The ISC Int'l Journal of Information Security

ISeCure

January 2010, Volume 2, Number 1 (pp. 3–11)

http://www.isecure-journal.org

INVITED PAPER

Stream Ciphers and the eSTREAM Project[⋆]

Vincent Rijmen[a,∗]

[a] *Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC and IBBT, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium, and Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria*

ARTICLE INFO.

Article history:
Received: 6 October 2009
Revised: —
Accepted: 30 November 2009
Published Online: 18 January 2010

Keywords:
Cryptology, Stream Ciphers, Time-Memory Trade-offs, eSTREAM

ABSTRACT

Stream ciphers are an important class of symmetric cryptographic algorithms. The eSTREAM project contributed significantly to the recent increase of activity in this field. In this paper, we present a survey of the eSTREAM project. We also review recent time/memory/data and time/memory/key trade-offs relevant for the generic attacks on stream ciphers.

1 Introduction

The design of secure stream ciphers is one of the oldest problems in cryptography. Although there exists a nicely developed theory that answers several of the important questions, the question is not fully solved (and it will probably never be). After the completion of the Advanced Encryption Standard (AES) process, block ciphers were firmly in the center of the cryptographic community's attention. Some people started wondering aloud whether there was still any practical application for stream ciphers or a reason to perform research on them. The eSTREAM project to evaluate stream ciphers, organized by the ECRYPT Network of Excellence, can be seen as an answer formulated by the part of the cryptographic community that does care about stream ciphers. It turned out to be a large part of the community.

In this paper, we give an overview of the eSTREAM project and we describe some lessons learnt on the design of secure stream ciphers. In Section 2 we start with one of the remarkable issues in the stream ciphers versus block ciphers debate, namely the fuzzy border separating them from one another. In Section 3, we describe some general results that were obtained during the eSTREAM competition. These generic attacks bound the best security that can be achieved for a given size of the secret key and the internal state. Section 4 discusses the eSTREAM highlights and events. We present some concluding remarks in Section 5.

2 Synchronous Stream Ciphers, Self-Synchronizing Stream Ciphers and Block Ciphers

Many introductory texts on symmetric cryptography distinguish two classes of primitives for symmetric encryption, namely block ciphers and stream ciphers. An often-cited example of a historical block cipher is the Caesar cipher. More modern examples include the public algorithms Data Encryption Standard (DES) and AES. Example stream ciphers are the historical Enigma cipher, RC4 and various ciphers based on Linear Feedback Shift Registers (LFSRs).

⋆ Sponsored by Onderzoeksfonds K.U.Leuven / Research fund K.U.Leuven OT/08/027.
∗ Corresponding author.

Email address: `vincent.rijmen@esat.kuleuven.be` (V. Rijmen).

ISSN: 2008-2045 © 2010 ISC. All rights reserved.

2.1 Definitions

The *Handbook of Applied Cryptography* gives the following definitions for a synchronous, respectively self-synchronizing stream cipher.

Definition 1 (Synchronous Stream Cipher [13, Definition 6.2]). A *synchronous stream cipher (SSC)* is one in which the keystream is generated independently of the plaintext message and of the ciphertext.

Denoting message blocks by m_i, ciphertext blocks by c_i, the key by κ and the content of the internal state of the stream cipher at time i by σ_i, an SSC can be described by the following equations:

$$\sigma_{i+1} = f(\sigma_i, \kappa), \quad (1)$$
$$z_i = g(\sigma_i, \kappa), \quad (2)$$
$$c_i = h(z_i, m_i). \quad (3)$$

Here σ_0 is the *initial value* of the internal state and $z_0, z_1, z_2, \ldots$ is the *keystream*. The function g is called the *output transformation*.

Definition 2 (Self-Synchronizing Stream Cipher [13, Definition 6.5]). A *self-synchronizing stream cipher (SSSC)* is one in which the keystream is generated as a function of the key and a fixed number of previous ciphertext digits.

