TRIVIUM
Specifications*

Christophe De Cannière** and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10,
B–3001 Heverlee, Belgium
{cdecanni, preneel}@esat.kuleuven.be

Abstract. This document specifies TRIVIUM, a hardware oriented synchronous stream cipher which aims to provide a flexible trade-off between speed and area. The description of the cipher is followed by some performance figures and a summary of the cryptographic properties of the algorithm. For a more theoretical discussion of the rationale behind the design, the reader is referred to the accompanying paper [1].

1 Introduction

TRIVIUM is a hardware oriented synchronous stream cipher. It was designed as an exercise in exploring how far a stream cipher can be simplified without sacrificing its security, speed or flexibility. While simple designs are more likely to be vulnerable to simple, and possibly devastating, attacks (which is why we strongly discourage the use of TRIVIUM at this stage), they certainly inspire more confidence than complex schemes, if they survive a long period of public scrutiny despite their simplicity.

The next section provides a complete description of the stream cipher proposal. Sect. 3 discusses the performance of the cipher, and Sect. 4 briefly touches some security aspects of the algorithm. A more theoretical discussion of the rationale behind the design can be found in a separate paper [1].

2 Specifications

TRIVIUM is a synchronous stream cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initial value (IV). As for most stream ciphers, this process consists of two phases: first the internal state of the cipher is initialized using the key and the IV, then the state is repeatedly updated and used to generate key stream bits. We first describe this second phase.

* The work described in this paper has been partly supported by the European Commission under contract IST-2002-507932 (ECRYPT).

** F.W.O. Research Assistant, Fund for Scientific Research – Flanders (Belgium).

Parameters	
Key size:	80 bit
IV size:	80 bit
Internal state:	288 bit

Table 1. Parameters of TRIVIUM

2.1 Key stream generation

The proposed design contains a 288-bit internal state denoted by $(s_1, \ldots, s_{288})$. The key stream generation consists of an iterative process which extracts the values of 15 specific state bits and uses them both to update 3 bits of the state and to compute 1 bit of key stream z_i. The state bits are then rotated and the process repeats itself until the requested $N \leq 2^{64}$ bits of key stream have been generated. A complete description is given by the following simple pseudo-code:

for $i = 1$ to N **do**
  $t_1 \leftarrow s_{66} + s_{93}$
  $t_2 \leftarrow s_{162} + s_{177}$
  $t_3 \leftarrow s_{243} + s_{288}$
  $z_i \leftarrow t_1 + t_2 + t_3$
  $t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
  $t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
  $t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

Note that here, and in the rest of this document, the '+' and '$\cdot$' operations stand for addition and multiplication over GF(2) (i.e., XOR and AND), respectively. A graphical representation of the key stream generation process can be found in Fig. 1.

2.2 Key and IV setup

The algorithm is initialized by loading an 80-bit key and an 80-bit IV into the 288-bit initial state, and setting all remaining bits to 0, except for s_{286}, s_{287}, and s_{288}. Then, the state is rotated over 4 full cycles, in the same way as explained above, but without generating key stream bits. This is summarized in the pseudo-code below:

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$
$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
for $i = 1$ to $4 \cdot 288$ **do**

