

# Some Thoughts about Implementation Properties of Stream Ciphers<sup>\*</sup>

Sandeep Kumar, Kerstin Lemke, Christof Paar

Horst Görtz Institute for IT Security  
Ruhr-Universität Bochum, Germany  
www.crypto.rub.de  
{kumar, lemke, cpaar}@crypto.rub.de

**Abstract.** This contribution describes general considerations for evaluating the quality of a cryptographic implementation, with a strong focus on hardware implementation of stream ciphers. In particular, the features area efficiency, power, and secure implementation are discussed. Even though the main target of the treatment here are stream ciphers, some of the thoughts presented are directly applicable to other crypto algorithms such as block ciphers.

## 1 Introduction

### 1.1 About this Document

Stream ciphers have always had the reputation of being more efficient than their cousins in the symmetric family, block ciphers. Even though this assumption is often true, it is not entirely clear what exactly “more efficient” means. Thus, we believe, a discussion about what constitutes a cipher with “good” implementation properties is a worthwhile undertaking. This contribution describes general considerations for evaluating the quality of a stream cipher implementation, with a strong focus on hardware implementation. Given the often predicted advent of pervasive computing applications, which may often require highly efficient (i.e., “cheap”) crypto solutions, stream ciphers are a particular interesting class of ciphers. In this context, we hope that our treatment can provide some aid for designing especially cost efficient stream ciphers. Consequently, our discussion will mainly be relevant for stream ciphers designed with hardware in mind, such as ciphers based on Linear Feedback Shift Registers (LFSRs). Ciphers which are optimized for software implementations, such as RC4, can also benefit from the discussion here but the applicability is not as straightforward.

Some of the material we provide here, in particular Sections 2 and 3, is readily available in the computer engineering literature. However, we tried to select the material most relevant to stream ciphers and to present it accessible to cryptographers with an electrical engineering background. We do not claim that we

---

<sup>\*</sup> The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, the European Network of Excellence in Cryptography

provide metrics or rules that are universally “true” and that can easily be turned into a cookbook for designing “optimum” ciphers. In fact, in practice one has to often strive for a balance between competing optimization parameters, e.g., power and throughput. Also, as will become clear when reading this document, there are quite a bit of open issues which have not been researched well enough. In any case, hope that our treatment helps to give a bit of an insight in implementation issues in general. Some of the discussion presented here might also be applicable to other crypto algorithms (or other digital systems for that matter).

## 1.2 Some General Comments about Implementing Stream Ciphers

Cryptographic algorithm have been implemented in digital form at least since the early 1970s. Almost all work up to the mid 1990s, and much of the more recent work, has focused on high throughput architectures. Typical examples for published work include high speed DES [7, 4, 15] or RSA architectures [13, 1]. The general attitude was “the faster the better”. Given the state of VLSI technology and computer clock rates, this was in fact a goal worthwhile pursuing: having RSA512 operations which take several seconds on PCs in the 1980s and early 1990s wasn’t too appealing. However, Moore’s Law (roughly speaking: computer power doubles every 18 months while the costs stay constant) is an exponential function which has made the achievement of sheer performance in crypto implementations an increasingly easier task in the last few years.

On the other hand, there have been (at least) two other main parameters emerging which can play a central role in many applications:

- **cost** (taken in a broad sense) and
- **security**

of the implementation. The fact that these two implementation parameters have become more important has several reasons: (i) Whereas in the past cryptography was mostly needed for specialized applications such as banking and government communication, the number of cost-sensitive consumer products which incorporate cryptographic functions has increased dramatically over the last decade. (ii) Mobile applications, which have barely been in existence before the 1990s, have specific needs (low power and low computational complexity) and are of great importance nowadays. (iii) Pervasive computing applications such as RFID will put even more restrictive demands on the implementation of cryptographic functionality due to its size, power and financial constraints. Based on all this arguments an initial general conclusion is that stream ciphers might gain importance, since they have the potential to allow efficient implementation.