In an SSSC, Equation (1) is replaced by:

$$\sigma_i = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}). \quad (4)$$

Note that the Handbook defines both synchronous stream ciphers and self-synchronizing stream ciphers in terms of the more general class of stream ciphers. Interestingly, the Handbook omits to give a general definition of a stream cipher. Instead, the Handbook defines block ciphers and provides guidelines to distinguish stream ciphers from block ciphers.

Although there exists a common and clear intuition about the difference between a block cipher and a stream cipher, capturing this intuition in precise mathematical statements proves to be a challenge. The Handbook defines a block cipher as follows:

Definition 3 ([13, Definition 7.1]). An n-bit *block cipher* is a function $E : V_n \times \mathcal{K} \rightarrow V_n$ such that for each key $\kappa \in \mathcal{K}$, $E_\kappa(P)$ is an invertible mapping from V_n to V_n, written $E_K(P)$.

The Handbook also points out two properties in which block ciphers and stream ciphers tend to differ [13, page 192]:

Block ciphers process plaintext in relatively large blocks (e.g. $n \geq 64$ bits). The same function is used to encrypt successive blocks; thus (pure) block ciphers are *memoryless*. In contract, stream ciphers process plaintext in blocks as small as a single bit, and the encryption function may vary as plaintext is processed; thus stream ciphers are said to have memory. They are sometimes called *state ciphers* ... This distinction between block and stream ciphers is not definitive (...); adding a small amount of memory to a block cipher (as in CBC mode) results in a stream cipher with large blocks.

The Handbook proposes hence to take the size of the blocks processed and the presence or absence of memory as criteria to distinguish between block ciphers and stream ciphers.

If we consider the modern stream cipher proposals, for example the proposals submitted to the eSTREAM competition, then it turns out that many of the best-performing stream ciphers work on relatively large blocks, just like block ciphers. This is true in particular for stream ciphers designed to have a high performance in software; they usually process large blocks in order to benefit from the large registers available on modern processors.

Secondly, it should be pointed out that for the vast majority of practical applications, and for all applications that require some kind of provable security, block ciphers are being used in a *mode of operation* which introduces state (memory). Hence, this criterion to distinguish between block ciphers and stream ciphers turns out to be imprecise. On the other hand, among cryptographers, there appears to be a common intuition about which primitive should be considered to be a stream cipher and which a block cipher. We think that this intuition is captured by the following working definition.

2.2 New Working Definition

Any secure encryption method contains a kind of internal state, a family $\{F_\kappa\}_\kappa$ of state update transformations and a family $\{G_\kappa\}_\kappa$ of mechanisms to produce ciphertext. In mathematical notation, we can write:

$$\sigma_{i+1} = F_\kappa(\sigma_i, m_i), \quad (5)$$
$$c_i = G_\kappa(\sigma_i, m_i). \quad (6)$$

A block cipher is a family $\{E_\kappa\}_\kappa$ of permutations which can be used in a certain configuration —known as a mode of operation— to define the transformations F_κ, G_κ. Hence a block cipher based encryption method is an example of a modular design, consisting of a block cipher on the one hand, and a mode of operation on the other hand.

A stream cipher is an encryption method which doesn't necessarily employ this modular approach. By allowing more general constructions for F_κ and G_κ, the designers aim to achieve a better tradeoff between performance, security and cost.

Note: In the Electronic Code Book (ECB) mode of operation, there is no internal state. It can be treated as a special case of the previous definition, with state size 0. Alternatively, we can exclude it from consideration because for the majority of applications, the ECB mode of operation can't be considered secure.

2.3 Constructions

Traditionally, definitely in academic research papers, SSC's have been constructed from Linear Feedback Shift Registers (LFSRs). The main advantages of LFSRs are their compactness in hardware, the well-developed mathematical theory surrounding their design and the good randomness properties. Their main disadvantage is of course their linearity, which needs to be destroyed or hidden. This is typically done by employing a nonlinear output filter, adding nonlinear components to the feedback function or using irregular clocking (decimation). Nonlinear output filters come with the most developed mathematical theory, but recent improvements in cryptanalysis methods have rendered most of the known designs insecure [5]. Nonlinear feedback functions or, more generally nonlinear state update transformations occur in large variety. They typically come with very little theory and consequently their security is not well understood. Many designs have been broken by means of fast correlation attacks [12], linear attacks [7, 8], resynchronization attacks [6], etc.

