A Key Recovery Attack on Edon80

Martin Hell and Thomas Johansson

Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{martin, thomas}@eit.lth.se

Abstract. Edon80 is a recent stream cipher design that has advanced to the third and last phase of the eSTREAM project. It has remained unbroken and untweaked since it was designed and submitted to eSTREAM. It is now one of the 8 final hardware candidates. In this paper we cryptanalyze the cipher by describing a key recovery attack. The complexity of the attack is around 2^{69} simple operations for a keystream of similar length.

1 Introduction

Edon80 is a recent stream cipher design, described in [1], that was submitted to the eSTREAM project. It uses a novel approach in stream cipher design, concatenating 80 basic building blocks derived from 4 different quasigroups of order 4. A quasigroup is basically a Latin square, a very simple combinatorial object.

The design has received a lot of attention and much work has been done based on Edon80. Regarding security, Hong observed in [2] that with some small probability, the period of the keystream sequence could be quite small. This was further studied by the designers themselves in [3] and later also in the paper [4]. However, this property could not be exploited in any kind of attack. A theoretical treatment of the quasigroups used in Edon80 is given in [5]. Finally, from an implementations point of view, it was shown in [6] that Edon80 can be implemented using less than 3000 gates. Even though the eSTREAM project has allowed tweaks, the Edon80 construction has remained untweaked since it was designed and submitted to eSTREAM. However, due to the probability of short periods, the designers has introduced a limitation in the number of keystream bits that can be produced per key/IV pair. This limitation is 2^{48} bits and was proposed in [7], when entering the second phase of eSTREAM.

The small implementation and the fact that the construction has remained untweaked are the main reasons for the success of Edon80 in eSTREAM – its advancement to the third and last phase phase of the eSTREAM project. It is now one of the 8 final hardware candidates.

In this paper we cryptanalyze the cipher by describing a key recovery attack. The complexity of the attack is around 2^{69} for a keystream of similar length. The design philosophy is not completely broken. A design using, say, 160 concatenated quasigroup operations would be out of scope of the new attack. On

the other hand, such a change of the design would double the implementation cost, making such a design much less interesting.

The new attack to be presented is based on exploiting some periodicity inside the generator. Using the fact that some elements will repeat with large probability, we can build a kind of test to find out the correct value of the key bits used at the end of the concatenation. This leads to a key recovery attack, where we may vary some parameters and obtain a trade-off between required length of the received key stream and the computational complexity.

The paper is organized as follows. In Section 2 we describe in more detail the stream cipher design Edon80. In Section 3 we summarize some previous work relating to the security of Edon80. In Section 4 we then give the basic ideas of the new attack, followed by a more detailed analysis in Section 5. In Section 6 we discuss how the attack can be efficiently implemented. In Section 7 we verify some of the claims by presenting simulation results. Finally, in Section 8 we derive some possible attack complexities and then we conclude.

2 Description of Edon80

In this section we give a description of the Edon80 stream cipher. An additive synchronous stream cipher is built around a keystream generator. A generator takes a key K and an IV value (nonce) IV as its input and produces an arbitrary long keystream sequence $Z = z_1, z_2, z_3, \ldots$. The keystream is then added to the plaintext in the encryption phase.

The sizes of the key and IV in Edon80 are 80 bits and 64 bits, respectively. The design of Edon80 is based on string transformation using 4 quasigroups of order 4 denoted $(Q, \bullet_j)$ $(0 \leq j \leq 3)$. The internal updated state consists of 80 memory cells of two bits each. Each memory cell, referred to as an e-transformer $*_i$ $(0 \leq i \leq 79)$, holds 2 bits representing a value between 0 and 3. The 80 e-transformers are connected in series and the result from $*_i$ is used as input to $*_{i+1}$.

The 80 bit key K is divided into 40 2-bit values $K = K_0 K_1 \ldots K_{39}$ each represented as a value $0 \leq K_i \leq 3$. The quasigroup $(Q, *_i)$, $(0 \leq i \leq 79)$ used by e-transformer $*_i$ is given by

$$(Q, *_i) \leftarrow \begin{cases} (Q, \bullet_{K_i}) & 0 \leq i \leq 39, \\ (Q, \bullet_{K_{i-40}}) & 40 \leq i \leq 79. \end{cases}$$

The quasigroups used in Edon80 are given in Figure 1.

Let the value in $*_i$ at time t be denoted $a_{i,t}$. Then the values are updated as

$$\begin{cases} a_{0,0} = a_0 *_0 0, & \\ a_{0,j} = a_{0,j-1} *_0 (j \mod 4), & 1 \leq j, \\ a_{i,0} = a_i *_i a_{i-1,0}, & 1 \leq i \leq 79, \\ a_{i,j} = a_{i,j-1} *_i a_{i-1,j}, & 1 \leq i \leq 79,\ 1 \leq j, \end{cases}$$

where a_i denotes the initial value of $*_i$ for $1 \leq i \leq 79$ at the beginning of the keystream generation phase.

