FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK–ESAT/SCD-COSIC
KASTEELPARK ARENBERG 10
B–3001 HEVERLEE

KATHOLIEKE
UNIVERSITEIT
LEUVEN

Security and Performance Analysis of ARIA

VERSION 1.2

FINAL REPORT

Alex Biryukov
Christophe De Cannière
Joseph Lano
Siddika Berna Ors
Bart Preneel

Dept. Electrical Engineering-ESAT/SCD-COSIC
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium

January 7, 2004

Contents

Introduction

ETRI has requested the COSIC Division of the Department of Electrical Engineering-ESAT of the K.U.Leuven to perform an evaluation of the block cipher Aria. This report summarizes the results of the analysis.

The study involved two aspects:

- An evaluation of the security of the block cipher against different types of attacks and an examination of ETRI's security evaluation.
- An efficiency evaluation of the block cipher, both in software and in hardware.

Implementation-based attacks such as timing attacks or power attacks are not considered in this report. The security analysis is exclusively based on the mathematical description of Aria, as specified in [14].

Document history

- Version 1.2, Jan. 7, 2003
- Version 1.1, Dec. 12, 2003
- Version 1.0, Nov. 15, 2003

Chapter 1

Security Evaluation

In this chapter, the security of Aria with respect to different types of attacks is evaluated. Both standard approaches and dedicated attacks will be discussed.

1.1 Short Description

Aria is a 128-bit block cipher accepting keys of 128, 192, or 256 bit. The cipher is a substitution permutation network (SPN) and uses an involutional binary 16×16 matrix in its diffusion layer. The substitution layer consists of sixteen 8×8-bit S-boxes based on the inversion in $GF(2^8)$. The number of rounds is 10, 12, or 14, depending on the key length. For a full description of the cipher, we refer to the specifications [14].

1.2 Classical Linear and Differential Cryptanalysis

The resistance of a block cipher to classical linear [12] and differential [2] cryptanalysis can be estimated by analyzing the maximum probability of linear and differential trails. In the case of SPN-ciphers, one can easily find a bound on these probabilities by considering the branch number of the diffusion layer (8 in the case of Aria) and the maximum linear and differential probabilities of the S-boxes (2^{-6}). The resulting bounds, calculated by the designers in [14], have been checked independently. From these results, one can conclude that Aria contains no classical 6-round linear or differential trail with a probability exceeding 2^{-144} and that the best 5-round trail has a probability of at most 2^{-102}. Considering the 128-bit block size, this suggests that 8-rounds of Aria provide sufficient resistance against these two classical attacks.

Note that this conclusion neglects the fact that different trails might cluster (causing differentials or linear hulls). This effect does occur in Aria, mainly because of its word-oriented structure. However, we do not believe that this will lead to significantly more efficient attacks than the truncated differential and dedicated linear attacks of Sections 1.3 and 1.4 (which in fact cover this clustering effect to a certain degree).

1.3 Truncated Differential Cryptanalysis

Truncated differentials [11] have lead to some successful attacks on ciphers with a pronounced word-oriented structure. Given that ARIA has this property (just as most modern ciphers), we investigate how these attacks apply to reduced-round versions of ARIA.

1.3.1 Truncated Differentials of Type $1 \to 7 \to 1$

The most straightforward class of truncated differentials in ARIA, is of the type $1 \to 7 \to 1 \to 7 \ldots$, where the numbers 1 and 7 designate the number of *active S-boxes* at each round. For example:

$$\begin{array}{rlll}
\Delta P = & \texttt{A0000000 00000000} & 1 & \text{round } 1 \\
 & \texttt{000BB0B0 BB000BB0} & 7 & \\
 & \texttt{C0000000 00000000} & 1 & \\
 & \texttt{000DD0D0 DD000DD0} & 7 & \\
 & \ldots & & \\
 & \texttt{000EE0E0 EE000EE0} & 7 & \text{round } r \\
\Delta C = & \texttt{000FG0H0 IJ000KL0} & &
\end{array}$$

All patterns shown in the example represent differences between texts entering the S-box layer, except for ΔC, which is taken at the output of the S-box layer.

The $1 \to 7$ transitions occur with probability 1, but the $7 \to 1$ transitions only take place when all 7 non-zero byte differences remain equal after the S-box layer. Taking into account the two different types of S-boxes S and S^{-1}, the probability that this happens is found[1] to be about $2^{-(2\cdot7+8+3\cdot7)} = 2^{-43}$. Using these two transitions, one could build a 7-round differential $1 \to 7 \ldots 1$, or an 8-round differential $1 \to 7 \ldots 7$, both with probability 2^{-129}. However, this does not seem to suffice for mounting a successful attack, given that the block size is 128 bits, and that the 1-byte difference at the top does not allow for additional filtering.

[1] The derivation uses the fact that a fixed non-zero difference at the input of a given 8×8-bit S-box can cause at most 2^7 different differences at the output. Consequently, the probability that two equal differences entering two identical S-boxes remain equal is 2^{-7}. This probability is 2^{-8} when the S-boxes are different (and sufficiently independent).

By shifting the previous pattern by one round, we find another 7-round differential of type $7 \rightarrow 1 \ldots 7$, but with a slightly reduced probability of 2^{-134}:

$$\begin{array}{rlll} \Delta P = & \texttt{000AB0C0\,DE000FG0} & 7 & \text{round 1} \\ & \texttt{H0000000\,00000000} & 1 & \\ & \texttt{000II0I0\,II000II0} & 7 & \\ & \texttt{J0000000\,00000000} & 1 & \\ & \qquad\qquad\qquad\ldots & & \\ & \texttt{000KK0K0\,KK000KK0} & 7 & \text{round 7} \\ \Delta C = & \texttt{000LM0N0\,PQ000RS0} & & \end{array}$$

The probability reduction is due to the fact that we do not require the differences in the 7 active plaintext bytes to be equal, which makes the first $7 \rightarrow 1$ transition a bit more expensive.

This time, the larger number of active bytes at the top allows the following 7-round attack:

- Encrypt 2^{25} pools of 2^{56} texts, which take on all the possible values in the 7 active bytes and are constant in the rest (in total we need 2^{81} chosen texts). These texts contain about $2^{25} \cdot 2^{111} = 2^{136}$ pairs with a difference of the form `000AB0C0 DE000FG0`, and we would therefore expect that about 4 of these pairs satisfy the truncated differential proposed above.
- For each pool, sort the ciphertexts according to the values of the 9 bytes corresponding to the zeros in the output differential, and return all pairs with identical values in these positions. In total, we expect to find $2^{136}/2^{72} = 2^{64}$ candidate pairs.
- Each candidate pair suggests on average about $2^8 \times 2^8$ possible values for the 2×7 active key bytes used in the first and the last key addition. In total we get 2^{80} suggestions for 14 key bytes (112 bits). Since 4 pairs satisfy the truncated differential, we know that the correct key is suggested at least 4 times. The probability that any other key appears 4 times is negligible.[2]

We make two remarks here. The first is that the 7-round truncated differential used in this attack has probability considerably smaller than 2^{-128}. Unlike regular differentials, truncated differentials with such small probabilities can still be useful, however. This was already demonstrated in [11].

A second remark concerns the success probability of the attack. We assumed that the correct key suggestion would appear exactly four times. This will of course not always be

[2] The set of 2^{80} key suggestions contains about $2^{4\cdot 80}/4!$ different quartets. Assuming that the suggestions look random, the probability that such a quartet has four identical 112-bit values is $2^{-3\cdot 112}$. From this we deduce that the probability of finding a quartet of identical keys in the set is about $2^{-3\cdot 112} \cdot 2^{4\cdot 80}/4! \approx 2^{-20}$.

the case. In practice, we will proceed in a slightly different way: we will encrypt the pools one by one, and repeatedly check if a suggestion appears for the fourth time. As soon as this happens, we stop. The derivation given above shows that the average number of pools needed is 2^{25}. This number can be slightly reduced by already checking suggestions appearing only three or even two times. The time needed to test wrong suggestions will be negligible compared to the full attack.

From the attack outlined above, we conclude that 14 round key bytes of 7-round Aria can be recovered with an average data and time complexity of 2^{81}. Note however that this attack requires a very large amount of memory to store the 2^{80} key suggestions. This memory could in principle be reduced if the key schedule allowed to find simple relations between the round key bytes used before the first and after the seventh round. This does not seem to be the case for Aria however.

1.3.2 Truncated Differentials of Type $4 \rightarrow 4$

A second interesting truncated differential propagates through 4 bytes of each round as follows:[3]

$$\begin{array}{rlll} \Delta P = & \mathtt{ABCD0000\,00000000} & 4 & \text{round } 1 \\ & \mathtt{EEEE0000\,00000000} & 4 & \\ & \mathtt{FFFF0000\,00000000} & 4 & \\ & & \cdots & \\ & \mathtt{GGGG0000\,00000000} & 4 & \text{round } r \\ \Delta C = & \mathtt{HIJK0000\,00000000} & & \end{array}$$

The probability that 4 random differences in the plaintext yield equal differences after the first S-box layer is 2^{-24}. The probability hat this property is preserved in the next round is 2^{-21} (note that the probability is increased by the fact that the active bytes enter 4 identical S-boxes). Since the transition in the last round of the differential trail occurs with probability 1, we conclude that this truncated differential allows to cover r rounds with a probability of $2^{-24-21\cdot(r-2)}$. For 7 rounds, we obtain 2^{-129}, just as in the previous subsection. This 7-round differential can be exploited in exactly the same way:

- Encrypt $5 \cdot 2^{66}$ pools of 2^{32} texts, which take on all the possible values in the first 4 bytes and are constant in the others (in total we need about 2^{100} chosen texts). These texts contain about $5 \cdot 2^{66} \cdot 2^{63} = 5 \cdot 2^{129} \approx 2^{131}$ pairs with a difference of the form `ABCD0000 00000000`, and we would therefore expect that 5 of these pairs satisfy the 7-round differential proposed above.

[3]The 1st four bytes may be replaced by other invariant subspaces.

- For each pool, sort the ciphertexts according to the values of the 12 last bytes of the ciphertexts, and return all pairs with identical values in these positions. In total, we expect to find about $2^{131}/2^{96} = 2^{35}$ candidate pairs.

- Each candidate pair suggests on average about $2^8 \times 2^8$ possible values for the 2×4 active key bytes used in the first and the last key addition. In total we get 2^{51} suggestions for 8 key bytes (64 bits). Since 5 pairs satisfy the truncated differential, we know that the correct key is suggested at least 5 times. The probability that any other key appears 5 times is small (about $2^{-4 \cdot 64} \cdot 2^{5 \cdot 51}/5! \approx 2^{-8}$).

The attack described above is clearly less efficient than the previous one (data and time complexity of 2^{100}). It requires less memory, however (2^{51} key suggestions need to be stored).

1.3.3 Truncated Differentials — Multiset Approach

In this subsection, we discuss an alternative way of exploiting the truncated differential of type $4 \rightarrow 4$. This time we also consider the values (and not only the differences) of the four leading bytes. The alternative approach described in this subsection does not immediately lead to a more efficient attack, but it illustrates some unexpected properties of Aria.

Let us first consider a single plaintext P_0 which has 4 equal values `AAAA` in the first 4 bytes at the input of the first S-box layer (i.e., after the first key addition). The probability that this happens for a random plaintext is 2^{-24}. Furthermore, let us assume that this special property is repeated over $r-1$ consecutive rounds (probability $2^{-24\cdot(r-1)}$). We now consider a second plaintext $P_1 = P_0 \oplus \Delta P$ with $\Delta P =$ `DDDD0000 00000000`. Due to the structure of the diffusion layer, the plaintext P_1 automatically inherits the special properties of P_0, and after r rounds we obtain two ciphertexts C_0 and C_1 with identical values in the last 12 bytes. The reason for this is that the values of the first 4 bytes of the intermediate texts do not influence the last 12 bytes, as long as these 4 values are equal. This suggests the following 6-round attack:

- Consider all 2^{120} plaintexts P_0 with a constant value `A` in the first byte (e.g., `A` = `0`). On the average, we expect[4] that 1 of these plaintexts will have the desired properties for $r = 6$. For each P_0, construct a corresponding plaintext $P_1 = P_0 \oplus \Delta P$ with $\Delta P =$ `DDDD0000 00000000` for a constant `D` (e.g., `D` = `1`).

- Encrypt the 2^{120} plaintext pairs (P_0, P_1) and check whether the corresponding ciphertexts C_0 and C_1 have equal values in the last 12 bytes. This will be the case for about

[4] Averaged over all keys, we expect 1 such "special" plaintext per key. However, these special plaintexts might not exist for all keys. If no such plaintext is found, the attack can be repeated using four other active S-boxes, e.g., $\Delta P =$ `0000DDDD 00000000`.

$2^{-96} \cdot 2^{120} = 2^{24}$ pairs. In order to filter out all wrong pairs, construct 2^{24} additional plaintexts $P_2 = P_0 \oplus \Delta P'$ with $\Delta P' = \texttt{EEEE0000 00000000}$ and $\texttt{E} \neq \texttt{D}$ (e.g., $\texttt{E} = \texttt{2}$). If P_0 has the properties described above, then the last 12 bytes of C_2 should remain constant in this case as well. For a random plaintext this is extremely unlikely.

- The correct triple (C_0, C_1, C_2) can be used to recover the first 4 key bytes of the final key layer, and the value of P_0 provides 24 bit of information about the first 4 key bytes of the initial key addition.

