalysis, extraction of keys or by bypassing the crypto mechanism.

RQ-2 *The design of the stream cipher and keyed hash function.* The design or choice of crypto algorithms is vital for the feasibility of SoC-TLS. Many preferred algo-

rithms for encryption and message authentication are too bulky for efficient large scale implementation in interconnect NIs. They do not fulfill the high level requirements set in section 1.4: high throughput and low latency, low cost and reasonable cryptographic security. The cryptographic class of stream cipher algorithms is most suitable for enciphering messages. *The further answer to this research question is classified as confidential.*

RQ-2a *PRNG selection.* The choice of internal PRNG is vital for the quality and practicality of the crypto core. *The further answer to this research question is classified as confidential.*

RQ-2b *Filter implementation.* The filter hides the internal random sequence. *The further answer to this research question is classified as confidential.*

RQ-2 *Key management scheme.* Proper management of cryptographic keys is crucial for the security of SoC-TLS. Key management requirements for SoC-TLS involve key secrecy, fast resynchronization, low latency channel switch and secure key distribution. *The further answer to this research question is classified as confidential.*

RQ-3 *What is the mathematical strength of the keyed hash function, e.g. what is the change of a collision? The further answer to this research question is classified as confidential.*

RQ-4 *What is the mathematical strength of the crypto function, e.g. what is the order of the cryptographic key length (2^n)? The further answer to this research question is classified as confidential.*

RQ-5 *What is the cost (resources) and performance (speed, latency) of the total solution? And how does this relate to other work?* The reference design crypto core for SoC-TLS has been implemented in a single channel (SC) and multichannel (MC-15) variant that can efficiently switch between 15 secure channels. *The further answer to this research question is classified as confidential.*

7.2 Recommendation and future research

This thesis explored a new field for securing intra SoC communications by means of intra chip transport layer security. The proposal for SoC-TLS is far too much work for a single thesis project. This section will shortly describe some recommendations and future research required to work towards a fully functional SoC-TLS.

7.2.1 Required and achieved cryptographic key length

SoC-TLS depends on crypto algorithms that are specifically tailored for the applicable level of treat to SoC interconnects. Existing algorithms are too bulky for a pragmatic solution. The proposed crypto core is at the heart of a transport security layer for intra SoC communications and is thought to be *good enough* for the applicable level of threat. Two questions are of main concern. One, what is good enough? Two, how

good is the proposed solution? Answers to both can be expressed in cryptographic key length, i.e. the number of trials in an exhaustive key search. The required and achieved cryptographic key length are ideally balanced. Too much distance would indicate either a security risk or a waste of resources. We believe that the achieved length well satisfies even conservative requirements. Nevertheless, further research to find more accurate required and achieved cryptographic key length is recommended.

Main problem in finding the answer to what is good enough is that we assume a different intelligence context for intra SoC communication than for typical cryptographic solutions. Cryptanalysis always depends on gathering huge amounts of data; ciphertext only or ciphertext with corresponding plaintext. The amount of processing time and memory space required makes on-chip cryptanalysis practically infeasible, even for relatively short cryptographic key lengths. Off-chip cryptanalysis would require a covert channel that leaks sufficient - preferably - continuous ciphertext. Again we think this is infeasible because 1) the high bandwidth of SoC-TLS makes covertly streaming continuous ciphertext infeasible and 2) such a covert channel is likely to be noticed. Further research or an industry survey is needed to find the minimum required cryptographic key length.

7.2.2 Towards a fully functional SoC-TLS solution

A fully functional SoC-TLS solution would require more than the crypto core that was the focus of this thesis. Key management, message protocol and flow control have been omitted due to the limited time available for this thesis. Further design and research are desirable to learn more about the total costs, performance and practical issues. In the future we would like to see a fully functional system with SoC-TLS integrated into the interconnect.

We suggest to model a simple system with multiple processor cores with integrated SoC-TLS in SystemC TLM (Transaction-level Modelling). This model can then be used to verify the SoC-TLS design and analyze the effects on typical interconnect traffic.

HW implementations of known crypto algorithms

A

Algo.	Cite	Optimized	FPGA	Freq MHz	Slices	BRAM	Throughput
AES	[21]	Speed	Virtex-E	184.8	16,693	0	23.6 Gbs
AES	[21]	Resource	Spartan-III	67	124	2	2.2 Mbs
AES	[7]	Resource	Spartan-II	-	222	3	166 Mbs
DES	[27]	Speed	XC4028EX	20.8	733	0	166,5 Mbs
SHA-1	[32]	Speed	Virtex-E	64	1480	0	1024 Mbs
SHA-512	[32]	Speed	Virtex-E	67	3521	0	929 Mbs
MD5	[10]	Speed	V1000FG680	71.4	4763	0	354 Mbs

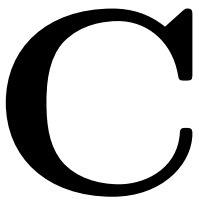
Table A.1: Performance and required resources for block cipher and hash implementations

FPGA synthesis results

B

This chapter is classified as confidential. It can be found in the confidential supplement.