## 1.3 What Are “Good” Implementation Properties Anyway?

Let’s now discuss the two parameters cost and security of implementations a little bit more. “Cost” can mean different things, in particular chip area (which is related to the bit complexity of the algorithm) and power consumption. However, there are more subtle aspects too such as regularity of the algorithm in

a hardware realization, depth of a boolean function (critical path), or the actual costs in a financial sense which are based on the chip area but also on the technology required. Finally, with the growing popularity importance of mobile application, the energy and power consumption of an algorithms has become another important implementation parameter.

In the reminder of this document, we will focus on the following three core parameters of stream cipher implementations:

- chip area measured in transistors
- power consumption
- security with respect to implementation attacks

## 2 Area Constraints

Implementations of stream ciphers, just as any other digital system, consist of a relatively small set of atomic functions. The main components in a stream cipher are the shift elements consisting of flip-flops (or other storage elements) and combinational elements creating a boolean logic. In addition, in actual VLSI implementation there are other functions, for instance pass transistors which can serve as switches. However, we will in the following restrict the discussion on the complexity of logic gates and storage elements. We assume CMOS logic, the dominant logic style for virtually all commercial IC nowadays.

If we attempt to evaluate the area costs of a cipher (e.g., for comparing whether cipher X is better than cipher Y) we need to a metric for comparing both logic and memory elements. We argue that stream ciphers can be more efficiently designed by going below the gate or storage element level. The atomic device on a digital chip is a transistor. Even though transistors may have different sizes on a chip, we argue that a transistor count provides a reasonable metric for comparing different building blocks. Thus, we provide in Table. 1, the area complexity for a standard cell library from [14, 5]. A different standard cell library might have slightly different relative sizes, but we believe that the table is a good approximation for practical purposes. For instance, one can see that since flip flops are more expensive than building combinatorial logic, a redistribution of the functionality within the stream cipher can allow further decrease in the size of the design.

**Table 1.** Area complexity of CMOS standard cells

	Transistors
2-input NAND	4
2-input AND	6
2-input XOR	12
D Flip Flop	26
2:1 Mux	12

There is similarly area differences in the type of storage element like SRAM, DRAM or ROM used for the implementation. A comparative study of different memory types can be found in [11] and a short Table.2 is presented here.

**Table 2.** Relative area complexity of memories

	Normalized Transistor count
DRAM	1.5
SRAM	6
ROM	1
EEPROM	2
Flash	1-1.5

### 3 Power Constraints

Power in modern devices is and will be a major concern, even if cost constraints may become less important due to decreasing feature sizes. This has been due to the fact that for mobile applications which run on battery, the technology for energy storage hasn't evolved as exponentially as for cell technology. Other concerns have been with cooling of the chip, which can be more costly than the manufacturing cost of the chip. For more futuristic applications like RFID powered by external electro-magnetic (EM) fields or sensor networks by scavenging energy in the environment for instance through small vibrations, the applications on them need to run at ultra-low power. We provide here Table 3 which shows the normalized power consumption of different cells in order to give a rough idea for the designers of stream ciphers. It should be noted that flip flops require a clock to drive them, which consumes extra power.

**Table 3.** Power consumption of  $0.18\mu m$  CMOS standard cells

	Normalized Power
2-input NAND	1
2-input AND	2.14
2-input XOR	3.36
D Flip Flop	22.55
2:1 Mux	2.77

There are also several *architectural* methods available [2] which can be used to reduce power. One such method we show here is to scale the voltage and frequency by parallelizing the architecture. This is essentially an area-power

trade-off. For a CMOS circuit, there are three components to the power dissipation: *dynamic (switching)*, *short-circuit* and *leakage power*. Short circuit power and leakage power depend on the cell technology being used. Short circuit power can be to some extent reduced by reducing the glitch in the circuit by having shorter critical paths. But the largest component of the power is the dynamic power, which for a CMOS circuit with a load capacitance  $C_L$  is given by

$$P_{dynamic} = N(f \cdot C_L \cdot V_{dd}^2)$$

where  $f$  is the clock frequency,  $V_{dd}$  the supply voltage and  $N$  the number of gates switching. Thus power is a quadratic function of the operating voltage. Reducing voltage is one way to reduce energy but it has to be noted that at lower voltages, the delay of the CMOS gate increases hence reducing the throughput. One way to overcome this is to parallelize the structure. We provide here an example from [2] which shows the gain in the power.