The distinguisher used in this alternative 6-round attack is clearly much stronger than necessary. It is therefore natural to try to extend the attack with an additional round. A possible approach is inspired by multiset attacks (also known as square, saturation, or integral attacks) and is based on the following observation: if we start from the special plaintext P_0 described above and encrypt a set of plaintexts $P_{\texttt{D}} = P_0 \oplus \Delta P_{\texttt{D}}$ with $\Delta P_{\texttt{D}} = \texttt{DDDD0000 00000000}$ and $\texttt{D} = \texttt{0}, \ldots, \texttt{255}$, then we obtain a set of ciphertexts with 4 saturated and 12 constant bytes after 6 rounds. The constant bytes disappear after an additional round, but the saturation of the first 4 bytes is preserved, and this can be detected. In order to avoid having to encrypt 256 texts for all 2^{120} P_0 candidates (which would require the full codebook of 2^{128} texts), we proceed as follows:

- Consider all 2^{120} plaintexts P_0 with a constant value $\texttt{A}$ in the first byte (e.g., $\texttt{A} = \texttt{0}$). On the average, we expect that 1 of these plaintexts will have the desired properties for $r = 6$. For each P_0, construct a corresponding plaintext $P_1 = P_0 \oplus \Delta P_1$ with $\Delta P_1 = \texttt{11110000 00000000}$.
- Encrypt the 2^{120} plaintext pairs (P_0, P_1) and check whether the corresponding ciphertexts C_0 and C_1 have equal values in any of the 4 first bytes. If a collision occurs, we know that P_0 does not have the desired properties.
- For all remaining pairs, encrypt an additional plaintext $P_2 = P_0 \oplus \Delta P_2$ with $\Delta P_2 = \texttt{22220000 00000000}$ and check again for collisions. Repeat this procedure until a single pair remains.

It can be shown that the 7-round attack described above requires slightly less than 2^{124} adaptively chosen plaintexts. This is much more than in the previous attacks. Note however that the distinguisher used in this attack is still much stronger than necessary.

The data complexity can be reduced by taking into account additional properties of the diffusion layer. We again start from the multisets described above, but decrease the number of round by 1, i.e., we want the 256 texts after the 4th diffusion layer to start with four equal bytes, taking on all values $x = 0 \ldots 255$. This happens with probability 2^{-96}. Denoting the 16 bytes of the texts after the 5th diffusion layer by $y_1, \ldots, y_{16}$, we can write:

$$\begin{aligned} y_i &= S(x \oplus k_i) \oplus c_i\,, && 1 \leq i \leq 4 \\ y_i &= S(x \oplus k_{m_i}) \oplus S(x \oplus k_{n_i}) \oplus c_i\,, && 5 \leq i \leq 16 \end{aligned}$$

where the c_i are constant for each multiset. This implies that the first four bytes of the multisets are saturated (we already used this property above), but also that every value in each of the 12 last bytes appears an even number of times (because $y_i(x) = y_i(x \oplus k_{m_i} \oplus k_{n_i})$). This property is preserved after the 6th S-box layer and generates balanced bytes at the input of the 7th S-box layer (cfr. SASAS [4]). This suggests the following 7-round attack:

- Consider 2^{96} random plaintexts P_0 with a constant value $\mathtt{A}$ in the first byte (e.g., $\mathtt{A} = \mathtt{0}$). On the average, we expect that 1 of these plaintexts will have the desired properties for $r = 5$. For each P_0, construct a set of plaintexts $P_{\mathtt{D}} = P_0 \oplus \Delta P_{\mathtt{D}}$ with $\Delta P_{\mathtt{D}} = \mathtt{DDDD0000\,00000000}$ and $\mathtt{D} = \mathtt{0}, \ldots, \mathtt{255}$.
- Encrypt the $2^8 \cdot 2^{96}$ plaintexts. For each multiset and for each byte of the ciphertext, determine the value of the key byte used in the last round such that the corresponding byte at the input of the 7the S-box layer is balanced. On the average, we expect that each multiset will suggest one key. Finally check whether the suggested key results in four saturated bytes after the 6th S-box layer. This is very unlikely to occur for a random multiset.

The attack requires 2^{104} plaintexts and has a time complexity of about 2^{112}. It seems that the distinguisher in this attack is still very strong however, and this suggests that there might be ways to extend the attack with one more round.

1.4 Dedicated Linear Attack

In this section we focus on the sum (in $GF(2^8)$) of the first four bytes of the state. The core of the attack is a distinguisher based on the following observation: if we denote the bytes at the input and the output of an S-box layer by x_i and y_i, respectively, then

$$P(\sum_{i=1}^{4} y_i = 0 \mid \sum_{i=1}^{4} x_i = 0) \approx 3 \cdot 2^{-8} = 2^{-8} + 2^{-7}\,. \tag{1.1}$$

This property does not depend on the choice of the S-boxes and holds as long as the 4 S-boxes involved in the expression above are equal. This can easily be explained by noting that, out of the 2^{24} combinations (x_0, x_1, x_2, x_3) which satisfy $x_0 + x_1 + x_2 + x_3 = 0$, about $3 \cdot 2^{16}$ are of the form (a, a, b, b), (a, b, a, b), or (a, b, b, a). These special patterns are preserved by the S-box layer, such that the y_i sum to 0 as well.

In order to build an r-round distinguisher based on the observation above, we also need to cross diffusion layers and key additions. Analyzing the diffusion matrix used in ARIA, one can easily see that this transformation preserves the sum of the first four bytes, but in order to cross the key additions, we need the first four bytes of the round keys to sum to zero. This implies that the attack will only succeed for a limited number of weak keys.

We first analyze the strength of a distinguisher spanning r S-box layers and $r-1$ key additions. The input bytes of the first S-box layer are denoted by x_i^1, and the outputs after the last S-box layer by y_i^r. If the first 4 bytes of each of the $r-1$ round keys sum to zero — this happens with probability $2^{-(r-1)\cdot 8}$ for random round keys — one can recursively derive that

$$P(\sum_{i=1}^{4} y_i^r = 0 \mid \sum_{i=1}^{4} x_i^1 = 0) \approx 2^{-8} + 2^{-r\cdot 7}\,. \tag{1.2}$$

In a next step we append additional key layers at the top and the bottom of the distinguisher. The bytes of the round keys of these two layers are denoted by k_i^0 and k_i^r respectively. In order to distinguish the cipher from a random permutation, an attacker could calculate $s_0 = \sum_{i=1}^{4} (x_i^1 + k_i^0)$ and $s_r = \sum_{i=1}^{4} (y_i^r + k_i^r)$ for n known P/C (plaintext/ciphertext) pairs, and construct a distribution table of (s_0, s_r). In the random case, we would expect each pair (s_0, s_r) to occur with probability 2^{-16}, but for Aria we can deduce from (1.2) that

$$P(\sum_{i=1}^{4} y_i^r = 0 \text{ and } \sum_{i=1}^{4} x_i^1 = 0) \approx 2^{-16} + 2^{-8}\cdot 2^{-r\cdot 7}\,, \tag{1.3}$$

which directly implies that the pair $(\sum_{i=1}^{4} k_i^0, \sum_{i=1}^{4} k_i^r)$ has a slightly higher probability. This causes a peak in the distribution table which will stick out of the surrounding noise as soon as

$$2^{-8}\cdot 2^{-r\cdot 7} > Q^{-1}(2^{-16})\cdot\sqrt{\frac{2^{-16}}{n}}\,, \tag{1.4}$$

or

$$n > 18\cdot 2^{2\cdot r\cdot 7} \tag{1.5}$$

Since n is limited by the block size of 128 bits, we conclude that the distinguisher is in principle effective up to $r = 8$ rounds. However, in order to be useful, the time complexity of the attack cannot exceed the size of the weak key class (if it does, a simple exhaustive search would be more efficient). Since the time complexity includes at least the time required to encrypt the necessary data, we find the following additional condition for the 128-bit key version:

$$18\cdot 2^{2\cdot r\cdot 7} < 2^{128}\cdot 2^{-(r-1)\cdot 8}\,, \tag{1.6}$$

or $r \le 5$. Similarly, for the 192-bit and 256-bit version we obtain $r \le 8$ and $r \le 11$ (note however that r cannot exceed 8 because of the block size).

We can now append two more rounds to the current distinguisher by guessing the first 4 key bytes of two additional key layers at the top and the bottom of the cipher (64 bits in total). For each guess, we can apply a partial encryption at the top and a partial decryption at the bottom, and compute the distribution table described above. If no peak is observed, we know that the guess was probably wrong. In order to filter out all wrong guesses, we need to strengthen the distinguisher. Considering the fact that the distinguisher is applied 2^{64} times and that $Q^{-1}(2^{-16} \cdot 2^{-64}) \approx 10$, we find

$$n > 100 \cdot 2^{2 \cdot r \cdot 7} \approx 2^{2 \cdot r \cdot 7 + 7} . \tag{1.7}$$

Hence, for attacking a 128-bit key version of Aria reduced to 7 rounds, using a 5-round distinguisher, 2^{77} known P/C pairs should suffice.

In order to avoid having to process all n plaintexts for each of the 2^{64} key guesses, we will build tables of counters and gradually transform them into distribution tables by guessing the key bytes one by one and partially evaluating the sums s_0 and s_r. The first table contains 2^{64} entries and is constructed by running through the n P/C pairs and increasing the counter corresponding to the value of the first 4 bytes of the plaintext and the ciphertext (64 bits in total). In the next step, we guess the 2 first key bytes at the top, apply a partial encryption, compute the sum of the first 2 terms of s_0, and create a table containing 2^{56} counters corresponding to the value of this sum, the 2 remaining bytes of the plaintext, and the first 4 bytes of the ciphertext (56 bits in total). In the next step, one more key byte is guessed and the previous table is used to compute a smaller table of 2^{48} counters, and so on. It can be shown that the total computational complexity of this approach is about $n + 2^{64} \cdot 2^{24}$ (in stead of $n \cdot 2^{64}$). The attack requires 2^{64} counters to be stored in memory.

The 7-round attack on the 128-bit key version provides the values of 2×4 round key bytes at the top and the bottom and recovers the sum of the first 4 key bytes used in the 6 inner key layers (112 bits in total). Table 1.1 summarizes the complexities for different key sizes.

Table 1.1: Complexities of a dedicated linear attack.

Key length:	128	192	256
Rounds:	7	10	10
Weak keys:	2^{96}	2^{136}	2^{200}
Data:	2^{77}	2^{119}	2^{119}
Memory[a]:	2^{64}	2^{64}	2^{64}
Counting complexity:	2^{88}	2^{88}	2^{88}
Recovered round key bits:	112	136	136

[a] Measured in number of counters. For the 128-bit version, 16-bit counters should suffice. The 192-bit and 256-bit versions require 64-bit counters.

1.5 Boomerang Attacks

Boomerang attacks [15] are useful when a cipher contains (regular or truncated) differentials with relatively high probability, but which cover only half the cipher and can not be combined. This does not seem to be the case for ARIA however: because of the involutional structure, differentials are usually easy to combine. We therefore do not expect improvements from boomerang attacks compared to normal (truncated) differential attacks. This is also confirmed by the calculations made by the designers for boomerang attacks based on regular differentials.

1.6 Square/Multiset/Collision Attacks

The best attacks on RIJNDAEL are multiset attacks [9] and collision attacks [10]. ARIA, whose design has many similarities with RIJNDAEL, seems to provide more resistance to these attacks, however. Both attacks exploit the relatively slow diffusion in a single round of RIJNDAEL. ARIA's diffusion layer has a branch number of 8 (instead of 5 for RIJNDAEL), and this affects the multiset and collision attacks in two ways. First, the faster diffusion does not allow a 3-round distinguisher as in RIJNDAEL, and secondly, extending the (2-round) distinguisher at the top and the bottom requires much more key bytes to be guessed. Therefore, it seems unlikely that a classical multiset attack would be able to cover more than 5 or 6 rounds depending on the key size.

Note however that the ideas of multiset attacks can be useful to extend other attacks, as demonstrated in Section 1.3.3.

1.7 Slide Attacks

Regular slide attacks [3] do not seem to apply to ARIA because of the irregular structure of the key schedule. Twisted slide attack as applied to the involutional cipher KHAZAD [6] are not likely to be efficient either: this attack would require at least two round keys to be equal, which is very unlikely given the size of ARIA's round keys.

1.8 Algebraic Structure

The algebraic structure of ARIA and RIJNDAEL are very similar. Both ciphers use S-boxes based on the inverse over $GF(2^8)$, and the diffusion layers of both ciphers have very simple representations in $GF(2^8)$. As a result, ARIA inherits most of the algebraic properties of RIJNDAEL, e.g.:

Dual ciphers. Many different dual ciphers as described in [1, 5] can be constructed by changing the affine transformations B and the constants $CK_{1,\dots,3}$. Since ARIA is not self-dual in a non-trivial way, this observation does not affect the security of the cipher as such. On the other hand, these dual ciphers can help to protect implementations against side channel attacks.

Potential vulnerability against XSL. The rounds of ARIA contain exactly the same number of S-boxes as RIJNDAEL, and the difference in the key schedule is small (48 vs. 40 S-boxes). Hence, if one could design a successful attack algorithm against RIJNDAEL based on XSL [7], it is very likely that it would apply to ARIA as well.

Big Encryption Scheme (BES). ARIA can be transformed into a 128-*byte* block cipher composed out of very simple operations in $GF(2^8)$ as described in [13]. This might speed up a hypothetical XSL attack.