3 Trade-offs in Generic Attacks on Stream Ciphers

One of the interesting effects of the eSTREAM competition was a renewed interest in generic attacks on stream ciphers. The literature describes several brute-force generic attacks on stream ciphers. From a high-level point of view, the attacks can be described as a sequence of two phases:

Precomputation phase: The results of the computations performed in this phase can be reused for several iterations of the attack. Typically, the results are some tables which are used to speed up the last phase of the attack. We denote the binary logarithm of the computational complexity of this phase by P.

Online phase: The attacker collects data, which usually consists of ciphertexts and corresponding known or chosen plaintexts. We count the amount of data collected in units of k bits, and denote the binary logarithm of this quantity by D. This data is combined with the results of the precomputation phase to recover the key. We denote the binary logarithm of the computational complexity of this phase by T.

We denote the binary logarithm of the sum of the memory requirements of the precomputation phase and the online phase of the attack by M.

3.1 Exhaustive Attacks

For a straightforward exhaustive key search, there is no precomputation, the attacker collects a negligible amount of known plaintexts, and uses a negligible amount of memory ($P = D = M \approx 0$). The online computational complexity is given by $T = k$.

On the other hand, we can also imagine an attack scenario where the attacker precomputes the ciphertext for a given chosen plaintext under all possible keys, and stores the result in a table. This gives $P = M = k$. The online phase of the attack consists of obtaining the ciphertext corresponding to the chosen plaintext and looking up the key in the table ($D \approx T = 0$).

Notes:

(1) For an SSC a chosen-plaintext attack can always be replaced by a known-plaintext attack with the same complexities.

(2) Some authors argue that it is not realistic to say that the time to look up data in a table doesn't depend on the size of the table [2]. All authors agree that memory cost and computation cost are very different things. Hence, when comparing various trade-off curves or points on a given trade-off curve, one has to be careful. A point with a much higher T, but a slightly decreased M compared to its alternatives, might well be the best choice.

Between the two extreme cases formed by exhaustive key search and a full table based attack, several trade-off scenarios can be defined. They are discussed next. We denote the key length (in bits) by k. The classical time-memory trade-off described by Hellman [10] has the following complexities:

$$T = M = 2k/3,\ P = k \text{ and } D = 0,$$

which can be generalized to

$$T + 2M = 2k,\ P = k \text{ and } D = 0.$$

This trade-off works for any one-way function, hence also for stream ciphers. The attack recovers the secret key.

3.2 Time-Memory-Data Trade-offs

For stream ciphers with an unkeyed output transformation, it is possible to define trade-off attacks targeting the internal state instead of the key. We denote the size of the internal state (in bits) by s. Babbage and Golić (BG) [1, 9] describe time-memory-data trade-off attacks with:

$$T + M = s,\ D = T \text{ and } P = M. \qquad (7)$$

Biryukov and Shamir (BS) [4] describe time-memory-data trade-off attacks targeting the internal state with:

$$T + 2M + 2D = 2s,\ T \geq 2D \text{ and } P + D = s. \qquad (8)$$

In 'practice', one usually chooses $T = 2D$ for the Biryukov-Shamir trade-off. If the state size is at least twice the key length, then both the Babbage-Golić trade-offs and the Biryukov-Shamir trade-offs become less efficient than the previously described attacks.

3.3 Time-Memory-Key Trade-offs

In [3, 11] a new type of trade-off is considered: the time-memory-key trade-off. The attack works in a scenario where an attacker can obtain a large number of short key streams, produced with different keys, but the same IV. The attack recovers one of the keys.