$\bullet_0$	0	1	2	3
0	0	2	1	3
1	2	1	3	0
2	1	3	0	2
3	3	0	2	1

$\bullet_1$	0	1	2	3
0	1	3	0	2
1	0	1	2	3
2	2	0	3	1
3	3	2	1	0

$\bullet_2$	0	1	2	3
0	2	1	0	3
1	1	2	3	0
2	3	0	2	1
3	0	3	1	2

$\bullet_3$	0	1	2	3
0	3	2	1	0
1	1	0	3	2
2	0	3	2	1
3	2	1	0	3

Fig. 1. The 4 quasigroups used in Edon80.

Summarizing, the infinite period 4 string $0, 1, 2, 3, 0, 1, 2, 3, 0, \ldots$ is transformed by $*_0$ and the resulting string is transformed by $*_1$ etc. The keystream is obtained by taking every second value produced by $*_{79}$, see Figure 2.

$*_i$		0	1	2	3	0	1	2	3	0
$*_0$	a_0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$	$a_{0,8}$
$*_1$	a_1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$	$a_{1,8}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$*_{79}$	a_{79}	$a_{79,0}$	$\boxed{a_{79,1}}$	$a_{79,2}$	$\boxed{a_{79,3}}$	$a_{79,4}$	$\boxed{a_{79,5}}$	$a_{79,6}$	$\boxed{a_{79,7}}$	$a_{79,8}$

Fig. 2. The quasigroup string e-transformation in keystream generation mode.

For simplicity, we adopt the notation $Z = z_1, z_3, z_5, \ldots$ as the received keystream, where

$$z_t = a_{79,t} \quad t \geq 0, \quad t \text{ odd}.$$

A schematic picture of Edon80 is given in Figure 3. Remember that only every second output is used in the keystream.

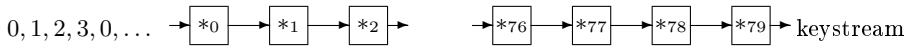

Fig. 3. The keystream generator Edon80.

The initial state of Edon80, $(a_0, a_1, \ldots, a_{79})$, is determined by the key K and the IV through an IV setup process. Exactly how this is done is not relevant in our analysis and we refer to the design document [1] for a detailed description of the IV setup. We can assume that the mapping from the 80-bit key and the 64-bit IV to the initial state $a_0, a_1, \ldots, a_{79}$ is a random mapping. However, the attack will still be applicable even if the mapping would be shown to suffer from some nonrandomness.

Edon80 is designed to be a hardware efficient stream cipher. The hardware description is slightly different from the algorithmic description given above. In

order to output 1 bit/clock, the implementation uses a second 2-bit memory cell in $*_i$ which stores the output from $*_{i-1}$. Though, in [6] the authors demonstrated an implementation which does not use this extra memory cell. The implementation required only a gate count of about 3000 but the output was decreased to 1/80 bit/clock resulting in a throughput of just a few Mbit/s. However, this small implementation cost shows that Edon80 is a very interesting candidate for a stream cipher suitable for constrained environments.

3 Previous Analysis of Edon80

In this section we review the previous results and known properties of Edon80 that will be used in our cryptanalysis. The most important property that will be exploited in the attack is the relatively short period of Edon80. In the design document [1] it was stated that the expected average period of the keystream is about 2^{103}. In [2], Hong argued that there are many key/IV pairs that produce a keystream with undesirably short period. Referring to Figure 2, using exhaustive search all d-row key/state pairs of period $p = 4, 8$ and 16 was found. Extrapolating the results to 40 rows, and then repeating the same key for the lower 40 rows, it was concluded that there are many key/IV pairs that produce a keystream with relatively short period. As an example, it was claimed that there is a 2^{-75} probability that a key/IV pair generates a keystream with period 2^{61}. In response to these results, the designers claimed in [3] that the values given by Hong was actually underestimated and that the probability of generating a keystream with period less than 2^{61} was $2^{-18.62}$. Thus, with a total of $2^{79.62}$ bits we can expect to find a sequence with period less than 2^{61}. Further, it was concluded that the average period of Edon80 is 2^{91}. A more detailed investigation of the periods was given in [4]. Each e-transformer increases the period of the incoming string by a factor 1, 2, 3 or 4. Let X_i denote the factor by which e-transformer $*_i$ increases the period. Considering several consecutive e-transformers, it was shown that the probability distribution for X_i converges to the stationary distribution

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 \\ \frac{1}{4} & \frac{1}{4} & \frac{11}{32} & \frac{5}{32} \end{pmatrix},$$

with expected value $E(X) = \frac{77}{32}$ and variance $\sigma^2 = V(X) = \frac{1079}{1024}$. Furthermore, let $2m$ be the total number of e-transformers and let P_{2m} be a random variable for the period after $2m$ e-transformers. Then when $m \to \infty$, probability density function (pdf) $f_{P_{2m}}$ can be approximated by the continuous function [4, section 2]

$$f_{P_{2m}}(s) = \frac{1}{0.701658s\sqrt{2\pi m}} \exp\left(-\frac{(\ln(s) - 1.535086m)^2}{0.984648m}\right), \quad 0 < s < \infty. \quad (1)$$

We refer to [4] for more details. Despite the relatively high probability of short periods, it has until now been unclear how to use this to obtain information about the key.