Despite its similarity with RIJNDAEL, ARIA might possibly introduce new weaknesses. One interesting observation is that the cipher can be split into two parts with particularly simple representations in $GF(2^8)$ (see Figure 1.1). The construction starts from the observation

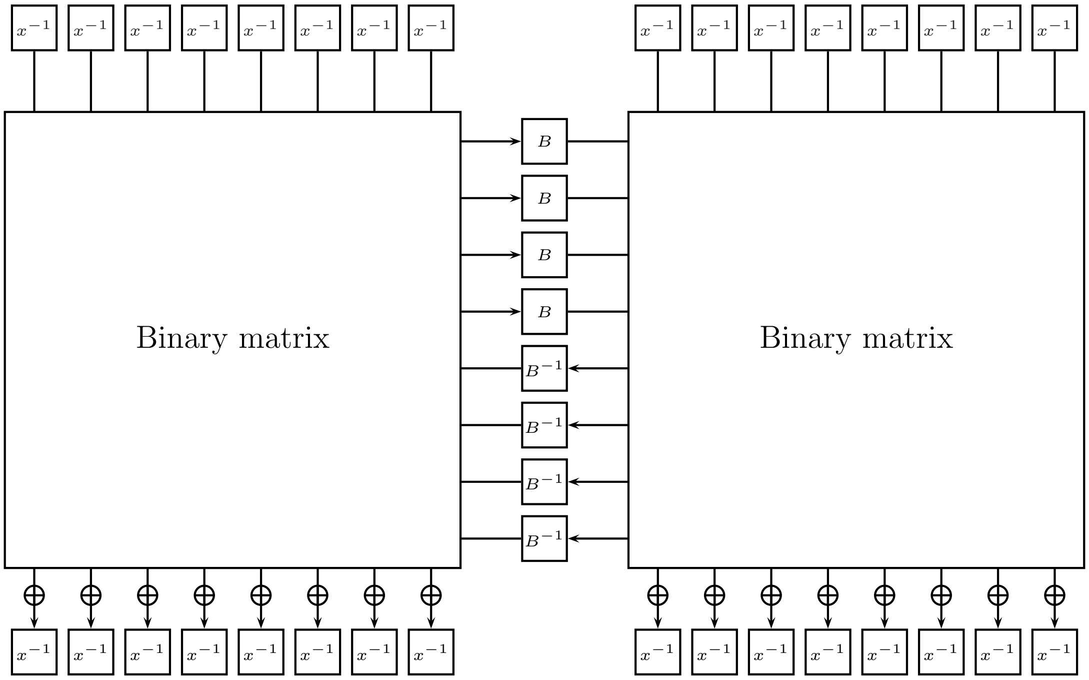

Figure 1.1: Equivalent representation

that the left half of the output of the diffusion layer $(y_0, \dots, y_7)$ depends on only four independent linear combinations of the right half of the input $(x_8, \dots, x_{15})$. This follows immediately from the fact that the upper right quarter of the binary diffusion matrix has rank 4. The same holds for the interaction between the right half of the output and the left half of the input. This property allows the linear diffusion layer to be split into two binary parts where all interaction between the parts goes through 4 bytes only (in both directions). A second observation is that the linear transformations B and B^{-1} at the output of the S-layers and at the input of the S^{-1}-layers commute with the binary operations of the diffusion layer, i.e., $B(x_i) + B(x_j) + \ldots + B(x_k) = B(x_i + x_j + \ldots + x_k)$. This also implies that an extra layer of B^{-1}-transforms can be freely inserted at the inputs of the left part of the diffusion layer, provided that a corresponding layer of B-transforms is inserted at the outputs. Since these extra layers cancel out with the linear transforms at the input and the output of the S-boxes, we obtain the construction shown in Figure 1.1. The result is a cipher which has a very simple representation in $GF(2^8)$, except for the eight B/B^{-1}-transforms in the middle.

Although we found some simple expressions relating inputs and outputs over three rounds without involving any B/B^{-1}-transforms[5] (which could be used for mounting an interpolation attack on a few rounds), we did not find a non-trivial way to exploit this property.

1.9 Key-Schedule Observation

The key-schedule of Aria consists of two phases:

1. a nonlinear expansion phase, in which a 128, 192, 256-bit key is expanded into four 128-bit words W_0, W_1, W_2, W_3;
2. a linear key-schedule phase in which the subkeys are derived via simple XORs and rotates from the words W_i.

Given the nonlinearity of the first phase and the irregular mixing of the second phase, we do not expect weak keys and related key attacks to be a serious threat. The key-schedule is efficient and provides good key-agility, but it means also that from knowing parts of the key-schedule other parts of the key-schedule may be recovered (note that the AES has a similar weakness). One of the properties that could be undesirable, is that for the 10-round version the subkey of the first and last rounds ek_1, ek_{11} are computed from the same words W_0, W_1. If an attack succeeds in recovering both subkeys (note that the two outer subkeys are usually the most vulnerable ones), it would in principle be possible to extract W_0 and W_1, which would allow to easily recover the master key.

We also stress the importance of the values of the constants CK_i. For example, if all constants were chosen to be 8-bit periodic (e.g., $CK_1 = \texttt{0x515151}\ldots\texttt{51}$), then an 8-bit periodic plaintext encrypted with an 8-bit periodic key would always result in an 8-bit

[5] For example, the (single) path connecting the first byte at the input of the first S-box layer and the third byte at the output of the third S-box layer does not leave the left hand side of the equivalent representation.

periodic ciphertext. This is a consequence of the fact that all components in Aria preserve 8-bit periodicity, both in the key schedule and in the cipher itself.

1.10 Increasing the Number of Different S-boxes

In this section we briefly discuss a variant of Aria which was proposed by the designers and uses four different S-boxes instead of two. The odd and even S-box layers in this variant could for example be constructed as follows:

$$\boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}$$

$$\boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}\ \boxed{S^{-1}}\ \boxed{R^{-1}}\ \boxed{S}\ \boxed{R}$$

Although we did not extensively study this variant, we believe that the proposed modification can only strengthen the cipher, provided that the S-boxes are well-chosen. We roughly estimate that the complexities of the attacks presented in Section 1.3.1 and 1.3.2 will increase with a factor 2^4 to 2^6. More importantly, it seems that this modification will prevent the dedicated attacks demonstrated in Section 1.3.3 and 1.4. It will also destroy the "splitting" property described in Section 1.8.

One possible drawback of using many different S-boxes is that it might increase the amount of memory required by the implementation. Note however that optimized 32-bit software implementations will not be affected.

Chapter 2

Efficiency Evaluation

In this chapter, the efficiency of Aria is estimated, both in software and hardware, and compared to Rijndael.

2.1 Software

Aria, Rijndael, and Camellia were implemented in C and compiled with gcc 3.3.1 using the flags `-O3 -fomit-frame-pointer`.[1] The three implementations were optimized for 32-bit processors using 8×32-bit table lookups. In the case of Aria, the special structure of the binary diffusion matrix was exploited as suggested in [14]. Table 2.1 lists the encryption times in cycles/byte measured on different machines (all 32-bit processors, except for the 64-bit PA-RISC) for the 128-bit key versions.

Table 2.1: Efficiency on 32/64-bit processors.

CPU		OS	Aria	Rijndael	Camellia
Pentium II (Klamath)	233MHz	Linux	31.2	23.4	33.9
Pentium III (Coppermine)	1.00GHz	Linux	31.1	23.3	33.4
Intel(R) Xeon(TM)	1.50GHz	Linux	41.5	21.0	82.2
AMD Athlon(TM) XP 2000+	1.66GHz	Linux	28.2	25.0	31.0
Intel(R) Pentium(R) 4	2.40GHz	Linux	40.8	30.5	83.9
PA-RISC 8500	400MHz	HP-UX	28.7	19.0	34.2

[1]The experiments showed that this combination of flags produced the fastest code for all three implementations. Note also that gcc is known to be slightly less efficient on the latest generation of Intel processors. For this reason, the code for the Xeon processor was also compiled using Intel's dedicated compiler. The resulting code is approximately 10% faster for Aria and Rijndael and 30% faster for Camellia.

It is interesting to note that, although both ARIA and RIJNDAEL were intentionally implemented in a very similar way, their behavior on some machines is rather different. The most striking are the Athlon, where both algorithms are approximately as fast, and the Xeon, where RIJNDAEL is almost twice as fast as ARIA. The most plausible explanation for this effect is a difference in cache size/speed (the 32-bit implementation of ARIA used in this experiment requires twice as much lookup tables as the RIJNDAEL implementation).

2.2 Hardware

This section compares hardware implementations of ARIA and RIJNDAEL. First the diffusion layers are compared, subsequently individual rounds, the complete encryption and decryption operations and then the key schedule. Finally an overview is presented of ASIC and FPGA implementations of RIJNDAEL.

2.2.1 ARIA Diffusion Layer

In Section 2.2.2 in [14] each XOR operation consists of eight 2-input XOR gates. There are 6 XOR operations for each y_i, $0 \leq i \leq 15$. Hence the number of 2-input XOR gates in the ARIA diffusion layer equals $8 \times 6 \times 16 = 768$. However, if the architecture proposed in Section 4.1 of [14] is used, the number of XOR gates can be reduced to 480.

The critical path of the diffusion layer of ARIA is 3 2-input XOR gates.

2.2.2 RIJNDAEL Diffusion Layer

RIJNDAEL needs 4 XOR operations for each byte of the state, b_i, $0 \leq i \leq 3$. In addition, at least one circuit for multiplying a four-byte column of the state with a fixed polynomial $c(x)$ modulo $x^4 + 1$ in MixColumn transformation needs to be implemented. The multiplication circuit consists of $8 \times 4 \times 4 = 128$ 2-input XOR gates; this circuit can be used four times in a loop (for the four columns of the state). Alternatively, we could unroll the circuit completely (as in the ARIA diffusion layer); in this case, the number of 2-input XOR gates will be $128 \times 4 = 512$.

Multiplying each byte of the state with '02' modulo $m(x) = x^8 + x^4 + x^3 + x + 1$ requires 3 2-input XOR gates. Again it is possible to implement one circuit for this multiplication and use it for all the bytes of the state. If we repeat instead the same circuit for all 16 bytes of the state then the number of 2-input XOR gates will be 48.

Hence the complete diffusion layer of RIJNDAEL consists of 560 2-input XOR gates.

The critical path of the RIJNDAEL MixColumn operation is 4 2-input XOR gates.

2.2.3 Aria One Round

Aria has different odd and even rounds. However, implementing a single round of Aria is sufficient, as this round can be iterated. Indeed, one can connect output bytes 1-8 of the odd rounds to input bytes 9-16 of the same implementation, and output bytes 9-16 of the even round to input bytes 1-8 of the odd round implementation.

Every round consists of three operations: *AddRoundKey*, *S-box* and *diffusion layer*. Therefore, if a round is fully pipelined, there can be at most three stages. In this case the critical path will be the same as the critical path of an *S-box*.

If the goal of the implementation is to make the clock frequency as high as possible, then the best choice for Aria is a loop or unrolled architecture with three pipeline stages in one round. A loop architecture requires 30 clock cycles for one encryption for the Aria version with a 128-bit key. A completely unrolled implementation requires a larger area, (the difference is created exclusively by the diffusion layers). The 128-bit version of Aria (encryption only) requires 4 800 2-input XOR gates less than Rijndael.

2.2.4 Rijndael One Round

All the rounds of Rijndael are the same. So implementing only one round is sufficient for a loop architecture without modified connections. Every round consists of three operations: *AddRoundKey*, *S-box* and *MixColumn*. Therefore, if a round is fully pipelined, there can be at most three stages. In this case the critical path will be the same as the critical path of an *S-box*. A loop architecture requires 30 clock cycles for one encryption for the Rijndael version with a 128-bit key, which is the same as for Aria.

2.2.5 Key Schedule

The key schedule is typically implemented using one of the following two methods: computing keys on-the-fly for every block of encrypted data or pre-computing them in advance and storing them. The computation of keys on-the-fly has an obvious advantage of changing keys fast with low or no delay.

In order to implement the key schedule on-the-fly the operation has to be finished in the same number of clock cycles as one round. Hence depending on the way one round has been implemented (pipelined/unrolled) the number of clock cycles available changes.

The smallest number of clock cycles that are used for one round will occur when an unrolled architecture is used. Then, the key schedule has to be finished in one clock cycle.

The number of pipeline stages can be at most three for both Aria and Rijndael as mentioned in Section 2.2.3 and Section 2.2.4. In this case the number of clock cycles to finish the key schedule can be three as well.

If the keys are pre-computed, it is possible to share S-boxes between the encryption and key schedule circuits.

2.2.5.1 Aria Key Schedule

In order to produce the keys on-the-fly and for the unrolled architecture the "initialization" and "round key generation" parts of the operation have to be implemented all as a combinational circuit. The critical path is $2F_o + F_e + 5XOR = 3\,(S + 3XOR) + 5XOR = 3S + 14XOR$.

When full pipelining is used in one round, the "initialization" part can be divided into three steps. Then the critical path is $F_o + 3XOR = (S + 3XOR) + 3XOR = S + 6XOR$.

If an architecture for pre-computing the keys that computes 32-bits of the key material in each clock is used, then in each clock cycle 32-bits of W_i, $i = 1, 2, 3$, can be computed. For the 128-bit case, computing one W_i takes 4 clock cycles. Because there are three W_is to compute, one round key can be computed in 12 clock cycles. The complete key schedule execution takes 132 clock cycles, which is 3 times slower than a similar architecture that is given in [44] for Rijndael.