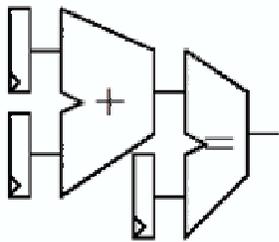


Fig. 1. Original Circuit

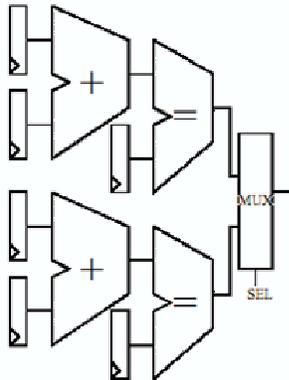


Fig. 2. Parallel Implementation

Here an adder-comparator circuit with a power consumption of  $P_{ref} = f_{ref}C_{ref}V_{ref}^2$  in a naive approach is doubled. Thus the resulting circuit as shown in Fig. 2 can be run at half the original frequency with the same throughput. The voltage can now be scaled down based on the voltage-delay function of the CMOS technology (which in the given example scales from 5 to 2.9 V). The effective capacitance of the circuit increases (also taking into account extra routing) by a factor of 2.15. Thus the power of the parallel circuit is given by

$$P_{par} = (f_{par}/2)(2.15C_{ref})(0.58V_{ref})^2 \approx 0.36P_{ref}$$

This assumes that the overhead from other components is not too large.

Designing a stream cipher which has possibilities for parallelization can be extremely useful not only in terms of power but also for most communication applications today where the smallest atomic units are 1 byte long. They can

be also less vulnerable to implementation attacks that we discuss later in the paper.

## 4 Issues on a Secure Implementation

Cryptographic modules are designed to provide IT security services, e.g. integrity and confidentiality of application data. For the corresponding security mechanisms, cryptographic keys are needed which have to be protected themselves against unauthorized disclosure and modification. In addition to considering efficient implementations, it is often even more important to secure the implementation of the stream cipher against physical attacks. Thus, the stream cipher shall be implemented both efficiently and securely. In the following, we collect previous works and aim to come up with some guidelines on preferred stream cipher constructions from a secure implementation point of view.

Implementation attacks target the physical construction of the stream cipher itself. A general pre-condition is that the attacker has physical access to the cryptographic device or at least to its near-by environment. We distinguish *active* and *passive* implementation attacks. Active attacks cover a broad range from low-budget non-invasive fault inductions to the high-end physical penetration and modification of the cryptographic device. Passive attacks leave the physical implementation intact, but gain additional knowledge just by observing its physical leakage.

### 4.1 Passive Implementation Attacks

**Side Channel Attacks** Side channel information is gained by observing either the timing, the power dissipation or the EM emanation during the processing of the cryptographic device. Combinations of these different physical channels are also feasible. Power analysis as originally introduced in [9] exploits internal data dependencies of the implementation and has yielded the most efficient methods for side channel cryptanalysis.

Side channel attacks are well studied for block ciphers and public key schemes, but there is only little information available on their application against stream ciphers. As the key generation of stream ciphers typically does not depend on externally known data and the internal state evolves permanently, a certain inherent defense against side channel techniques results.

The susceptibility of stream ciphers towards differential side channel analysis further depends on the protocol used. If the stream cipher is embedded in an authentication protocol (e.g., using random numbers) differential attacks can be mounted, as the random numbers involved have to be fed into the key generator.