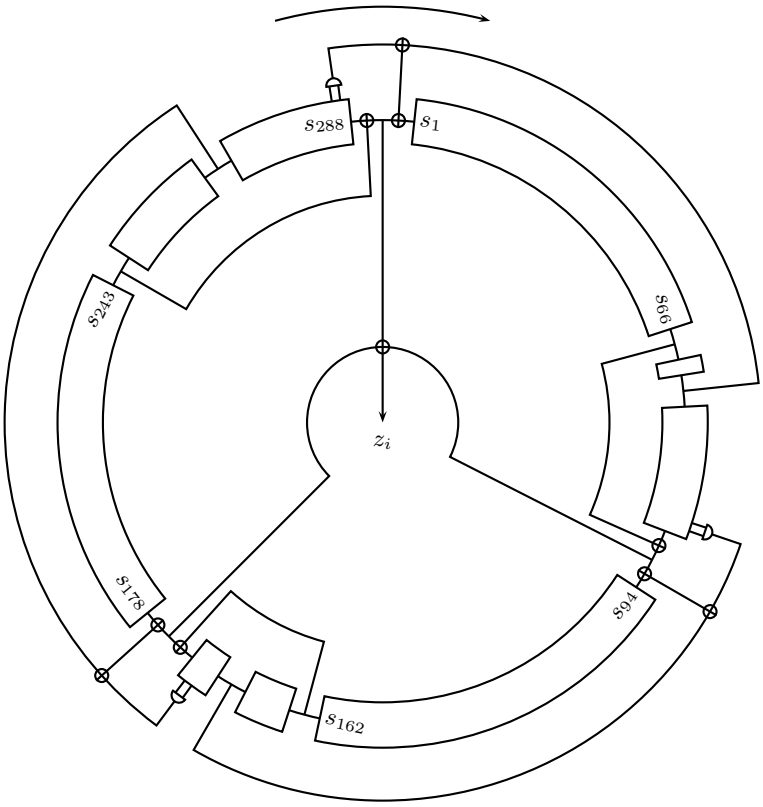

Fig. 1. TRIVIUM

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

3 Implementation

3.1 Hardware

TRIVIUM is a hardware oriented design focussed on flexibility. It aims to be compact in environments with restrictions on the gate count, power-efficient on platforms with limited power resources, and fast in applications that require high-speed encryption.

The requirement for a compact implementation suggests a bit-oriented approach. It also favors the use of a nonlinear internal state, in order not to waste all painfully built up nonlinearity at the output of the key stream generator. In order to allow power-efficient and fast implementations, the design must also provide a way to parallelize its operations. In the case TRIVIUM, this is done by ensuring that any state bit is not used for at least 64 iterations after it has been modified. This way, up to 64 iterations can be computed at once, provided that the 3 AND gates and 11 XOR gates in the original scheme are duplicated a corresponding number of times. This allows the clock frequency to be divided by a factor 64 without affecting the throughput.

Based on the figures stated in [2] (i.e., 12 NAND gates per Flip-flop, 2.5 gates per XOR, and 1.5 gates per AND), we can compute an estimation of the gate count for different degrees of parallelization. The results are listed in Table. 2.

Components	1-bit	8-bit	16-bit	32-bit	64-bit
Flip-flops:	288	288	288	288	288
AND gates:	3	24	48	96	192
XOR gates:	11	88	176	352	704
NAND gate count:	3488	3712	3968	4480	5504

Table 2. Estimated gate counts of 1-bit to 64-bit hardware implementations

3.2 Software

Despite the fact that TRIVIUM does not target software applications, the cipher is still reasonably efficient on a standard PC. The measured performance of the reference C-code on an 1.5 GHz Xeon processor can be found in Table 3.

Operation	
Stream generation:	12 cycles/byte
Key setup:	55 cycles
IV setup:	2050 cycles

Table 3. Measured performance on an Intel® Xeon™ CPU 1.5 GHz

4 Security

In this section we briefly discuss some of the cryptographic properties of TRIVIUM. For a more detailed analysis of the cipher, we refer to the paper [1].

The security requirement we impose on TRIVIUM is that any type of cryptographic attack should not be significantly easier to apply to TRIVIUM than to any other imaginable stream cipher with the same external parameters (i.e., any cipher capable of generating up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit IV). Unfortunately, this requirement is not easy to verify, and the best we can do is to provide arguments why we believe that certain common types of attacks are not likely to affect the security of the cipher.

4.1 Correlations

When analyzing the security of a synchronous stream cipher, a cryptanalyst will typically consider two different types of correlations The first type are correlations between linear combinations of key stream bits and internal state bits, which can potentially lead to a complete recovery of the state. The second type, exploited by distinguishing attacks, are correlations between the key stream bits themselves.

Obviously, linear correlations between key stream bits and internal state bits are easy to find, since z_i is simply defined to be equal to $s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$. However, as opposed to LFSR based ciphers, TRIVIUM's state evolves in a nonlinear way, and it is not clear how the attacker should combine these equations in order to efficiently recover the state.