2.2.5.2 Rijndael Key Schedule

For an on-the-fly architecture and the unrolled case the critical path for the Rijndael key schedule (with an 128-bit key and 128-bit data blocks) is reported as $2S + 8XOR$ in [17]. This result is $S + 6XOR$ shorter path than the critical path of Aria key schedule.

When full pipelining is used in one round, the architecture given in [17] can be divided into three steps. Then the critical path is $S+4XOR$. Again this result is shorter than the critical path of the Aria key schedule for the same choice.

The architecture for pre-computing the keys can be found in [44]. Their architecture computes 32-bits of the key material per clock cycle. Therefore the key schedule execution takes 44 clock cycles. By repeating the circuit given in [44] four times, it is possible to decrease the number of clock cycles to 11.

2.2.6 Encryption and Decryption

An advantage of Aria over Rijndael is that Aria can use the same hardware for encryption and decryption, as it is an involution. At first sight this will result in a savings of 50% for Aria (or even more, since for Rijndael decryption is slightly more complex than decryption).

However, it is shown in [44] that it is not necessary to implement two different circuits for encryption and decryption of Rijndael. A substantial part of the hardware can be shared between both. According to [44] the area can be decreased by 75% through sharing of circuits (compared with the total area of an encryption plus a decryption circuit); the loss of throughput of this approach is only 25%.

If the key schedule is done in advance then the area of a circuit for encryption of Aria will be smaller than Rijndael, because of the diffusion layer. The critical path for this architecture is the same for both block ciphers. But the number of clock cycles needed to compute the round keys is 3 times more in Aria.

If the key schedule is on-the-fly and the area used for the key schedule operation is the same for both ciphers, the critical path for the Aria circuit is longer than for the Rijndael circuit.

2.2.7 Rijndael Implementations

The properties of all the published ASIC implementations of Rijndael are listed in Table 2.2, while Table 2.3 and Table 2.4 describe the published FPGA implementations of Rijndael.

Table 2.2: ASIC Implementations

Ref.	Gate Count (10^3)	Clock Freq. (MHz)	Throughput (Gb/s)	Indicator (10^{-3} bits/gate)	Special Remarks
[16]	612.834	15.243	1.950	0.209	• Fully unrolled • Keys are generated and stored in subkey registers before encryption • No pipeline
[17]	173	100	1.82	0.105	• S-box is made by boolean functions • 1 round is done in each clock cycle • One encryption round is implemented • Only encryption is implemented
[18]	256	32	7.5	0.915	• S-box uses composite field arithmetic • Only encryption is implemented • 1 round is done in each clock cycle
[19]	5.4	131	0.3	0.42	
[20, 21]	173	125	1.6	0.07	• Non-pipelined encryption • Key schedule on-the-fly
[22]	40	1.66			
[23]	5.7	100	0.12	0.21	
[24]	14.9	114	4.79	2.819	• S-box uses multiplicative inverter in $GF(2^8)$ • Multiplicative inversion is in composite fields • Hardware is shared with Camellia
[25]	173	133	2.29	0.099	• Architecture is similar to [17] • It is tested

Table 2.3: FPGA Implementations

Ref.	CLB Count	BRAMs	Nbr of LUT	Clock Freq. (MHz)	Throughput (Gb/s)	Special Remarks
[26]	5673				0.353	• S-box is LUT in ROM
[27, 28, 29, 30]	2507				0.414	
[31]		8	780			• S-box is LUT
[32]	2 507			32	0.414	• Fully unrolled
[32]	2 057		8	99	1.265	• One round is pipelined
[32]	12 600	80		95	12.160	• Mixed with pipeline and unrolling
[33]	5 575				0.3	
[34]	1 257				0.964	• S-box is LUT
[35]	2222	100		54.35	7	• 1 round in each clock cycle • Fully pipelined • Hardware for 10 encryption rounds • S-box is in ROM (LUT)
[36]		20		2.5	7.6	• S-box is in ROM • Keys are generated and stored in 44-word registers before encryption • Fully pipelined • Hardware for 10 encryption rounds
[37]	4 325	1		75	0.739	• Keys are generated and stored in 11×128-bit registers before encryption
[38]	5 350	80	Nbr of	45	2.5	• Totally unrolled

Table 2.4: FPGA Implementations-Cont.

Ref.	CLB Count	BRAMs	Nbr of LUT	Clock Freq. (MHz)	Throughput (Gb/s)	Special Remarks
[41]	2 358			22	0.259	
[41]	17 314			28.5	3.65	• Using pipelining
[42]	2 580			38.8	0.451	• Key schedule is on-the-fly
[43]	702				0.755	
[44]	222	3		50	0.139	• S-box is in BlockRAM • MixColumns in 4-input LUTs • Both for encryption and decryption • Keys are generated and stored in a single Block RAM before encryption
[44]	222	3		60	0.166	
[45]						
[46]	2784	100	3516	92	0.004	• S-box is in RAM • unrolled architecture
[46]	542	10	877	119	1.45	• S-box is in RAM • loop architecture
[46]	1767		2524	167	2.085	• S-box is by using composite fields • loop architecture
[46]	2257		3846	169	2.008	• S-box is in LUT • loop architecture

Chapter 3

Conclusions

We have performed an independent security and efficiency evaluation of the ARIA block cipher. We have considered three versions, with 10, 12, and 14 rounds and with key lengths respectively of 128, 192 and 256 bits. The main conclusion of our analysis is that we have not found a practical or even a certificational weakness in any of the three versions of ARIA.

The performance of ARIA in software is slightly worse than that of RIJNDAEL; for hardware the performance of the two algorithms is comparable (there are small differences based on the specific requirements).

The ARIA block cipher offers a considerable security margin; nevertheless, for a new cipher that is designed for a long lifetime, a larger security margin (e.g., 2 additional rounds) may be desirable. We have some concerns about the specific properties of the diffusion matrix, but we have not been able to develop attacks that exploit these properties, except for reduced-round versions and with impractical complexities for 7 rounds already.

Our most important observations can be summarized as follows:

- Similarity to AES-RIJNDAEL. The current design is a 10-12-14 round 128-bit SPN using two types of S-boxes (based on inverse functions over $GF(2^8)$ as in RIJNDAEL followed by an affine mapping, which is different from the one used in RIJNDAEL). The diffusion mappings are based on binary self-inverse matrix multiplication. Due to the similarity with RIJNDAEL it is likely that ARIA will inherit several strong properties of RIJNDAEL. However, if any serious weakness would be discovered in RIJNDAEL, it seems highly likely that it would apply to ARIA as well. One such potential threat would be the possibility of an 'algebraic' attack of some kind. This means that ARIA could be a suitable alternative to RIJNDAEL, but may be less suited if one would only want to use it as a backup algorithm in the case that RIJNDAEL would be broken.

Table 3.1: Summary of attacks on reduced-round ARIA.

Attack	Key size	Rounds	Weak keys	Plaintext		Workload	Memory
Truncated:	128–256	7 of 10–14	-	2^{81}	CP	2^{81}	2^{80}
	128–256	7 of 10–14	-	2^{100}	CP	2^{100}	2^{51}
Tr./Multiset[a]:	128–256	7 of 10–14	-	2^{124}	ACP	2^{124}	2^{8}
	128–256	7 of 10–14	-	2^{104}	CP	2^{112}	2^{8}
Linear[a]:	128	7 of 10	2^{96}	2^{77}	KP	2^{88}	2^{64}
	192	10 of 12	2^{136}	2^{119}	KP	2^{119}	2^{64}
	256	10 of 14	2^{200}	2^{119}	KP	2^{119}	2^{64}

[a]Dedicated attacks exploiting the interaction between the diffusion matrix and the structure of the S-box layer.

- ARIA has very specific features (involutional diffusion, special structure of the matrix), which may open avenues to new attack methods. As an example, the binary matrix used by the cipher has very special properties. The attacker could exploit linear subspaces corresponding to eigenvectors of the type: $(1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)$. This property, together with the fact that the corresponding active S-boxes are identical, allows for dedicated attack approaches described in this report, that are effective against reduced-round versions of ARIA. However, in our analysis, we have not been able to find a way to extend this to the full versions of ARIA.

- Truncated differential attacks: due to the previous observation we found a truncated differential attack on a variant of ARIA reduced to 7 rounds. We recommend to perform a deeper analysis of truncated differential attacks, since we were not yet able to exploit all the structure identified during our analysis. Nevertheless, with the current state of knowledge, it seems unlikely that further analysis will uncover a practical attack on the full ARIA.

Bibliography

[1] E. Barkan, E. Biham, "In how many ways can you write Rijndael," in *Advances in Cryptology – ASIACRYPT 2002* (Y. Zheng, ed.), vol. 2501 of *Lecture Notes in Computer Science*, pp. 160–175, Springer-Verlag, 2002.

[2] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[3] A. Biryukov and D. Wagner, "Slide attacks," in *Fast Software Encryption, FSE 1999* (L. R. Knudsen, ed.), vol. 1636 of *Lecture Notes in Computer Science*, pp. 245–259, Springer-Verlag, 1999.

[4] A. Biryukov and A. Shamir, "Structural cryptanalysis of SASAS;" in *Advances in Cryptology – EUROCRYPT 2001* (B. Pfitzmann, ed.), vol. 2045 of *Lecture Notes in Computer Science*, pp. 394–405, Springer-Verlag, 2001.

[5] A. Biryukov, C. De Cannière, A. Braeken, and B. Preneel, "A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms," in *Advances in Cryptology – EUROCRYPT 2003* (E. Biham, ed.), vol. 2656 of *Lecture Notes in Computer Science*, pp. 33–50, Springer-Verlag, 2003.

[6] A. Biryukov, "Analysis of involutional ciphers: Khazad and Anubis," in *Fast Software Encryption, FSE 2003* (T. Johansson, ed.), vol. 2887 of *Lecture Notes in Computer Science*, pp. 45–53, Springer-Verlag, 2003.

[7] N. T. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," in *Advances in Cryptology – ASIACRYPT 2002* (Y. Zheng, ed.), vol. 2501 of *Lecture Notes in Computer Science*, pp. 267–287, Springer-Verlag, 2002. Earlier version available from `http://www.iacr.org`.

[8] J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer-Verlag, 2002.

[9] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of Rijndael," in *Fast Software Encryption, FSE 2000* (B. Schneier, ed.), vol. 1978 of *Lecture Notes in Computer Science*, pp. 213–230, Springer-Verlag, 2001.

[10] H. Gilbert and M. Minier, “A collision attack on seven rounds of Rijndael,” in *Proceedings of the Third AES Candidate Conference*, pp. 230–241.

[11] L. R. Knudsen, “Truncated Differentials of SAFER,” in *Fast Software Encryption, FSE'96* (D. Gollmann, ed.), vol. 1039 of *Lecture Notes in Computer Science*, pp. 15–26, Springer-Verlag, 1996.

[12] M. Matsui, “Linear cryptanalysis method for DES cipher,” in *Advances in Cryptology – EUROCRYPT'93* (T. Helleseth, ed.), vol. 765 of *Lecture Notes in Computer Science*, pp. 386–397, Springer-Verlag, 1993.

[13] S. Murphy and M. J. B. Robshaw, “Essential algebraic structure within the AES,” in *Advances in Cryptology – CRYPTO 2002* (M. Yung, ed.), vol. 2442 of *Lecture Notes in Computer Science*, pp. 17–38, Springer-Verlag, 2002.

[14] National Security Research Institute, *Specification of ARIA*, Version 0.8, August, 2003.

[15] D. Wagner, “The boomerang attack,” in *Fast Software Encryption, FSE'99* (L. R. Knudsen, ed.), vol. 1636 of *Lecture Notes in Computer Science*, pp. 156–170, Springer-Verlag, 1999.

[16] T. Ichikawa, T. Kasuya, and M. Matsui. Hardware evaluation of the AES finalists. In *Proceedings of the third AES Candidate Conference*, New York, USA, April 13-14 2000.

[17] H. Kuo and I. Verbauwhede. Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES Rijndael algorithm. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, page 5164, Paris, France, May 13-16 2001. Springer-Verlag.

[18] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi. Efficient Rijndael encryption implementation with composite field arithmetic. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, page 171 184, Paris, France, May 14-16 2001. Springer-Verlag.

[19] A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A compact Rijndael hardware architecture with S-box optimization. In C. Boyd, editor, *Proceedings of Advances in Cryptology - ASIACRYPT: 7th International Conference on the Theory and Application of Cryptology and Information Security*, number 2248 in Lecture Notes in Computer Science, pages 239–254, Gold Coast, Australia, December 2001. Springer-Verlag.

[20] H. Kuo, I. Verbauwhede, and P. Schaumont. A 2.29 Gbits/sec, 56 mW non-pipelined Rijndael AES encryption IC in a 1.8V, 0.18 μm CMOS technology. In *IEEE Custom Integrated Circuits Conference (CICC)*, pages 147–150, May 2002.

[21] P. R. Schaumont, H. Kuo, and I. M. Verbauwhede. Unlocking the design secrets of a 2.29 Gb/s Rijndael processor. In *Proceeding of The Design Automation Conference (DAC)*, New Orleans, LO, USA, June 10-14 2002.

[22] Y.K. Lee and Y.S. Park. Implementation of Rijndael block cipher algorithm. In *Proceedings of the International Conference on Circuits/Systems Computers and Communications (ITC-CSCC)*, Phuket, Thailand, July 16-19 2002.

[23] J. Wolkerstorfer, E. Oswald, and M. Lamberger. An ASIC implementation of the AES S-boxes. In B. Preneel, editor, *Proceedings of the RSA Conference - Topics in Cryptography (CT-RSA)*, number 2271 in Lecture Notes in Computer Science, pages 67–78, San Jose, USA, February 18-22 2002. Springer-Verlag.