We denote by K the binary logarithm of the number of keys that are attacked in parallel. The trade-offs (7) and (8) become now:

$$T + M = k,\ K = T \text{ and } P = M, \qquad (9)$$

$$T + 2M + 2D = 2k,\ T \geq 2K \text{ and } P + K = k\ . \qquad (10)$$

The data complexity of the attacks equals 1 k-bit string per key.

Figure 1 compares the Babbage-Golić trade-offs and the Biryukov-Shamir trade-offs. For increasing K, the Biryukov-Shamir has a faster decreasing memory complexity, but a slower decreasing computational complexity. It follows that the Biryukov-Shamir approach is better adapted to the current state of computing technology, where computations are cheaper than memory.

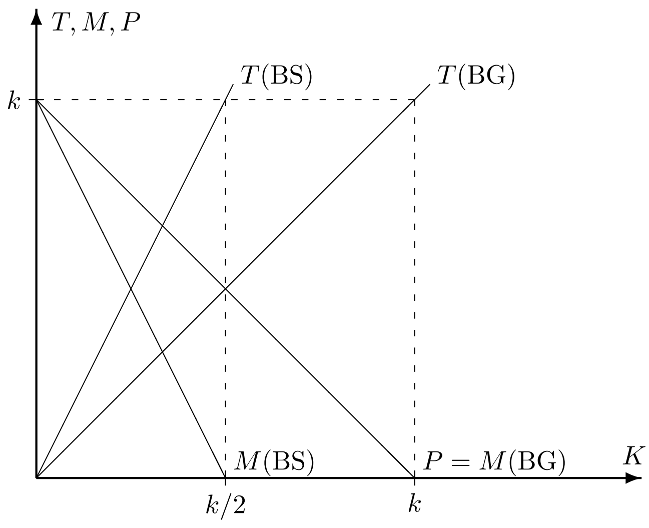

Figure 1. Time-Memory-Key trade-offs for the Babbage-Golić (BG) and for the Biryukov-Shamir (BS) curves. The precomputation (P) curve is the same for both cases, and equal to the BG memory M curve.

4 eSTREAM

The eSTREAM competition was a project in the *Network of Excellence (NoE)* ECRYPT, which was funded by the EU in the Framework 6 Information Society Technologies (IST) programme, under the strategic objective 2.3.1.5 "Towards a global dependability and security framework." ECRYPT ran from February 2004 until July 2008. In August 2008, it was succeeded by ECRYPT II.

4.1 ECRYPT

ECRYPT counted five *virtual labs* grouping researchers from each of the partners around 5 different themes in cryptology and information security:

STVL stood for Symmetric Techniques Virtual Lab and dealt with themes in symmetric cryptography,
Aztec stood for Asymmetric Techniques Virtual Lab and dealt with themes in asymmetric cryptography,
Provilab stood for Protocols Virtual Lab and coordinated research in cryptographic protocols,
Vampire stood for Virtual Applications and Implementations Research Lab and researched new techniques related to efficient and secure implementations (of cryptographic algorithms),
Wavila stood for Watermarking Virtual Lab and has goal to bring watermarking and perceptual hashing to a higher degree of maturity.

Note that in ECRYPT II, there are only three virtual labs remaining. STVL has been renamed into Symlab, Aztec and Provilab have merged into a new virtual lab called Maya, Vampire has not changed, and the activities of Wavila are not continued within ECRYPT II.

4.2 eSTREAM Goals

The eSTREAM competition was administrated by members of STVL. The goals of the competition were twofold: firstly, to advance our understanding of the design and analysis of secure stream ciphers, and secondly, to identify a portfolio of promising stream ciphers. In contrast to the Advanced Encrypotion Standard (AES) process organized by NIST, eSTREAM was strictly speaking *not* a standardization effort. In reality however, several standardization bodies, e.g. ISO, ETSI, paid close attention to the outcomes of the eSTREAM evaluation process. Indeed, several researchers participating in the eSTREAM evaluation, who were at the same time experts in standardization bodies, ensured an informal exchange of information. An important advantage of the lack of formal status was the increased freedom with respect to procedures. For instance, during the evaluation process, submitters were allowed to significantly modify their designs in order to counter newly-developed cryptanalytic attacks. This was done in order to give designers of interesting algorithms the opportunity to fix flaws that were identified early in the process.