Previous work on side channel cryptanalysis has focused on software implementations of the RC4 stream cipher. A sophisticated technique called “Template Attack” was introduced in [3] and applied to the RC4. It is claimed that only one single invocation of the stream cipher using the secret key is needed. For the preparation it is assumed that the attacker owns an identical programmable

device which is used for a precise characterization of the noise. An experimental study of template attacks on a micro-controller based implementation of RC4 can also be found in [12].

To the knowledge of the authors, side-channel related publications on implementations of other stream ciphers — especially hardware implementations based on LFSRs — are not available in the public literature yet. LFSRs are the basic building blocks of many keystream generators and well-suited for hardware implementations.

The data dependent part of the power dissipation of an LFSR is assumed to be composed of two parts: one part that can be modelled by the current Hamming weight of the LFSR and the other part that depends on the number of cells changed during a transition. Let  $(s_{j,0}, s_{j,1}, \dots, s_{j,n-1})$  be the internal state  $s$  of a  $n$ -bit sized LFSR after  $j$  clock cycles. The Hamming weight contribution of the power dissipation is (in a first approximation) proportional to  $\sum_{i=0}^{n-1} s_{j,i}$ . The power contribution resulting from the transition  $j \rightarrow j+1$  is approximately proportional to  $\sum_{i=0}^{n-1} (s_{j,i} \oplus s_{j+1,i}) = \sum_{i=0}^{n-2} (s_{j,i} \oplus s_{j,i+1}) + (s_{j,n-1} \oplus s_{j+1,n-1})$ . An additional power contribution originates from the tapped cells which evaluate the feedback bit  $s_{j+1,n-1}$ . Because of the sequential processing, glitches are likely to occur along this critical path. Under optimal conditions, it can be further assumed that various leakage types occur at slightly different positions in time within a one clock cycle.

As a consequence, significant leakage about the internal state is expected if extreme values of the Hamming weight or transition count can be observed. The probability to observe these events by chance during the operation of the stream cipher is minimized by choosing sufficiently long bit-sized LFSRs. Moreover, if multiple LFSRs are operated in parallel, the power contribution of one specific LFSR can hardly be identified.

One remaining critical point is the key initialization sequence if the initial state of the LFSRs is zero as for instance in the A5/1 cipher. In the initialization process of the A5/1, one key bit enters all three LFSRs at each clock cycle. If this sequence is observed, probably the entire sequence of the key bits can be disclosed.

As result, a general guideline for the cipher design is that the bit-wise key transfer into LFSRs with a known initialization state should be avoided. A parallel loading of multiple key bits is encouraged. If different LFSRs are operated in parallel, they should be entered by different key bits at the same point in time.

Another application of the power dissipation leakage concerns clock controlled generators, which induce an irregular motion of the LFSRs depending on an internal state of the LFSRs. We guess that the overall power dissipation is different according to the number of LFSRs in operation. If the overall bit-length of the clocked LFSRs is different, this can offer a possibility to decide which LFSRs are clocked.

The most promising approach to counteract side channel leakage of hardware based stream cipher is the usage of special logic styles, as e.g. SABL (see [8]).

## 4.2 Active Implementation Attacks

**Non-invasive Attacks** Fault attacks on stream ciphers are thoroughly studied in [6].

In case of block ciphers and public-key cryptosystems, faults can be detected by computing the same operation twice or by using the inverse operation. This is not feasible for stream ciphers. Possible hardware countermeasures include dual-rail logic styles ([10]) with an alarm mechanism and a redundant design of the stream cipher including a comparison of the output bits.

**Invasive Attacks** Invasive attacks target the physical security of the device. Typical scenarios include penetration and modification. The common aim is to intercept data at the internal communication lines or to read out memory cells in order to determine the secret keys stored inside the device.

The internal construction of a secure hardware implementation should avert these invasive attacks. A prominent example are secure smart-cards which provide passive protection against physical attacks ('Tamper Resistance').

Critical points of a hardware design are connection lines which are used for the key transfer. If the keying of the stream cipher is done bit by bit, the implementation can be vulnerable against probing once this position is identified in the layout. One successful probing on this line would be sufficient to monitor the entire key.