[24] A. Satoh and S. Morioka. Unified hardware architecture for 128-bit block ciphers AES and Camellia. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, page 304318, Cologne, Germany, September 7-10 2003. Springer-Verlag.

[25] I. Verbauwhede, P. Schaumont, and H. Kuo. Design and performance testing of a 2.29-Gb/s Rijndael processor. *IEEE Journal of Solid-State Circuits*, 38(3):569–572, March 2003.

[26] A. Dandalis, V. K. Prasanna, and J. D. P. Rolimy. A comparative study of performance of AES final candidates using FPGAs. In *Proceedings of the third Advanced Encryption Standard (AES) Candidate Conference*, New York, USA, April 13-14 2000.

[27] K. Gaj and P. Chodowiec. Hardware performance of the AES finalists - survey and analysis of results. Technical report, George Mason University, September 2000.

[28] K. Gaj and P. Chodowiec. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. In *Proceedings of the third Advanced Encryption Standard (AES) Candidate Conference*, New York, USA, April 13-14 2000.

[29] K. Gaj and P. Chodowiec. Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays. In D. Naccache, editor, *Proceedings of RSA Security Conference: Topics in Cryptology - CT-RSA*, number 2020 in Lecture Notes in Computer Science, San Francisco, CA, USA, April 8-12 2001. Springer-Verlag.

[30] P. Chodowiec. Comparison of the hardware performance of the AES candidates using reconfigurable hardware. Master of science, George Mason University, Fairfax, VA, 2002.

[31] N. Weaver and J. Wawrzynek. A comparison of the AES candidates amenability to FPGA implementation. In *Proceedings of the third Advanced Encryption Standard (AES) Candidate Conference*, New York, USA, April 13-14 2000.

[32] P. Chodowiec, P. Khuon, and K. Gaj. Fast implementations of secret-key block ciphers using mixed inner and outer-round pipelining. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 94–102, Monterey, CA, USA, February 11-13 2001.

[33] A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. *IEEE Transactions on VLSI*, 9(4):545–559, August 2001.

[34] V. Fischer and M. Drutarovský. Two methods of Rijndael implementation in reconfigurable hardware. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, page 7792, Paris, France, May 13-16 2001. Springer-Verlag.

[35] M. McLoone and J.V McCanny. High performance single-chip FPGA Rijndael algorithm implementations. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in Lecture Notes in Computer Science, page 6576, Paris, France, May 13-16 2001. Springer-Verlag.

[36] M. Alam, W. Badawy, and G. Jullien. A novel pipelined threads architecture for AES encryption algorithm. In M. Schulte, S. Bhattacharyya, N. Burgess, and R. Schreiber, editors, *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, pages 296–302, San Jose, CA, USA, July 17-19 2002. IEEE Computer Society Press.

[37] C. Chitu, D. Chien, C. Chien, I. Verbauwhede, and F. Chang. A hardware implementation in FPGA of the Rijndael algorithm. In *Proceedings of the 45th Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 507–510, August 2002.

[38] T. Kerins, E. Popovici, A. Daly, and W. Marnane. Hardware encryption engines for e-commerce. In *Proceedings of ISSC*, Cork, June 25-26 2002.

[39] O. Kwon, H. Seike, H. Kajisaki, and T. Kurokawa. Implementation of AES and Triple-DES cryptography using a PCI-based FPGA board. In *Proceedings of the International Technical Conference On Circuits/Systems, Computers and Communications (ITC-CSCC)*, 2002.

[40] R. Karri and Y. Kim. Field programmable gate array implementation of advanced encryption standard Rijndael.

[41] N. Sklavos and O. Koufopavlou. Architectures and VLSI implementations of the AES-proposal Rijndael. *IEEE Transactions on Computers*, 51(12):1454–1459, December 2002.

[42] J. H. Shim, D. W. Kim, Y. K. Kang, T. W. Kwon, and J. R. Choi. A Rijndael cryptoprocessor using shared on-the-fly key scheduler. In *Proceedings of the third IEEE Asia-Pacific Conference on ASICs*, Taipei, Taiwan, August 6-8 2002.

[43] D. K. Y. Tong, P. S. Lo, K. H. Lee, and P. H. W. Leong. A system level implementation of Rijndael on a memory-slot based FPGA card. 2002.

[44] P. Chodowiec and K. Gaj. Very compact FPGA implementation of the AES algorithm. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, page 319333, Cologne, Germany, September 7-10 2003. Springer-Verlag.

[45] F.-X. Standaert, G. Rouvroy, J.-J Quisquater, and J.-D. Legat. A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES Rijndael. In *Proceedings of the Field Programmable Logic Array Conference (FPGA) 2003*, Monterey, CA, USA, February 23-25 2003.

[46] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2779 in Lecture Notes in Computer Science, pages 334–350, Cologne, Germany, September 7-10 2003. Springer-Verlag.

Appendix A

Statistical tests

A.1 Overview

The ARIA block cipher was submitted to a series of statistical tests. The test results did not indicate a deviation from random behaviour.

Here is an overview of the tests that were performed:

- Block cipher tests:
  - `dependence`
  - `findlinfacts`
- Stream cipher tests (in CTR and OFB mode):
  - `collision`
  - `correlation`
  - `coupon`
  - `fastspectral`
  - `frequency`
  - `gap`
  - `lincompl`
  - `maurer`
  - `mtuple`
  - `nonlincompl`
  - `poker`

– `rank`
– `run`
– `zivlempel`

A.2 Block cipher tests on ARIA

A.2.1 Dependence Test

The dependence test evaluates the dependence matrix and the distance matrix of the cipher. Furthermore, the degree of completeness, the degree of avalanche effect and the degree of strict avalanche criterion of the cipher are computed. A cryptographic function is complete if each output bit depends on each input bit. For a function to exhibit the avalanche effect, an average of one half of the output bits should change whenever a single input bit is complemented. A function satisfies the strict avalanche criterion if each output bit changes with a probability of one half whenever a single input bit is complemented.

```
DEPENDENCE TEST for
ARIA Block Cipher


Number of inputs:     10000

Average number of output bits changed: 63.992001

Degree of completeness              : 1.000000
Degree of avalanche effect          : 0.999203
Degree of strict avalanche criterion : 0.991993



ANALYSIS OF THE DISTANCE MATRIX

Average fractions of inputs yielding distance j if one bit is complemented,
and the expected fractions for a random function

 j           0        1        2        3        4        5        6        7
exp.  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
av.   0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

 j           8        9       10       11       12       13       14       15
exp.  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
av.   0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

 j          16       17       18       19       20       21       22       23
exp.  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
av.   0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

 j          24       25       26       27       28       29       30       31
exp.  0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
av.   0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

 j          32       33       34       35       36       37       38       39
exp.  0.000000 0.000000 0.000000 0.000000 0.000000 0.000001 0.000001 0.000003
av.   0.000000 0.000000 0.000000 0.000000 0.000000 0.000002 0.000002 0.000005

 j          40       41       42       43       44       45       46       47
exp.  0.000007 0.000016 0.000033 0.000067 0.000129 0.000240 0.000433 0.000756
av.   0.000013 0.000020 0.000028 0.000064 0.000130 0.000230 0.000436 0.000816

 j          48       49       50       51       52       53       54       55
exp.  0.001276 0.002082 0.003290 0.005032 0.007452 0.010685 0.014841 0.019967
av.   0.001255 0.002038 0.003341 0.004912 0.007415 0.010720 0.014713 0.020001
```

j	56	57	58	59	60	61	62	63
exp.	0.026029	0.032879	0.040248	0.047752	0.054915	0.061216	0.066153	0.069303
av.	0.026266	0.032885	0.040310	0.048091	0.054747	0.061049	0.066731	0.069120

j	64	65	66	67	68	69	70	71
exp.	0.070386	0.069303	0.066153	0.061216	0.054915	0.047752	0.040248	0.032879
av.	0.070433	0.069708	0.066130	0.060966	0.054583	0.047830	0.040223	0.032923

j	72	73	74	75	76	77	78	79
exp.	0.026029	0.019967	0.014841	0.010685	0.007452	0.005032	0.003290	0.002082
av.	0.025916	0.019716	0.014861	0.010630	0.007364	0.005139	0.003327	0.001995

j	80	81	82	83	84	85	86	87
exp.	0.001276	0.000756	0.000433	0.000240	0.000129	0.000067	0.000033	0.000016
av.	0.001248	0.000766	0.000427	0.000240	0.000112	0.000055	0.000036	0.000017

j	88	89	90	91	92	93	94	95
exp.	0.000007	0.000003	0.000001	0.000001	0.000000	0.000000	0.000000	0.000000
av.	0.000012	0.000003	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

j	96	97	98	99	100	101	102	103
exp.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
av.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

j	104	105	106	107	108	109	110	111
exp.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
av.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

j	112	113	114	115	116	117	118	119
exp.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
av.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

j	120	121	122	123	124	125	126	127
exp.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
av.	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

j	128
exp.	0.000000
av.	0.000000

Application of the Chi-square test to the rows of the distance matrix
(% levels of significance)

row	%	row	%	row	%	row	%	row	%
1	72.4	2	98.2	3	87.4	4	75.3	5	23.7
6	14.1	7	66.5	8	11.7	9	77.1	10	28.5
11	23.3	12	10.3	13	15.3	14	98.8	15	88.5
16	42.3	17	3.4	18	93.6	19	25.9	20	4.2
21	70.6	22	12.3	23	19.5	24	20.2	25	91.2
26	53.4	27	84.8	28	10.1	29	51.1	30	41.3
31	76.3	32	0.1	33	81.4	34	72.4	35	75.3
36	85.7	37	32.2	38	19.7	39	50.6	40	90.5
41	53.4	42	76.7	43	3.2	44	41.1	45	36.3
46	23.6	47	64.5	48	80.8	49	75.2	50	80.0
51	1.3	52	89.7	53	47.9	54	90.8	55	61.1
56	27.8	57	56.3	58	74.4	59	89.3	60	64.6
61	33.0	62	17.2	63	50.3	64	87.8	65	15.5
66	33.3	67	91.5	68	89.6	69	13.0	70	82.4
71	31.1	72	57.6	73	1.0	74	79.3	75	53.0
76	15.2	77	6.2	78	38.3	79	5.8	80	58.7
81	84.2	82	53.7	83	23.5	84	20.0	85	0.8
86	4.2	87	24.8	88	51.7	89	48.0	90	80.8
91	42.6	92	1.7	93	24.7	94	50.8	95	18.2

96	99.7	97	33.3	98	26.6	99	20.1	100	83.4
101	19.1	102	33.9	103	41.6	104	84.8	105	76.6
106	48.7	107	34.8	108	42.9	109	6.0	110	30.2
111	42.4	112	52.0	113	41.2	114	85.1	115	88.4
116	72.4	117	0.4	118	72.8	119	60.3	120	18.9
121	15.2	122	80.7	123	42.2	124	72.7	125	64.9
126	18.1	127	50.8	128	46.0				

ANALYSIS OF THE DEPENDENCE MATRIX

Row average of the dependence matrix

i		i		i		i	
1	0.499600	2	0.499853	3	0.499417	4	0.500339
5	0.500495	6	0.499591	7	0.500015	8	0.499119
9	0.499860	10	0.500397	11	0.499624	12	0.499425
13	0.499182	14	0.500255	15	0.500950	16	0.500385
17	0.500039	18	0.500380	19	0.499792	20	0.500355
21	0.499814	22	0.499298	23	0.499277	24	0.499466
25	0.500087	26	0.498973	27	0.500232	28	0.500692
29	0.500384	30	0.500005	31	0.499671	32	0.501041
33	0.499230	34	0.499784	35	0.499894	36	0.499709
37	0.499828	38	0.500027	39	0.499268	40	0.499491
41	0.500310	42	0.500305	43	0.500387	44	0.500791
45	0.499764	46	0.499959	47	0.499768	48	0.500176
49	0.499895	50	0.499977	51	0.499970	52	0.499901
53	0.500725	54	0.499869	55	0.499941	56	0.500775
57	0.498984	58	0.499819	59	0.499752	60	0.499948
61	0.500042	62	0.500727	63	0.499309	64	0.500712
65	0.499906	66	0.500535	67	0.499570	68	0.499928
69	0.499329	70	0.499352	71	0.500558	72	0.499899
73	0.499527	74	0.499383	75	0.499591	76	0.499331
77	0.499471	78	0.500041	79	0.500836	80	0.500261
81	0.500773	82	0.499359	83	0.500568	84	0.499727
85	0.499879	86	0.500132	87	0.499431	88	0.500915
89	0.499752	90	0.499938	91	0.499770	92	0.500292
93	0.500114	94	0.499525	95	0.500278	96	0.500183
97	0.499566	98	0.499508	99	0.499018	100	0.500310
101	0.500222	102	0.499934	103	0.500438	104	0.499673
105	0.499582	106	0.499702	107	0.499291	108	0.500427
109	0.499975	110	0.500016	111	0.499905	112	0.500518
113	0.500555	114	0.500211	115	0.499907	116	0.498889
117	0.498472	118	0.500251	119	0.499695	120	0.500164
121	0.499771	122	0.500520	123	0.500251	124	0.499914
125	0.500929	126	0.500190	127	0.499280	128	0.499713