Also an earlier EU-sponsored research project, NESSIE, which ran from 2000 until 2003, included a competition for cryptographic algorithms. The scope of NESSIE was much wider then eSTREAM, and fewer stream ciphers were submitted. All submissions were broken during or shortly after the NESSIE competition. This illustrated once more the apparent lack of knowledge about the design of fast and secure stream ciphers. It was also one of the reasons to allow the submitters in eSTREAM to fix the flaws in their designs.

Figure 2 shows the time line of eSTREAM. After the initial Call for Primitives, which was published in November 2004, there were 3 evaluation phases, each followed by a narrowing-down of the submissions. The final selection, or portfolio, was presented in April 2008 (updated in September 2009). Three *State of the Art of Stream Ciphers (SASC)* conferences were organized, one near the end of each of the three evaluation phases and almost exclusively dedicated to the eSTREAM project. Additionally, one *Symmetric Key Encryption Workshop (SKEW)* was organized at the start of the evaluation process.

4.3 Preparation Phase

During the preparation phase, several players in the field, both from industry and academia, were contacted and brought together to discuss the requirements to put forward in the evaluation and selection. The central theme was to define one or more environments in which stream ciphers could offer substantial advantages over other cryptographic primitives.

The Call for Primitives was published in November 2004. It specified two types of environments, for which it was believed that stream ciphers could offer the largest advantage over block ciphers, in particular over the Advanced Encryption Standard (AES).

Profile 1, also known as the *software profile* aimed for stream ciphers with an extremely high throughput in software implementations. Ciphers in this category needed to support key lengths of at least 128 bits and IV lengths of 64 and 128 bits.

Profile 2, also known as the *hardware profile* aimed for stream ciphers suited to implementations in restricted hardware environments (small area, small power/energy consumption). Ciphers in this category needed to support key length of at least 80 bits and IV lengths of 32 and 64 bits.

The Call mentioned also that associated authentication mechanisms could be proposed and would be evaluated. At the end of the competition, it turned out that no authentication mechanism was deemed to be secure enough and at the same time running with a sufficiently high performance.

The official evaluation criteria were:

(1) Security,
(2) Performance, when compared to AES and when compared to other submissions,
(3) Justification and supporting analysis,
(4) Simplicity and flexibility of the design,
(5) Completeness and clarity of the submission.

The most important criterion was of course security, but apart from the Boolean variable broken/unbroken it is difficult to distinguish different designs based on their security alone. Performance comparisons, on the other hand, lead automatically to rankings, which are more convenient when comparing several designs. In order to make these rankings as fair as possible, an extensive software performance comparison tool was developed, which tested out various platforms and compiler options.

The three remaining criteria were used mostly as a filter. For example, designs that were completely lacking supporting analysis or had ambiguous documentation, had a larger chance of being rejected early on in the evaluation process.

eSTREAM received in total, 34 submissions. The majority of the submissions came from European teams, but Australia, China, USA, Canada, . . . were represented, too.

Nine designers submitted their algorithm to Profile 1 and twelve to Profile 2. Thirteen of the designers submitted their algorithm to both profiles. There was no punishment for submitting an algorithm to both profiles, so this was probably the most sensible choice from the designers' point of view. Only two of the submissions were SSSC. Seven submissions included a method to authenticate data. The designs were based on LFSRs, NLFSRs, T-functions, SP-networks or a