Constructions which process the majority of key bits sequentially should be avoided in the implementation. The transfer of key bits should be parallelized on different lines as much as possible. An alternative hardware countermeasure is an internal random masking of the key bits processed at gate level.

For the design of stream ciphers it is recommended to avoid one long bit-sized LFSR and to prefer the use of multiple LFSRs operating in parallel.

## 5 Conclusion

We introduced some metrics for evaluating the implementation costs for stream ciphers, and summarized what is known about side channel attacks. We believe much work can still be done in the area of modern stream cipher implementation. Here are a few suggestions that we can make:

- Researching and implementing side-channel attacks which are specific for stream ciphers, such as observing LFSR through side channels.
- Design of stream ciphers at extremely low costs, say a few 100 gates, for pervasive computing applications.
- Stream ciphers which have internal parallelization for low-power design.

## References

1. T. Blum and C. Paar. High radix Montgomery modular exponentiation on reconfigurable hardware. *IEEE Transactions on Computers*, 50(7):759–764, July 2001.

2. A. Chandrakasan and R. Brodersen. *Low-Power CMOS design*. IEEE Press, 1998.
3. S. Chari, J.R. Rao, and P. Rohatgi. Template attacks. In B.S. Kaliski, Ç Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems*, volume 2523 of *LNCS*, pages 13–28. Springer-Verlag, 2003.
4. H. Eberle. A high-speed DES implementation for network applications. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume LNCS 740, pages 521–539, Berlin, Germany, August 16–20 1993. Springer-Verlag. Conference Location: Santa Barbara, California, USA.
5. J. Grad and J.E. Stine. A standard cell library for student projects. In *International Conference on Microelectronic Systems Education*, pages 98–99. IEEE Computer Society, 2003.
6. J.J. Hoch and A. Shamir. Fault analysis of stream ciphers. In Marc Joye and Jean-Jacques Quisquater, editors, *Chryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *LNCS*, pages 240–253. Springer-Verlag, 2004.
7. F. Hoornaert, J. Goubert, and Y. Desmedt. Efficient hardware implementation of the DES. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO'84*, volume LNCS 196, Santa Barbara, California, USA, August 19–22 1985. Springer-Verlag.
8. I. Verbauwhede K. Tiri. Securing encryption algorithms against dpa at the logic level: Next generation smart card technology. In C. D. Walter, Ç Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2003*, volume 2779 of *LNCS*, pages 125–136. Springer-Verlag, 2003.
9. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.
10. S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor. Improving smart card security using self-timed circuits”. In *Proc. 8th IEEE International Symposium on Asynchronous Circuits and Systems — ASYNC '02*, pages 23–58. IEEE, 2002.
11. B. Prince. Quality memory blocks-balancing the trade-offs. In *IEEE 2000 First International Symposium on Quality Electronic Design, 2000. ISQED 2000*, pages 109–114, San Jose, CA , USA, March 2000.
12. C. Rechberger. Side channel analysis of stream ciphers. Master’s thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria, 2004.
13. M. Shand and J. Vuillemin. Fast implementations of RSA cryptography. In E. Swartzlander, Jr., M. J. Irwin, and G. Jullien, editors, *Proceedings of the 11th IEEE Symposium on Computer Arithmetic (ARITH-11)*, pages 252–259, 1993.
14. VLSI Computer Architecture, Arithmetic, and CAD Research Group – Department of Electrical Engineering, IIT, Chicago, IL. IIT Standard Cells for AMI 0.5 $\mu$ m and TSMC 0.25 $\mu$ m/0.18 $\mu$ m (Version 1.6.0) , 2003. Library and documentation available from <http://www.ece.iit.edu/~vlsi/scells>.
15. D.C. Wilcox, L. Pierson, P. Robertson, E. Witzke, and K. Gass. A DES ASIC Suitable for Network Encryption at 10 Gbps and Beyond. In Ç. Koç and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 1999*, volume LNCS 1717, pages 37–48, Worcester, Massachusetts, USA, August 1999. Springer-Verlag.