Column average of the dep. matrix

i		i		i		i	
1	0.499224	2	0.500333	3	0.500255	4	0.499726
5	0.500292	6	0.499481	7	0.500495	8	0.500123
9	0.500256	10	0.499598	11	0.500160	12	0.500284
13	0.500209	14	0.499995	15	0.500297	16	0.500227
17	0.499446	18	0.500373	19	0.500345	20	0.500088
21	0.499009	22	0.499940	23	0.500608	24	0.500466
25	0.499510	26	0.500134	27	0.499271	28	0.499678
29	0.500292	30	0.499840	31	0.499370	32	0.499947
33	0.500195	34	0.499425	35	0.499127	36	0.500339
37	0.499769	38	0.500187	39	0.499159	40	0.500240
41	0.500633	42	0.498991	43	0.499258	44	0.500041
45	0.499979	46	0.500672	47	0.499760	48	0.499802
49	0.500410	50	0.499664	51	0.500800	52	0.499281

53	0.500341	54	0.500686	55	0.499839	56	0.500366
57	0.500955	58	0.500104	59	0.498612	60	0.499921
61	0.499187	62	0.499575	63	0.499768	64	0.499784
65	0.499644	66	0.500432	67	0.500378	68	0.500041
69	0.500248	70	0.500587	71	0.499937	72	0.499890
73	0.499796	74	0.499916	75	0.499926	76	0.499396
77	0.499267	78	0.500051	79	0.499351	80	0.500723
81	0.500493	82	0.499631	83	0.499247	84	0.500334
85	0.500232	86	0.499370	87	0.499890	88	0.499995
89	0.499823	90	0.499836	91	0.499830	92	0.500062
93	0.500237	94	0.499944	95	0.500440	96	0.500541
97	0.499534	98	0.500345	99	0.500247	100	0.499857
101	0.500159	102	0.499520	103	0.499467	104	0.500513
105	0.501022	106	0.500083	107	0.500092	108	0.500130
109	0.500106	110	0.500015	111	0.499866	112	0.499547
113	0.500066	114	0.499176	115	0.499968	116	0.500827
117	0.499477	118	0.499957	119	0.499119	120	0.500211
121	0.499382	122	0.499354	123	0.499251	124	0.499573
125	0.499958	126	0.500313	127	0.499554	128	0.499691

A.2.2 Linear Factors Test

The linear factors test is used to find out whether there are any linear combinations of output bits which, for all keys and plaintexts, are independent of one or more key or plaintext bits. Such a linear combination is called a linear factor. It is practically impossible to check a potential linear factor for all keys and plaintexts. Therefore, we only consider for a sufficiently large number of pairs of random keys and random plaintexts.

For Aria, no linear factors were found.

A.3 Aria in OFB Mode

A.3.1 Collision Test

The collision test splits up the bit sequence into subsequent, disjoint m-tuples of bits. The test evaluates statistically how often such m-tuples occur more than once.

```
Collision Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 24
# of blocks: 416666 blocksize: 24 collisions: 5258
```

A.3.2 Correlation Test

The correlation test determines in how many places the original sequence and the sequence shifted by n bits have the same value. This is done for all shifts n up to the length of the original sequence. To support the interpretation of the results, for each shift the probability for a sequence of random, independent, and uniformly distributed bits to have this number or less coincidences with its shifted copy is determined. Only values where these probabilities are close to 0 or 1 are printed. The print level is the maximal deviation from 0 or 1 for these probabilities in order to be printed.

```
Correlation Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 1000000
Printlevel: 0.000010
shift:0 equal: 1000000 probability: 1.0000000   0e+00
shift:78002 equal: 497824 probability: 0.0000068   7e-06
shift:166724 equal: 497816 probability: 0.0000063   6e-06
shift:260279 equal: 502208 probability: 0.9999950   5e-06
shift:288114 equal: 502224 probability: 0.9999957   4e-06
shift:293064 equal: 502383 probability: 0.9999991   9e-07
shift:298759 equal: 502311 probability: 0.9999981   2e-06
shift:317583 equal: 497759 probability: 0.0000037   4e-06
shift:333656 equal: 502241 probability: 0.9999963   4e-06
shift:372626 equal: 502194 probability: 0.9999943   6e-06
shift:456563 equal: 497702 probability: 0.0000022   2e-06
shift:510811 equal: 497740 probability: 0.0000031   3e-06
shift:555239 equal: 497748 probability: 0.0000034   3e-06
shift:650228 equal: 497623 probability: 0.0000010   1e-06
shift:653155 equal: 502202 probability: 0.9999947   5e-06
shift:705989 equal: 502452 probability: 0.9999995   5e-07
shift:714809 equal: 502137 probability: 0.9999904   1e-05
shift:736026 equal: 502140 probability: 0.9999907   9e-06
shift:750269 equal: 502137 probability: 0.9999904   1e-05
shift:753153 equal: 502283 probability: 0.9999975   2e-06
shift:762585 equal: 502172 probability: 0.9999930   7e-06
shift:785731 equal: 502314 probability: 0.9999982   2e-06
shift:837853 equal: 502133 probability: 0.9999901   1e-05
shift:930146 equal: 502136 probability: 0.9999904   1e-05
```

```
shift:936972 equal: 497783 probability: 0.0000047   5e-06
shift:938343 equal: 497714 probability: 0.0000024   2e-06
```

A.3.3 Coupon Collector's Test

The coupon collector's test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the word length of the test. In the test, the number of subsequent m-tuples it takes until all possible 2^m m-tuples have appeared, is evaluated statistically. The coupon collector's test is also applied to cyclic shifts of the original sequence.

```
COUPON COLLECTOR'S TEST for
ARIA Block Cipher in OFB mode
Sequencelength = 10000000 bits        Wordlength = 8 bits
Ideal distribution:       mean    variance
                     1567.8323   105979.0660
Real distribution:
The results are for cyclic shifts
Shift:               0        1        2        3        4        5        6        7
             -----------------------------------------------------------------------
 mean:        1564.018 1557.151 1563.635 1556.137 1573.928 1570.389 1553.832 1563.620
 error:          -0.24%   -0.68%   -0.27%   -0.75%    0.39%    0.16%   -0.89%   -0.27%
 variance:       96552   100397   101294    97483    99609   104829   100415    91931
 error:          -8.89%   -5.27%   -4.42%   -8.02%   -6.01%   -1.09%   -5.25%  -13.26%

    p=           52.73%   23.15%    8.14%   10.45%   23.91%   51.24%   64.06%   56.51%

p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

A.3.4 Fast Spectral Test

The fast spectral test applies the fast Walsh transform to the given sequence. It uses two values derived from the transform to assess the randomness of the sequence.

```
Fast Spectral Test for
ARIA Block Cipher in OFB mode


The results are:

The statistic D(4) = 1.109035E-01; percentage level of significance:  54.4%
The statistic D(6) = 3.319439E-01; percentage level of significance:  63.0%
```

A.3.5 Frequency Test

The frequency test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the blocksize of the test. The frequencies of the occurrences of these m-tuples are counted and evaluated statistically. This test is performed for various values of m.

```
Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 1
sequencelength= 10000000 blocksize= 1
block: 0  count: 4999131
block: 1  count: 5000869
chisquare = 0.302064  nu= 1
Percentage Level of Acceptance:  58.26


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 2
sequencelength= 10000000 blocksize= 2
block: 0  count: 1250113
block: 1  count: 1248498
block: 2  count: 1249284
block: 3  count: 1252105
chisquare = 5.769963  nu= 3
Percentage Level of Acceptance:  12.34


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 3
sequencelength= 10000000 blocksize= 3
block: 0  count: 416471
block: 1  count: 416642
block: 2  count: 416992
block: 3  count: 416992
block: 4  count: 416770
block: 5  count: 417074
block: 6  count: 415999
block: 7  count: 416393
chisquare = 2.274835  nu= 7
Percentage Level of Acceptance:  94.31


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 4
sequencelength= 10000000 blocksize= 4
block: 0  count: 156215
block: 1  count: 155865
block: 2  count: 155908
block: 3  count: 155533
block: 4  count: 156951
block: 5  count: 156513
block: 6  count: 156182
block: 7  count: 155926
block: 8  count: 156307
block: 9  count: 156814
block: 10  count: 155977
block: 11  count: 156337
block: 12  count: 156042
block: 13  count: 156415
block: 14  count: 156393
```

```
block: 15  count: 156622
chisquare = 13.334003  nu= 15
Percentage Level of Acceptance:  57.65


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 5
sequencelength= 10000000 blocksize= 5
block: 0  count: 62336
block: 1  count: 62375
block: 2  count: 62264
block: 3  count: 62464
block: 4  count: 62761
block: 5  count: 62364
block: 6  count: 62678
block: 7  count: 62527
block: 8  count: 62710
block: 9  count: 62411
block: 10  count: 62534
block: 11  count: 62123
block: 12  count: 62672
block: 13  count: 62674
block: 14  count: 62129
block: 15  count: 62742
block: 16  count: 62110
block: 17  count: 62413
block: 18  count: 62768
block: 19  count: 62378
block: 20  count: 62960
block: 21  count: 62627
block: 22  count: 62784
block: 23  count: 62238
block: 24  count: 62485
block: 25  count: 62499
block: 26  count: 62443
block: 27  count: 63158
block: 28  count: 62646
block: 29  count: 62044
block: 30  count: 62224
block: 31  count: 62459
chisquare = 32.589888  nu= 31
Percentage Level of Acceptance:  38.86


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 6
sequencelength= 10000000 blocksize= 6
chisquare = 69.839738  nu= 63
Percentage Level of Acceptance:  25.87


Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 7
sequencelength= 10000000 blocksize= 7
chisquare = 158.224356  nu= 127
Percentage Level of Acceptance:  3.15
```

```
Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 8
sequencelength= 10000000 blocksize= 8
chisquare = 272.999424  nu= 255
Percentage Level of Acceptance:  20.94
```

```
Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 12
sequencelength= 10000000 blocksize= 12
chisquare = 4093.920571  nu= 4095
Percentage Level of Acceptance:  50.18
```

```
Frequency Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 16
sequencelength= 10000000 blocksize= 16
chisquare = 65566.778061  nu= 65535
Percentage Level of Acceptance:  46.43
```

A.3.6 Gap Test

The gap test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the word length of the test. The m-tuples are interpreted as binary representations of numbers, and the lengths of gaps, where the numbers are not within a numerical range given as a parameter of the test, are registered and evaluated statistically. The gap test is also applied to cyclic shifts of the original sequence.

```
GAP TEST for
ARIA Block Cipher in OFB mode
Sequencelength = 10000000 bits        Wordlength = 10 bits
Length of gaps between occurences in the range 256 - 768
Ideal distribution:       mean    variance
                         1.000       2.000
Real distribution:
Shift:            0       1       2       3       4       5       6       7       8       9
            --------------------------------------------------------------------------------
 mean:         1.002   1.001   1.001   1.003   1.000   1.001   1.001   1.004   1.001   0.998
 error:         0.25%   0.11%   0.11%   0.31%  -0.02%   0.11%   0.07%   0.38%   0.11%  -0.22%
 variance:     2.001   2.006   2.012   2.006   2.004   1.997   2.007   2.000   1.995   1.988
 error:         0.06%   0.29%   0.61%   0.31%   0.20%  -0.17%   0.33%  -0.02%  -0.25%  -0.59%

    p=         15.36%  49.85%  20.86%  38.82%  22.11%   3.33%  84.54%  12.38%  20.80%  80.39%

p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

		499383	499722	499718	499224	500050	499727	499834	499054	499713	500559
0	5.00e-01	249377	249812	250084	249169	250496	249198	249736	248681	249685	250488
1	2.50e-01	124363	125008	124604	124905	124490	125415	125387	124779	124729	125057
2	1.25e-01	63111	62337	62374	62463	62476	62831	62271	62855	62383	62604
3	6.25e-02	31210	31217	31341	31221	31324	31159	31079	31467	31803	31279
4	3.12e-02	15766	15481	15536	15699	15630	15553	15595	15729	15570	15668
5	1.56e-02	7834	7961	7883	7956	7666	7704	7872	7879	7745	7757
6	7.81e-03	3777	4026	3947	3856	3934	3946	3969	3838	3954	3899
7	3.91e-03	1995	1919	2002	1962	2032	1971	1970	1872	1964	1877
8	1.95e-03	974	970	950	989	1035	1000	972	981	966	973
9	9.77e-04	487	516	478	529	512	504	474	479	449	476
10	4.88e-04	261	239	269	266	237	193	256	245	228	238
11	2.44e-04	102	110	110	106	112	116	112	117	120	125
12	1.22e-04	64	70	64	50	51	68	74	57	52	53
13	6.10e-05	32	24	36	26	32	39	35	45	35	35
14	3.05e-05	14	18	18	16	13	16	15	14	15	21
15	1.53e-05	7	8	6	6	3	3	9	8	5	5
16	7.63e-06	6	4	9	2	6	8	5	5	2	3
17	3.81e-06	1	1	5	1	1	2	0	2	4	0
18	1.91e-06	2	1	1	2	0	1	3	1	3	1
19	9.54e-07	0	0	0	0	0	0	0	0	1	0
20	9.54e-07	0	0	1	0	0	0	0	0	0	0