Phase	Date	Event
Preparation (10/'04–4/'05)		
	14-15/10/'04	SASC 2004
	11/'04	Publication of the First Call for Primitives
	29/4/'05	Deadline Submission of Primitives
Phase 1 (5/'05–3/'06)		
	26-27/5/'05	SKEW 2005: Presentation of Submissions
	2-3/2/'06	SASC 2006
Phase 2 (8/'06–3/'07)		
	31/1-1/2/'07	SASC 2007
	26/3/'07	Publication of Phase 2 Report
Phase 3 (4/'07–4/'08)		
	12-13/11/'07	STVL Stream Cipher Retreat
	13-14/2/'08	SASC 2008
	15/4/'08	Publication of the eSTREAM Portfolio
	8/9/'08	Publication of the eSTREAM Portfolio Rev1

Figure 2. Main eSTREAM events.

combination of several of them. The majority of the submissions were presented at the Symmetric Key Encryption Workshop (SKEW 2005) organized by Ecrypt/STVL on May 26-27, 2005, which marked the start of Evaluation Phase 1.

4.4 Phase 1

Phase 1 ran from May 4th, 2005 until February 2006. It was characterized by a large number of cryptanalytic results on the submissions. Security issues were identified in no less than 22 candidates. This relatively large number of weaknesses can again be seen as a proof of the fact that stream cipher design was (is) not a fully understood discipline (yet).

During Phase 1, the eSTREAM framework for software performance testing was finalized and made available to the public. The framework made it possible to measure the software performance of algorithms on a variety of platforms and compilers. The typical stream cipher encryption process consists of three parts: key setup, IV setup and raw keystream production. In order to get a balanced view on the performance of the submissions in practical applications, the performance was measured for keystreams of 40 bytes, 576 bytes, 1500 bytes and for stream encryption, i.e. without taking into account the setup phases.

Producing a good hardware implementation of a cipher takes more time, and is much less an automated process than software compilation. During Phase 1, it became clear that it is difficult to compare hardware implementations made by different teams because of the many different hardware libraries. There appeared also to be a shortage of teams with good background in both cryptology and hardware implementations. This led to suboptimal implementations. As a result, the outcome of hardware implementation efforts was not decisive in the early phases of eSTREAM.

At the end of Phase 1, a panel of experts read all reports and evaluated the 34 submissions. Seven submissions were not advanced to Phase 2, or were *archived*. A submission was archived only when security weaknesses had been identified or a very bad software performance had been reported, *and* the submitters had failed to propose a fix or didn't submit updated code and documentation.

4.5 Phase 2

Phase 2 ran from August 2006 until March 2007. Many of the submissions that had been fixed after Phase 1 also changed names slightly in this phase. For example, the new version of the cipher Mosquito was called Moustique. The cipher Mickey continued as two different ciphers, namely Mickey v2 and Mickey-128 v2. This made for a total of 28 submissions at the start of

Phase 2.

This large number of survivors, despite the large amount of cryptanalytic results in Phase 1, was a consequence of the liberal policy towards algorithm fixes. From the start of Phase two on, no more algorithm fixes would be allowed.

In order to focus the attention of the evaluators on a smaller group of promising ciphers, the surviving submissions were divided into two categories: *Phase 2 algorithms* and *Focus phase 2 algorithms*. New results of cryptanalysis could move a submission from the Focus phase 2 category to the Phase 2 category, and this could lead to the 'promotion' of new algorithms into the Focus category. From the submitters' point of view, being in the Focus category was a dubious honor, because the increased attention from cryptanalysts could be expected to result in the sooner discovery of weaknesses in the submission, and hence its elimination from the competition. Furthermore, it remained unclear until the end of Phase 2 whether a submission needed to be in the Focus category in order to be selectable for Phase 3 at all or not. Although it is difficult to find out whether the introduction of the Focus category contributed much to the evaluation process, it remains a fact that also in the second phase of the eSTREAM evaluation there was a multitude of cryptanalytic results and performance comparisons.

4.6 Phase 3

Phase 3 ran from April 2007 until April 2008. This phase started with the publication of the end report on Phase 2 [15]. The 16 ciphers selected for Phase 3, 8 for each profile, are listed in Table 1.