An easy way to find correlations of the second type is to follow linear trails through the cipher and to approximate the outputs of all encountered AND gates by 0. However, the positions of the taps in TRIVIUM have been chosen in such a way that any trail of this specific type is forced to approximate at least 72 AND gate outputs. An example of a correlated linear combination of key stream bits obtained this way is

$$z_1 + z_{16} + z_{28} + z_{43} + z_{46} + z_{55} + z_{61} + z_{73} + z_{88} + z_{124} + z_{133} + z_{142} + z_{202} + z_{211} + z_{220} + z_{289}\,.$$

If we assume that the correlation of this linear combination is completely explained by the specific trail we considered, then it would have a correlation

coefficient of 2^{-72}. Detecting such a correlation would require at least 2^{144} bits of key stream, which is well above the security requirement.

Other more complicated types of linear trails with larger correlations might exist, but at this stage it seems unlikely that these correlations will exceed 2^{-40}. This issue is discussed in more details in the paper [1].

4.2 Period

Because of the fact that the internal state of TRIVIUM evolves in a nonlinear way, its period is hard to determine. Still, a number of observations can be made. First, if the AND gates are omitted (resulting in a completely linear scheme), one can show that any key/IV pair would generate a stream with a period of at least $2^{96-3} - 1$. This has no immediate implications for TRIVIUM itself, but it might be seen as an indication that the taps have been chosen properly.

Secondly, TRIVIUM's state is updated in a reversible way, and the initialization of $(s_{178}, \ldots, s_{288})$ prevents the state from cycling in less than 111 iterations. If we believe that TRIVIUM behaves as a random permutation after a sufficient number of iterations, then all cycle lengths up to 2^{288} would be equiprobable, and hence the probability for a given key/IV pair to cause a cycle smaller than 2^{80} would be 2^{-208}.

4.3 Guess and Determine attacks

In each iteration of TRIVIUM, only a few bits of the state are used, despite the general rule-of-thumb that sparse update functions should be avoided. As a result, guess and determine attacks are certainly a concern. A straightforward attack would guess $(s_{25}, \ldots, s_{93})$, $(s_{97}, \ldots, s_{177})$, and $(s_{244}, \ldots, s_{288})$, 195 bits in total, after which the rest of the bits can immediately be determined from the key stream. Further research should be conducted to examine to which extent more sophisticated attacks can reduce this number.

4.4 Algebraic attacks

TRIVIUM seems to be a particularly attractive target for algebraic attacks. The complete scheme can easily be described with extremely sparse equations of low degree. However, its state does not evolve in a linear way, and hence the efficient linearization techniques used to solve the systems of equations generated by LFSR based schemes will be hard to apply. However, other techniques might be applicable and their efficiency in solving this particular system of equations needs to be investigated.

4.5 Resynchronization attacks

Another type of attacks are resynchronization attacks, where the adversary is allowed to manipulate the value of the IV, and tries to extract information about

the key by examining the corresponding key stream. TRIVIUM tries to preclude this type of attacks by cycling the state a sufficient number of times before producing any output. It can be shown that each state bit depends on each key and IV bit in a nonlinear way after two full cycles (i.e., $2 \cdot 288$ iterations). We expect that two more cycles will suffice to protect the cipher against resynchronization attacks.

5 Conclusion

TRIVIUM is a simple synchronous stream cipher which seems to be particularly well suited for application which require a flexible hardware implementation. It is clearly in an experimental stage though, and further research will reveal whether it meets its security requirements or not.

Remarks concerning the name. The word *trivium* is Latin for "the three-fold way", and refers to the three-fold symmetry of TRIVIUM. The adjective *trivial*, which was derived from it, has a connotation of simplicity, which is also one of the characteristics of TRIVIUM. Moreover, with some imagination, one might recognize the shape of a Trivial Pursuit board in Fig. 1 (while we admit that in this respect "Merecedes" would have been a more appropriate name). Finally, the name provides a nice title for a subsequent cryptanalysis paper: "Three Trivial Attacks on Trivium".

References

1. C. De Cannière and B. Preneel, "TRIVIUM – A Stream Cipher Construction Inspired by Block Cipher Design Principles," to be uploaded to `http://www.ecrypt.eu.org/stream` soon.
2. J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede, "Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism," in *ECRYPT Workshop, SASC – The State of the Art of Stream Ciphers*, pp. 327–333, 2004.