A.3.7 Linear Complexity Test

The linear complexity test uses the Berlekamp Massey algorithm to determine the length of the shortest linear feedback shift register which can produce the given bit sequence. For the linear complexity profile, this is done for the first 1, 2, 3, . . . bits of the sequence. Some properties of this profile are evaluated.

```
Linear Complexity Test for
ARIA Block Cipher in OFB mode


-------------------- Final results ------------------

N= 100000      L= 50000      X= 4

N    is the number of input bits.

L    is the linear complexity.

X-1  is the number of bits which has been treated since
     the last change of linear complexity.

------------------------- End ----------------------
The linear complexity profile:


Jumps in the linear complexity profile:



ssl =  99999.000    ssqsl = 500217.000
msl =      4.000    varsl =      4.009
```

```
ssh =  50000.000    ssqsh = 149770.000
msh =      2.000    varsh =      1.991



  ssl   is the sum of the sl's
  ssqsl is the sum of the squares of the sl's
  msl   is the mean of the sl's
  varsl is the variance of the sl's
  The number of jumps used in the calculation of msl and varsl is:  25000
  The first sl is not counted because it is 0

  ssh   is the sum of the sh's
  ssqsh is the sum of the squares of the sh's
  msh   is the mean of the sh's
  varsh is the variance of the sh's
  The number of jumps used in the calculation of msh and varsh is:  25000
maximal step-height 15.000000


  sl  is the steplength
  sh  is the stepheigth
  nj  is the number of jumps
```

A.3.8 Maurer Test

The universal Maurer test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the blocksize of the test. The test evaluates statistically how many m-tuples later an m-tuple re-appears in the sequence. The test result of the Maurer test is closely related to the entropy of the bit sequence.

```
Maurer Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 1000000
Block size: 8
Initial Blocks= 10000
blocks tested (including initial blocks): 125000
Maurer Test Value: 7.186465
```

A.3.9 Overlapping m-tuple Test

The overlapping m-tuple test splits up the bit sequence into m-tuples of words. Each word contains a fixed number of bits. In the overlapping m-tuple test, the m-tuples are not disjoint; to take the next m-tuple, an m-word-window on the original sequence is shifted by one word. So the next m-tuple consists of $m - 1$ shifted words of the previous m-tuple and one new word. Since subsequent m-tuples are not independent, the statistical evaluation is more involved than in the case of the frequency test, but this is handled by the test program. This test is also applied to cyclic shifts of the original sequence.

```
OVERLAPPING M-TUPLE TEST for
ARIA Block Cipher in OFB mode
Sequencelength = 10000000 bits        Wordlength = 5 bits
m = 2

Shift     p
--------------
  0     76.91%
  1     51.17%
  2     10.98%
  3     49.59%
  4     91.45%


p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

A.3.10 Maximum Order Complexity Test

The maximum order complexity test determines the length of the shortest possibly non-linear feedback shift register which can produce the given bit sequence. For the MOC profile, this is done for the first 1, 2, 3, . . . bits of the sequence. The changes in this profile are studied.

```
Maximum Order Complexity (MOC) Test for
ARIA Block Cipher in OFB mode


The changes in the MOC profile:
    2    1   ( 2.00)
    4    2   ( 4.00)
    7    4   ( 5.61)
   16    6   ( 8.00)
   20    7   ( 8.64)
   30   13   ( 9.81)
  207   15   (15.39)
  306   17   (16.51)
  467   18   (17.73)
 1066   20   (20.12)
 2137   21   (22.12)
 3135   23   (23.23)
 3160   24   (23.25)
 6240   26   (25.21)
 8833   27   (26.22)
12573   28   (27.24)
34820   30   (30.18)
40466   31   (30.61)
79864   32   (32.57)
176634   34   (34.86)
297505   38   (36.37)


The number of inputcharacters: 1000000
The number of nodes: 1999958
The number of edges: 2754619

The MOC is: 38
```

A.3.11 Poker Test

The poker test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the word length of the test. The sequence of m-tuples is split up into subsequent, disjoint k-tuples of m-tuples. The poker test evaluates statistically how many of the m-tuples in a k-tuple are equal. The poker test is also applied to cyclic shifts of the original sequence.

```
POKER TEST for
ARIA Block Cipher in OFB mode
Sequencelength = 9999360 bits        Wordlength = 8 bits
Elements in a k-tuple: 128
Ideal distribution:       mean    variance
                      100.8801     13.9923
Real distribution:
```

Shift:	0	1	2	3	4	5	6	7
mean:	100.924	100.891	100.846	100.845	100.877	100.901	100.884	100.928
error:	0.04%	0.01%	-0.03%	-0.04%	-0.00%	0.02%	0.00%	0.05%
variance:	14.094	13.906	13.974	13.788	13.884	13.798	13.889	13.889
error:	0.73%	-0.61%	-0.13%	-1.46%	-0.77%	-1.39%	-0.73%	-0.74%
p=	61.02%	39.95%	59.36%	32.05%	25.75%	51.71%	77.19%	78.63%

```
p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

n=	9765								
65	0.0	0	0	0	0	0	0	0	0
66	0.0	0	0	0	0	0	0	0	0
67	0.0	0	0	0	0	0	0	0	0
68	0.0	0	0	0	0	0	0	0	0
69	0.0	0	0	0	0	0	0	0	0
70	0.0	0	0	0	0	0	0	0	0
71	0.0	0	0	0	0	0	0	0	0
72	0.0	0	0	0	0	0	0	0	0
73	0.0	0	0	0	0	0	0	0	0
74	0.0	0	0	0	0	0	0	0	0
75	0.0	0	0	0	0	0	0	0	0
76	0.0	0	0	0	0	0	0	0	0
77	0.0	0	0	0	0	0	0	0	0
78	0.0	0	0	0	0	0	0	0	0
79	0.0	0	0	0	0	0	0	0	0
80	0.0	0	0	0	0	0	0	0	0
81	0.0	0	0	0	0	0	0	0	0
82	0.0	0	0	0	0	0	0	0	0
83	0.0	0	0	0	0	0	0	0	0
84	0.1	0	0	0	0	0	0	0	0
85	0.2	0	0	0	0	0	0	0	0
86	0.5	0	0	0	0	1	1	0	0
87	1.4	1	1	0	0	3	1	2	2
88	3.4	5	3	0	4	5	3	4	7
89	7.9	5	7	5	4	6	14	9	5
90	17.1	9	18	19	22	16	17	13	10
91	34.6	33	28	31	34	31	28	39	32
92	65.6	77	63	72	74	63	63	56	62
93	116.4	106	124	125	108	147	106	118	132
94	193.7	207	190	200	200	173	196	182	188

95	301.5	300	293	298	316	289	294	291	282
96	439.3	420	423	470	425	445	413	439	430
97	598.4	582	629	575	556	574	570	589	590
98	761.8	781	752	790	807	768	790	811	746
99	905.7	947	921	923	901	900	941	914	907
100	1004.9	983	991	1006	1023	1037	1015	993	982
101	1039.5	993	1048	1017	1107	1038	1042	1068	1057
102	1001.7	963	988	1013	1014	1001	994	1035	1045
103	898.3	910	924	856	840	917	880	834	875
104	748.7	768	743	783	713	725	801	729	779
105	579.2	566	583	546	577	581	541	567	577
106	415.3	447	442	389	437	440	440	422	408
107	275.6	290	234	289	266	247	268	299	296
108	168.9	182	158	170	167	188	164	166	170
109	95.5	101	108	100	87	84	87	98	90
110	49.6	52	45	50	52	56	55	44	54
111	23.7	22	21	21	17	16	28	28	22
112	10.3	9	19	12	8	10	8	11	8
113	4.1	5	7	3	4	3	3	1	7
114	1.5	1	1	2	2	1	2	2	2
115	0.5	0	1	0	0	0	0	1	0
116	0.1	0	0	0	0	0	0	0	0
117	0.0	0	0	0	0	0	0	0	0
118	0.0	0	0	0	0	0	0	0	0
119	0.0	0	0	0	0	0	0	0	0
120	0.0	0	0	0	0	0	0	0	0
121	0.0	0	0	0	0	0	0	0	0
122	0.0	0	0	0	0	0	0	0	0
123	0.0	0	0	0	0	0	0	0	0
124	0.0	0	0	0	0	0	0	0	0
125	0.0	0	0	0	0	0	0	0	0
126	0.0	0	0	0	0	0	0	0	0
127	0.0	0	0	0	0	0	0	0	0
128	0.0	0	0	0	0	0	0	0	0

A.3.12 Rank Test

In the rank test, the bits of the sequence to test are used to fill square matrices. The bits are treated as elements of the field $GF(2)$, and the ranks of the matrices are evaluated statistically.

```
Rank test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Order of the matrix: 16
Number of ranks counted individually: 3
 11308 matrices with rank 16, expected: 11280.8
 22398 matrices with rank 15, expected: 22561.3
 5142 matrices with rank 14, expected: 5013.5
 214 matrices with rank 13 or less, expected: 206.4
chisquare = 4.819368 nu = 3
Percentage level of acceptance 18.55
```

A.3.13 Run Test

The run test splits up the bit sequence into subsequent, disjoint m-tuples of bits. m is called the word length of the test. The m-tuples are interpreted as binary representations of numbers. The lengths of subsequences of consecutive, strictly increasing numbers are evaluated statistically.

```
Run Test for
ARIA Block Cipher in OFB mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 16
Maximal run length registered individually: 5
Total of 229976 runs
 115081 runs of length 1  expected: 114989.8
 76502 runs of length 2  expected: 76658.7
 28919 runs of length 3  expected: 28746.1
  7582 runs of length 4  expected: 7665.3
  1548 runs of length 5  expected: 1596.8
   344 runs of length 6 or more expected: 319.3
chisquare = 5.735269 nu = 5
Percentage level of acceptance 33.28
```

A.3.14 Ziv Lempel Test

The Ziv Lempel complexity test measures how well a bit sequence can be reconstructed from earlier parts of the bit sequence.

```
ARIA Block Cipher in OFB mode1000000 input bits of the input file have been handled.
The Ziv Lempel complexity equals 50722.
((1000000 / log2(1000000)) = 50171.665944)

A sequence of length n is considered to be a good pseudo-random
sequence if its Ziv Lempel complexity is greater than n/log2(n).

The maximum length of a component in the history equals 34.
(log2(1000000) = 19.931569)
```

A.4 Aria in CTR Mode

(For a short explanation of the tests, we refer to Section A.3.)

A.4.1 Collision Test

```
Collision Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 24
# of blocks: 416666 blocksize: 24 collisions: 5037
```

A.4.2 Correlation Test

```
Correlation Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 1000000
Printlevel: 0.000010
shift:0 equal: 1000000 probability: 1.0000000   0e+00
shift:54885 equal: 502136 probability: 0.9999904   1e-05
shift:140731 equal: 502204 probability: 0.9999948   5e-06
shift:171111 equal: 497510 probability: 0.0000003   3e-07
shift:251253 equal: 497660 probability: 0.0000014   1e-06
shift:265161 equal: 502193 probability: 0.9999942   6e-06
shift:292623 equal: 497834 probability: 0.0000074   7e-06
shift:332018 equal: 502512 probability: 0.9999997   3e-07
shift:483799 equal: 497713 probability: 0.0000024   2e-06
shift:531567 equal: 502174 probability: 0.9999932   7e-06
shift:547000 equal: 502161 probability: 0.9999923   8e-06
shift:571434 equal: 497844 probability: 0.0000081   8e-06
shift:712586 equal: 502141 probability: 0.9999908   9e-06
shift:847853 equal: 497843 probability: 0.0000081   8e-06
shift:883874 equal: 497827 probability: 0.0000070   7e-06
shift:892412 equal: 497782 probability: 0.0000046   5e-06
shift:959250 equal: 502222 probability: 0.9999956   4e-06
shift:994331 equal: 502138 probability: 0.9999905   9e-06
```

A.4.3 Coupon Collector's Test

```
COUPON COLLECTOR'S TEST for
ARIA Block Cipher in CTR mode
Sequencelength = 10000000 bits        Wordlength = 8 bits
Ideal distribution:       mean    variance
                     1567.8323   105979.0660
Real distribution:
The results are for cyclic shifts
Shift:              0        1        2        3        4        5        6        7
             ------------------------------------------------------------------------
 mean:        1568.305 1577.630 1577.013 1557.544 1551.794 1577.855 1546.583 1575.942
 error:          0.03%    0.62%    0.59%   -0.66%   -1.02%    0.64%   -1.36%    0.52%
 variance:      101936   104888   111366   102445   101865   103196    97178   101377
 error:         -3.82%   -1.03%    5.08%   -3.33%   -3.88%   -2.63%   -8.30%   -4.34%
```

```
    p=          25.78%   76.83%   38.74%   94.13%   60.54%    2.45%   79.23%   90.74%

p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

A.4.4 Fast Spectral Test

```
Fast Spectral Test for
ARIA Block Cipher in CTR mode


The results are:

The statistic D(4) = 2.563838E-01; percentage level of significance:  60.1%
The statistic D(6) = 1.710740E-01; percentage level of significance:  56.8%
```

A.4.5 Frequency Test

```
Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 12
sequencelength= 10000000 blocksize= 12
chisquare = 4044.719399  nu= 4095
Percentage Level of Acceptance:  70.90


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 16
sequencelength= 10000000 blocksize= 16
chisquare = 65672.264806  nu= 65535
Percentage Level of Acceptance:  35.17


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 1
sequencelength= 10000000 blocksize= 1
block: 0  count: 4996829
block: 1  count: 5003171
chisquare = 4.022096  nu= 1
Percentage Level of Acceptance:  4.49


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 2
sequencelength= 10000000 blocksize= 2
```

```
block: 0  count: 1248111
block: 1  count: 1251196
block: 2  count: 1250787
block: 3  count: 1249906
chisquare = 4.501554  nu= 3
Percentage Level of Acceptance:  21.22


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 3
sequencelength= 10000000 blocksize= 3
block: 0  count: 416774
block: 1  count: 416268
block: 2  count: 416447
block: 3  count: 417067
block: 4  count: 417602
block: 5  count: 417038
block: 6  count: 416692
block: 7  count: 415445
chisquare = 6.923578  nu= 7
Percentage Level of Acceptance:  43.69