Seven of these ciphers were not Focus phase 2 ciphers. It is remarkable that at the start of Phase 3, there were no authentication methods left, because they were either broken or too slow. Even in this advanced stage of the evaluation process there were still cryptanalytic results on some of the ciphers.

In the Escargot project (European Stream Ciphers are Ready to Go), the eight surviving Profile 2 ciphers were implemented together on an ASIC and fabricated on 0.18 μm CMOS [17]. The chips were available free of charge to groups with a recognized capability for side-channel analysis.

4.7 The eSTREAM Portfolio

The eSTREAM project did not end with the selection of one 'winner.' Instead, a portfolio of algorithms was selected. There are several motivations for selecting a portfolio. Firstly, since the stream ciphers are mainly intended for use in 'extreme' environments, it makes sense to offer a selection with varying characteristics, suitable for different applications: fast stream encryption, short packet encryption, large security margin, novel design or based on classical components, etc. Secondly, because of the immature nature of the area, it was expected from the start that some of the ciphers would drop out after new cryptanalytic results would have become available.

Table 1. The Phase 3 ciphers. Ciphers with * were Focus phase 2 ciphers.

Profile 1	**Profile 2**
CryptMT	Decim v2
Dragon*	Edon-80
HC-128*	F-FCSR-H
LEX*	Grain v1*
NLS*	Mickey v2*
Rabbit	Moustique
Salsa20*	Pomaranch
Sosemanuk*	Trivium*

Table 2. The eSTREAM portfolio, Rev1, Sept. 2008. Only Rabbit was not in the Focus phase 2 category.

Profile 1	**Profile 2**
HC-128	Grain v1
Rabbit	Mickey v2
Salsa20/12	Trivium
Sosemanuk	

The first eSTREAM portfolio was announced in April 2008, together with a report on the findings of Phase 3 [16]. The portfolio contained eight algorithms, four in each profile. Shortly after the announcement of the portfolio, results were announced on the stream cipher F-FCSR-H. This cipher was hence removed from the September 2008 revision of the portfolio, which is listed in Table 2. Since then, no changes have been made to the portfolio. Rabbit is the only stream cipher in the portfolio that was never in the Focus phase 2 category.

5 Concluding Remarks

The eSTREAM project resulted in a portfolio of 8 (now 7) stream ciphers that are recommended for further study. Although the most visible, the portfolio is definitely not the only, and probably not the most important result of eSTREAM.

The eSTREAM project succeeded in enlivening the research area of stream ciphers. The eSTREAM web site attracted several hundreds of thousands of visits. The online discussion forum attracted high-quality postings. It appears that the competitive aspect of the evaluation effort increased the attractiveness to designers, implementors and evaluators, driving many researchers all over the world to spend time and effort on this project. The research performed during this project resulted in papers presented at the dedicated series of SASC workshops, but also at international top conferences like FSE, SAC and Eurocrypt. The question is of course whether the level of activity in the field will remain at this increased level or not.

During the evaluation of the submissions, it also became clear that some tough stream cipher related research questions remain open. For example, all of the submitted self-synchronizing stream ciphers were broken, so the design of a secure and efficient SSSC remains an open problem. Secondly, almost all practical applications of encryption need in fact authenticated encryption. Few authentication mechanisms were submitted to eSTREAM, and all of them were broken or an order of magnitude slower than the encryption process. Also this remains an area where further research is welcomed.

A survey of the eSTREAM submissions and survey papers of their software and hardware performance as recorded by eSTREAM, can be found in [14].

Acknowledgements

This article is based on a talk I gave at ICISC 2008; I would like to thank its organizers for inviting me to talk on this topic. Secondly, I want to thank Bart Preneel, the coordinator of ECRYPT and ECRYPT2 for providing me with supporting material.

References

[1] Steve Babbage, A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers, *European Convention on Security and Detection,* Volume 408, 1995.