Simulation Results



This chapter is classified as confidential. It can be found in the confidential supplement.

Bibliography

[1] S. Adee. The hunt for the kill switch. *IEEE Spectrum*, 2005.

[2] ARM. *AMBA Specification (rev 2.0)*. 1999.

[3] L. Benini, A. Galati, A. Macii, E. Macii, and M. Poncino. Energy-efficient data scrambling on memory-processor interfaces. In *Proceedings of the 2003 international symposium on Low power electronics and design*, page 29. ACM, 2003.

[4] D. Bertozzi and L. Benini. Xpipes: A network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits and Systems Magazine*, 4(2):18–31, 2004.

[5] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15:364, 1986.

[6] R. G. Brown. Dieharder: A Random Number Test Suite. http://www.phy.duke.edu/~rgb/General/dieharder.php, accessed Jul 2009.

[7] P. Chodowiec and K. Gaj. Very compact FPGA implementation of the AES algorithm. *Cryptographic Hardware and Embedded Systems-CHES 2003*, pages 319–333, 2003.

[8] J. Coburn, S. Ravi, A. Raghunathan, and S. Chakradhar. SECA: security-enhanced communication architecture. *Proceedings of the 2005 international conference on Compilers, architectures and synthesis for embedded systems*, page 89, 2005.

[9] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. *Design Automation Conference, 2001. Proceedings*, pages 684–689, 2001.

[10] J. Deepakumara, H.M. Heys, and R. Venkatesan. FPGA implementation of MD5 hash algorithm. *Canadian Conference on Electrical and Computer Engineering*, 2:919–924, 2001.

[11] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. Springer, 2002.

[12] J.P. Diguet, S. Evain, R. Vaslin, G. Gogniat, and E. Juin. NoC-centric security of reconfigurable soc. In *Proceedings of the First International Symposium on Networks-on-Chip*, pages 223–232. IEEE Computer Society, 2007.

[13] S. Evain and J.P. Diguet. From NoC security analysis to design solutions. *IEEE Workshop on Signal Processing Systems Design and Implementation, 2005*, pages 166–171, 2005.

[14] L. Fiorin, G. Palermo, S. Lukovic, V. Catalano, and C. Silvano. Secure Memory Accesses on Networks-on-Chip. *IEEE Transactions on Computers*, 57(9):1216–1229, 2008.

[15] L. Fiorin, G. Palermo, S. Lukovic, and C. Silvano. A data protection unit for NoC-based architectures. *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, page 172, 2007.

[16] L. Fiorin, G. Palermo, and C. Silvano. A security monitoring service for NoCs. In *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/-Software codesign and system synthesis*, pages 197–202. ACM, 2008.

[17] L. Fiorin, C. Silvano, and M. Sami. Security aspects in networks-on-chips: Overview and proposals for secure implementations. 2007.

[18] F. Gebali, L Fiorin, H. Elmiligi, and M.W. El-Kharashi. *Networks-on-Chips: Theory and Practice*. CRC Press, Inc., Boca Raton, FL, USA, 2009.

[19] C.H. Gebotys and R.J. Gebotys. A framework for security on NoC technologies. 2003.

[20] C.H. Gebotys and Y. Zhang. Security wrappers and power analysis for SoC technology. In *First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2003*, pages 162–167, 2003.

[21] T. Good and M. Benaissa. AES on FPGA from the Fastest to the Smallest. *Cryptographic Hardware and Embedded Systems–CHES 2005*, pages 427–440, 2005.

[22] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[23] K.G. Hamilton. Pseudorandom number generators for personal computers. *Computer physics communications*, 75(1-2):105–117, 1993.

[24] J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. *IACR ePrint Archive, Report*, 90, 2005.

[25] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine. Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems. 2007.

[26] ISO/IEC 17799:2005. *Information technology - Security techniques - Code of practice for information security management*. 2005.

[27] J. Kaps and C. Paar. Fast DES implementations for FPGAs and its application to a universal key-search machine. In *Selected Areas in Cryptography*, pages 630–630. Springer, 1999.

[28] R. Kastner and T. Huffmire. Threats and Challenges in Reconfigurable Hardware Security. 2008.

[29] D. E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. 1997.

[30] D.H. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings of the Second Symposium on Large Scale Digital Computing Machinery, Harvard University Press, Cambridge, MA*, pages 141–146, 1951.

[31] T.G. Lewis and W.H. Payne. Generalized feedback shift register pseudorandom number algorithm. *Journal of the ACM (JACM)*, 20(3):468, 1973.

[32] R. Lien, T. Grembowski, and K. Gaj. A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512. *Topics in Cryptology–CT-RSA 2004*, pages 1995–1995, 2004.

[33] G. Marsaglia. Random numbers fall mainly in the planes. *Proceedings of the National Academy of Sciences of the United States of America*, 61(1):25, 1968.