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 4
sequencelength= 10000000 blocksize= 4
block: 0  count: 156542
block: 1  count: 155889
block: 2  count: 156863
block: 3  count: 156415
block: 4  count: 156146
block: 5  count: 155621
block: 6  count: 156211
block: 7  count: 156038
block: 8  count: 156445
block: 9  count: 156250
block: 10  count: 156139
block: 11  count: 156511
block: 12  count: 156238
block: 13  count: 156465
block: 14  count: 156000
block: 15  count: 156227
chisquare = 8.315942  nu= 15
Percentage Level of Acceptance:  91.05


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 5
sequencelength= 10000000 blocksize= 5
block: 0  count: 62778
block: 1  count: 62388
block: 2  count: 62392
block: 3  count: 62733
block: 4  count: 62780
block: 5  count: 62300
block: 6  count: 62432
```

```
block: 7  count: 62642
block: 8  count: 62530
block: 9  count: 62118
block: 10  count: 62345
block: 11  count: 62439
block: 12  count: 62898
block: 13  count: 62225
block: 14  count: 62430
block: 15  count: 62442
block: 16  count: 62487
block: 17  count: 62844
block: 18  count: 62449
block: 19  count: 62439
block: 20  count: 62630
block: 21  count: 62775
block: 22  count: 62416
block: 23  count: 62527
block: 24  count: 62803
block: 25  count: 62218
block: 26  count: 61925
block: 27  count: 62757
block: 28  count: 62454
block: 29  count: 62564
block: 30  count: 62352
block: 31  count: 62488
chisquare = 24.595776  nu= 31
Percentage Level of Acceptance:  78.55


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 6
sequencelength= 10000000 blocksize= 6
chisquare = 61.602166  nu= 63
Percentage Level of Acceptance:  52.63


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 7
sequencelength= 10000000 blocksize= 7
chisquare = 107.662165  nu= 127
Percentage Level of Acceptance:  89.23


Frequency Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 8
sequencelength= 10000000 blocksize= 8
chisquare = 253.698662  nu= 255
Percentage Level of Acceptance:  51.12
```

A.4.6 Gap Test

```
GAP TEST for
ARIA Block Cipher in CTR mode
```

```
Sequencelength = 10000000 bits        Wordlength = 10 bits
Length of gaps between occurences in the range 256 - 768
Ideal distribution:       mean    variance
                         1.000       2.000
Real distribution:
```

Shift:	0	1	2	3	4	5	6	7	8	9
mean:	1.001	1.001	0.999	0.996	1.003	1.001	1.000	0.998	1.003	1.002
error:	0.05%	0.09%	-0.06%	-0.40%	0.26%	0.10%	-0.05%	-0.19%	0.30%	0.22%
variance:	2.004	2.005	1.997	1.982	2.004	2.007	1.992	1.989	2.015	2.014
error:	0.22%	0.26%	-0.16%	-0.92%	0.18%	0.36%	-0.39%	-0.56%	0.76%	0.70%
p=	82.00%	65.96%	77.71%	45.91%	2.61%	48.22%	50.56%	44.18%	18.59%	25.38%

```
p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

		499867	499781	500139	500992	499353	499758	500121	500487	499250	499452
0	5.00e-01	249908	250177	249923	250457	249047	249883	249972	250038	249240	249481
1	2.50e-01	125207	124391	125312	125871	125236	125081	125130	125699	124995	125050
2	1.25e-01	62120	62579	62593	62515	62609	62135	62468	62326	62327	62341
3	6.25e-02	31285	31218	31005	31334	30972	31182	31226	31540	31395	31320
4	3.12e-02	15671	15662	15685	15375	15653	15816	15852	15359	15631	15525
5	1.56e-02	7785	7929	7917	7691	8118	7729	7739	7798	7724	7739
6	7.81e-03	3934	3879	3821	3889	3889	4011	3920	3837	3887	4013
7	3.91e-03	1973	2042	1961	1952	1857	1965	1912	1955	2027	1952
8	1.95e-03	990	959	940	961	992	973	933	971	1017	997
9	9.77e-04	499	470	488	472	485	485	502	479	505	498
10	4.88e-04	259	231	239	230	257	254	228	242	229	287
11	2.44e-04	131	118	124	129	112	128	128	125	133	125
12	1.22e-04	48	61	70	61	63	63	48	54	59	59
13	6.10e-05	32	32	28	32	28	22	27	35	48	30
14	3.05e-05	12	19	18	10	22	10	18	16	16	23
15	1.53e-05	9	5	5	8	5	11	12	4	8	4
16	7.63e-06	1	3	4	0	4	5	4	6	4	4
17	3.81e-06	1	3	3	2	2	3	1	2	4	1
18	1.91e-06	1	1	1	2	1	0	1	1	0	2
19	9.54e-07	1	1	1	1	0	1	0	0	1	1
20	9.54e-07	0	1	1	0	1	1	0	0	0	0

A.4.7 Linear Complexity Test

```
Linear Complexity Test for
ARIA Block Cipher in CTR mode


-------------------- Final results ------------------

N= 100000      L= 50000      X= 2

N    is the number of input bits.

L    is the linear complexity.

X-1  is the number of bits which has been treated since
     the last change of linear complexity.

------------------------ End ----------------------
The linear complexity profile:
```

```
Jumps in the linear complexity profile:



ssl =  99999.000    ssqsl = 500385.000
msl =      4.005    varsl =      4.000



ssh =  50000.000    ssqsh = 149742.000
msh =      2.003    varsh =      1.987



  ssl   is the sum of the sl's
  ssqsl is the sum of the squares of the sl's
  msl   is the mean of the sl's
  varsl is the variance of the sl's
  The number of jumps used in the calculation of msl and varsl is:  24967

  ssh   is the sum of the sh's
  ssqsh is the sum of the squares of the sh's
  msh   is the mean of the sh's
  varsh is the variance of the sh's
  The number of jumps used in the calculation of msh and varsh is:  24966
maximal step-height 16.000000


  sl  is the steplength
  sh  is the stepheigth
  nj  is the number of jumps
```

A.4.8 Maurer Test

```
Maurer Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 1000000
Block size: 8
Initial Blocks= 10000
blocks tested (including initial blocks): 125000
Maurer Test Value: 7.186712
```

A.4.9 Overlapping m-tuple Test

```
OVERLAPPING M-TUPLE TEST for
ARIA Block Cipher in CTR mode
Sequencelength = 10000000 bits        Wordlength = 5 bits
m = 2

Shift     p
--------------
  0     33.43%
  1     66.34%
  2     81.14%
  3     98.08%
```

```
  4     92.39%


p gives the percentage level of acceptance of the chi-square test
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

A.4.10 Maximum Order Complexity Test

```
Maximum Order Complexity (MOC) Test for
ARIA Block Cipher in CTR mode


The changes in the MOC profile:
    4    3   ( 4.00)
    8    4   ( 6.00)
   13    6   ( 7.40)
   21    8   ( 8.78)
   70   10   (12.26)
  100   12   (13.29)
  203   15   (15.33)
  471   16   (17.76)
 1480   18   (21.06)
 1609   21   (21.30)
 2569   25   (22.65)
 4117   26   (24.01)
18566   28   (28.36)
53522   29   (31.42)
68436   30   (32.12)
73208   36   (32.32)
337039   37   (36.73)


The number of inputcharacters: 1000000
The number of nodes: 1999963
The number of edges: 2755189

The MOC is: 37
```

A.4.11 Poker Test

```
POKER TEST for
ARIA Block Cipher in CTR mode
Sequencelength = 9999360 bits        Wordlength = 8 bits
Elements in a k-tuple: 128
Ideal distribution:       mean    variance
                      100.8801     13.9923
Real distribution:
Shift:            0       1       2       3       4       5       6       7
            ---------------------------------------------------------------
 mean:       100.874 100.879 100.862 100.965 100.942 100.908 100.893 100.929
 error:        -0.01%  -0.00%  -0.02%   0.08%   0.06%   0.03%   0.01%   0.05%
 variance:    13.897  13.803  13.815  13.935  13.527  13.766  13.760  13.715
 error:        -0.68%  -1.35%  -1.27%  -0.41%  -3.32%  -1.62%  -1.66%  -1.98%

    p=         68.77%  39.08%  83.46%  17.98%  74.42%  43.26%  26.80%   7.68%

p gives the percentage level of acceptance of the chi-square test
```

```
This percentage level gives the probability that a truly random
sequence has a chi-square value greater than the chi-square value
observed in this execution of the test.
```

n= 9765

65	0.0	0	0	0	0	0	0	0	0
66	0.0	0	0	0	0	0	0	0	0
67	0.0	0	0	0	0	0	0	0	0
68	0.0	0	0	0	0	0	0	0	0
69	0.0	0	0	0	0	0	0	0	0
70	0.0	0	0	0	0	0	0	0	0
71	0.0	0	0	0	0	0	0	0	0
72	0.0	0	0	0	0	0	0	0	0
73	0.0	0	0	0	0	0	0	0	0
74	0.0	0	0	0	0	0	0	0	0
75	0.0	0	0	0	0	0	0	0	0
76	0.0	0	0	0	0	0	0	0	0
77	0.0	0	0	0	0	0	0	0	0
78	0.0	0	0	0	0	0	0	0	0
79	0.0	0	0	0	0	0	0	0	0
80	0.0	0	0	0	0	0	0	0	0
81	0.0	0	0	0	0	0	0	0	0
82	0.0	0	0	0	0	0	0	0	0
83	0.0	0	0	0	0	0	0	0	0
84	0.1	0	0	0	0	0	0	0	0
85	0.2	0	0	0	0	0	0	0	0
86	0.5	0	2	0	0	0	0	0	0
87	1.4	3	0	3	1	3	1	1	0
88	3.4	4	3	5	9	2	2	4	1
89	7.9	5	8	9	10	7	4	10	5
90	17.1	14	13	11	14	17	14	17	15
91	34.6	41	43	28	33	26	32	24	34
92	65.6	77	71	70	64	68	51	51	73
93	116.4	110	130	125	111	102	112	138	129
94	193.7	198	179	198	174	188	190	177	178
95	301.5	310	274	291	316	302	299	310	246
96	439.3	413	423	407	423	390	444	410	401
97	598.4	562	608	620	542	594	610	592	632
98	761.8	775	732	758	728	717	766	783	751
99	905.7	943	888	925	917	921	958	896	925
100	1004.9	978	1057	1004	1004	1034	976	1016	1035
101	1039.5	1049	1068	1067	1039	1070	1028	1092	1085
102	1001.7	1014	1024	1010	1017	1011	1051	1049	981
103	898.3	936	911	922	886	909	812	822	878
104	748.7	725	752	750	820	780	748	739	748
105	579.2	580	601	571	612	593	607	589	602
106	415.3	412	370	370	399	435	410	431	408
107	275.6	253	270	262	272	258	291	265	295
108	168.9	186	148	172	181	163	174	161	150
109	95.5	100	94	92	111	96	98	95	94
110	49.6	44	50	50	52	44	48	50	54
111	23.7	18	23	28	19	20	21	26	30
112	10.3	8	15	12	6	9	12	11	8
113	4.1	4	6	1	1	5	6	6	4
114	1.5	3	0	3	2	1	0	0	2
115	0.5	0	1	0	0	0	0	0	1
116	0.1	0	1	1	1	0	0	0	0
117	0.0	0	0	0	1	0	0	0	0
118	0.0	0	0	0	0	0	0	0	0
119	0.0	0	0	0	0	0	0	0	0
120	0.0	0	0	0	0	0	0	0	0
121	0.0	0	0	0	0	0	0	0	0
122	0.0	0	0	0	0	0	0	0	0
123	0.0	0	0	0	0	0	0	0	0

124	0.0	0	0	0	0	0	0	0	0
125	0.0	0	0	0	0	0	0	0	0
126	0.0	0	0	0	0	0	0	0	0
127	0.0	0	0	0	0	0	0	0	0
128	0.0	0	0	0	0	0	0	0	0

A.4.12 Rank Test

```
Rank test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Order of the matrix: 16
Number of ranks counted individually: 3
 11154 matrices with rank 16, expected: 11280.8
 22619 matrices with rank 15, expected: 22561.3
 5071 matrices with rank 14, expected: 5013.5
 218 matrices with rank 13 or less, expected: 206.4
chisquare = 2.880692 nu = 3
Percentage level of acceptance 41.04
```

A.4.13 Run Test

```
Run Test for
ARIA Block Cipher in CTR mode
Number of bits generated and ignored before starting to test: 0
Number of bits used for testing: 10000000
Block size: 16
Maximal run length registered individually: 5
Total of 230147 runs
 115236 runs of length 1  expected: 115075.3
 76813 runs of length 2  expected: 76715.7
 28648 runs of length 3  expected: 28767.5
  7568 runs of length 4  expected: 7671.0
  1571 runs of length 5  expected: 1598.0
   311 runs of length 6 or more expected: 319.6
chisquare = 2.914010 nu = 5
Percentage level of acceptance 71.32
```

A.4.14 Ziv Lempel Test

```
ARIA Block Cipher in CTR mode1000000 input bits of the input file have been handled.
The Ziv Lempel complexity equals 50787.
((1000000 / log2(1000000)) = 50171.665944)

A sequence of length n is considered to be a good pseudo-random
sequence if its Ziv Lempel complexity is greater than n/log2(n).

The maximum length of a component in the history equals 39.
(log2(1000000) = 19.931569)
```