[2] Dan J. Bernstein, Understanding Brute Force, *Workshop on Symmetric Key Encryption (SKEW 2005),* Århus, May 27th, 2005. http://cr.yp.to/talks/2005.05.27/slides.pdf

[3] Alex Biryukov, Sourav Mukhopadhyay, Palash Sarkar, Improved Time-Memory Trade-offs with Multiple Data, *Selected Areas in Cryptography (SAC 2005),* LNCS 3897, pages 110–127, Springer-Verlag, 2006.

[4] Alex Biryukov, Adi Shamir, Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers, *ASIACRYPT 2000,* LNCS 1976, pages 1–13, Springer-Verlag, 2000.

[5] Nicolas Courtois, Fast Algebraic Attacks on Stream Ciphers with Linear Feedback, *CRYPTO 2003,* LNCS 2729, pages 176–194, Springer-Verlag, 2003.

[6] Joan Daemen, René Govaerts, Joos Vandewalle, Resynchronization Weaknesses in Synchronous Stream Ciphers, *EUROCRYPT 1993,* LNCS 765, pages 159–167, Springer-Verlag, 1994.

[7] Jovan Dj. Golić, Correlation via Linear Sequential Circuit Approximation of Combiners with Memory, *EUROCRYPT 1992,* LNCS 658, pages 113–123, Springer-Verlag, 1993.

[8] Jovan Dj. Golić, Linear Cryptanalysis of Stream Ciphers, Fast Software Encryption (FSE 1994), LNCS 1008, pages 154–169, Springer-Verlag, 1995.

[9] Jovan Dj. Golić, Cryptanalysis of Alleged A5 Stream Cipher, *EUROCRYPT 1997,* LNCS 1233, pages 239–255, Springer-Verlag, 1997.

[10] Martin Hellman, A Cryptanalytic Time-Memory Trade-off, *IEEE Transactions on Information Theory,* Volume 26, pages 401–406, 1980.

[11] Jin Hong, Palash Sarkar, New Applications of Time Memory Data Trade-offs, *ASIACRYPT 2005,* LNCS 3788, pages 353–372, Springer-Verlag, 2005.

[12] Willi Meier, Othmar Staffelbach, Fast Correlation Attacks on Certain Stream Ciphers, *J. Cryptology* Vol. 1, No. 3, pages 159–176, 1989.

[13] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography,* CRC Press, 1996.

[14] Matthew Robshaw, Olivier Billet, *New Stream Cipher Designs,* LNCS 4986, 2008.

[15] Steve Babbage, Christophe de Canniére, Anne Canteaut, Carlos Cid, Henri Gilbert, Thomas Johansson, Christof Paar, Matthew Parker, Bart Preneel, Vincent Rijmen, Matt Robshaw, Hongjun Wu, eSTREAM, Short Report on the End of the Second Phase, http://www.ecrypt.eu.org/stream/PhaseIIreport.pdf

[16] Steve Babbage, Christophe De Cannière, Anne Canteaut, Carlos Cid, Henri Gilbert, Thomas Johansson, Matthew Parker, Bart Preneel, Vincent Rijmen, Matthew Robshaw, The eSTREAM Portfolio, http://www.ecrypt.eu.org/stream/portfolio.pdf

[17] Tim Good, Escargot, http://www.shef.ac.uk/eee/escargot/

Vincent Rijmen graduated in 1993 as electronics engineer from the University of Leuven, Belgium (KU Leuven) and finished in 1993 his doctoral dissertation on the design and analysis of block ciphers in 1997.

He is co-designer of the algorithm Rijndael, which in October 2000 was selected by the National Institute for Standards and Technology (NIST) to become the Advanced Encryption Standard (AES) the successor to the existing Data Encryption Standard (DES).

In 2001, Rijmen became Chief Cryptographer of Cryptomathic, a European company developing software for cryptographic applications. In 2004, he became full professor at the Graz University of Technology, where he heads the research unit "Krypto" of the institute of applied information processing and communications (IAIK). Since September 2007, Rijmen is full professor at the KU Leuven and part time full professor at the Graz University of Technology.