[34] G. Marsaglia. A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface*, pages 3–10, 1985.

[35] G. Marsaglia. Random number generation. In *Encyclopedia of Computer Science*, pages 1499–1503. John Wiley and Sons Ltd., 2003.

[36] G. Marsaglia. The Marsaglia Random Number CDROM including the Diehard Battery of Tests for Randomness. http://www.stat.fsu.edu/pub/diehard/, accessed Jul 2009.

[37] P. Martin. An analysis of random number generators for a hardware implementation of genetic programming using FPGAs and Handel-C. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 837–844. Citeseer, 2002.

[38] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969.

[39] M. Matsumoto and Y. Kurita. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2(3):179–194, 1992.

[40] M. Matsumoto and Y. Kurita. Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3):254–266, 2004.

[41] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[42] OpenCores. Wishbone b4 specification. 2010.

[43] A. Rukhin, J. Soto, J. Nechvatal, and Smid. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *NIST*, 2001.

[44] B. Schneider. *Applied cryptography*. John Wiley & Sons, 1996.

[45] IBM Systems and Technology Group. Ibm coreconnect bus cores product brief. 2006.

[46] S. Tezuka. A unified view of long-period random number generators. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 37(3):211–227, 1994.

Nomenclature

AES	Advanced Encryption Standard
ALFG	Additive Lagged Fibonacci Generator
ASIC	Application Specific Integrated Circuit
BE	Best Effort
CRC	Cyclic Redundancy Check
DPA	Differential Power Analysis
DRM	Digital Rights Management
DoS	Denial-of-Service
EM	Electro Magnetic
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FSR	Feedback Shift Register
GFSR	Generalized Feedback Shift Register
GF	Galois Field
GS	Guaranteed Service
HDL	Hardware Description Language
IP	Intellectual Property
IV	Initialization Vector
LCG	Linear Congruential Generator
LC	Linear Complexity
LFG	Lagged Fibonacci Generator
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LUT	Lookup Table
MAC	Message Authentication Code
MC	Multichannel (design variant of the SoC-TLS core)
MD5	Message Digest algorithm 5
MLFG	Multiplicative Lagged Fibonacci Generator
MSB	Most Significant Bit
MT	Mersenne Twister
NIST	National Institute of Standards and Technology
NI	Network Interface
NLFSR	Non-Linear Feedback Shift Register
NRE	Non-Recurring Engineering (costs)
NoC	Network-on-Chip
PDA	Personal Digital Assistant
PRNG	Pseudo Random Number Generator
PRS	Pseudo Random Sequence
PR	Pseudo Random

RAM	Random Access Memory
SCP	Self Complemented Path (coding)
SC	Single Channel (design variant of the SoC-TLS core)
SECA	Security-Enhanced Communication Architecture
SHA	Secure Hash Algorithm
SIMD	Single Instruction Multiple Data
SoC	System-on-Chip
TCAM	Ternary Content Addressable Memory
TGFSR	Twisted Generalized Feedback Shift Register
TLS	Transport Layer Security
TTGFSR	Tempered Twisted Generalized Feedback Shift Register
VC	Virtual Channel
VHDL	Very-high-speed integrated circuit Hardware Description Language
VPN	Virtual Private Network
XOR	exclusive or

Glossary

Asymmetrical key algorithm	Class of crypto algorithms that uses different keys for encryption and decryption. Also known as public-key cryptography. Examples are RSA and Diffie-Hellman. Often used for secure symmetrical key exchange.
Block cipher	Class of symmetrical crypto algorithms that encipher plaintext per block of certain size. Each block is enciphered in multiple round, ciphertext is output only when the whole block has finished. Block encryption has a considerable longer latency compared to stream ciphers. Examples are AES and DES.
Checksum	Code - usually a hash - that provides (certain) guarantee about the integrity of a received message.
Ciphertext	Encrypted (or enciphered) message.
Digital signature	Message digest computed with asymmetrical keyed hash function. Sender uses his private key to compute, receiver uses the senders public key to verify. This way only sender can compute and others can verify. Provides non-repudiation
Hash collision	Two messages that yield the same hash.
Hash function	Function that computes a hash corresponding to a message.
Hash	Digest of a message that can 'uniquely' identify the message. Any change in a message results in a different hash.
Keyed hash function	A hash function that depends on a key.
MAC	Message Authentication Code; provides (certain) guarantee about integrity and authenticity of a received message. MACs are computed with a keyed hash function.
Plaintext	Unencrypted (decrypted or deciphered) message.
Stream cipher	Class of symmetrical crypto algorithms in which a message is encrypted by XORing plaintext characters with a pseudo random sequence. The message is decrypted by XORing the ciphertext message with the same pseudo random sequence.

Snooping	Unauthorized access to data. Eavesdropping or listening in on communications.
Spoofing	Pretending to be someone else.
Symmetric key algorithm	Class of crypto algorithms that uses the same key for encryption and decryption. Requires secure key exchange. Generally considerably faster than asymmetrical key algorithms.