4 A Key Recovery Attack – Basic Ideas

In this section we give the ideas behind our key recovery attack on Edon80. The details are then given in Section 5. We assume a known plaintext scenario i.e., the keystream sequence $Z = z_1, z_3, z_5, \ldots$ is known to the adversary. The basic ideas behind the attack are based on the following properties of the cipher,

- The quasigroup $(Q, \bullet_j)$ $(0 \leq j \leq 3)$ used in e-transformer $*_i$ $(0 \leq i \leq 79)$ is completely determined by the key. For example, if we know which quasigroup is used in the last e-transformer, we also know 2 key bits.
- The period of the string produced by $*_i$ can be expected to be moderately small for small i. In fact, some internal values (output from e-transformers) will repeat with large probability due to the periodicity.

We visualize the attack in Figure 4 by considering a matrix with elements $a_{i,j}$, $(0 \leq i \leq 79, t \leq j \leq t+u+v)$, u, v to be defined later. Every column here corresponds to one specific time instance t. Also, the ith row corresponds to the ith e-transformer. Thus we have 80 rows in the Edon80 description. A restriction to the first B rows simply corresponds to an Edon instance with only B e-transformers.

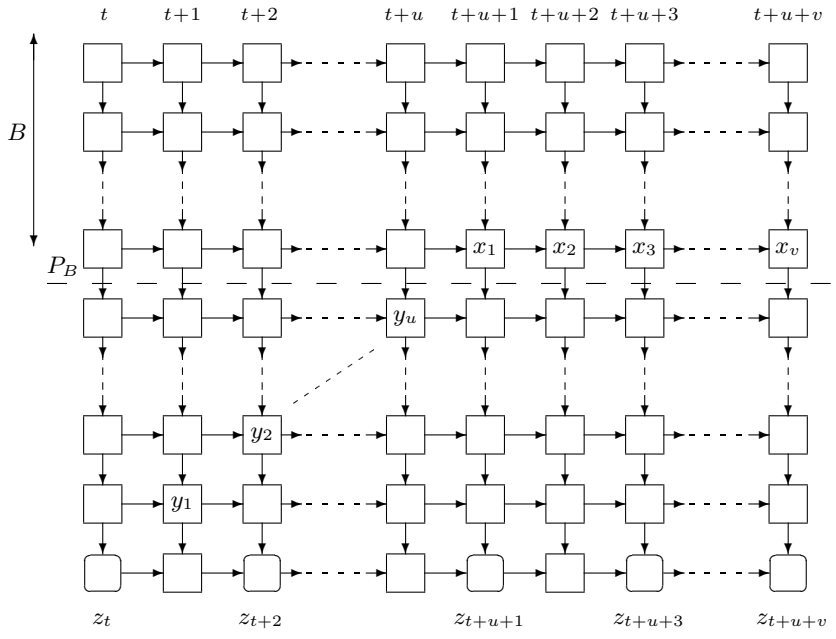

Fig. 4. Visualization of the attack idea.

Looking at a specific value $a_{i,j}$, this value is calculated from its neighbours to the left and above. I.e., the value at position (i, j) will depend on all values at positions (i', j') for $i' < i$ and $j' < j$, i.e., all values above and to the left in the matrix.

In order to set up the attack, we select the B top rows as one part (upper part) and the remaining rows below as a second part (lower part) of the e-transformers. Consider two vectors, X and Y of length $v = |X|$ and $u = |Y|$ respectively,

$$X = (x_1, x_2, \ldots, x_v),$$
$$Y = (y_1, y_2, \ldots, y_u),$$

$x_i, y_j \in \{0, 1, 2, 3\}, i = 1, 2, \ldots v; j = 1, 2, \ldots, u$, with the values located as shown in Figure 4. For Edon80, we then have $B = 80 - u - 1$. As can be seen, the $X = (x_1, x_2, \ldots, x_v)$ vector is simply v symbols coming out of the chain of B e-transformers starting from some predetermined time. The $Y = (y_1, y_2, \ldots, y_u)$ can be characterized as the values needed to compute the internal state of the second part of the e-transformers.

Each quasigroup transformation will increase the period of the initial string by a factor of 1, 2, 3 or 4. Thus the period, denoted P_i, of the sequence produced by $*_i$ is given by

$$P_i = 2^{\mu_1} 3^{\mu_2}, \tag{2}$$

for some $\mu_1, \mu_2 \in \mathbb{Z}$. Let P_B be the period of the sequence produced by the upper part of the e-transformers, giving output corresponding to the vector $X = (x_1, x_2, \ldots, x_v)$. Then, the matrix corresponding to time instance t and time instance $t + kP_B$, $k = 0, 1, 2, \ldots$ will have the same values in the e-transformers $*_i$ for $i \leq B$. More specifically, and which will be used in the attack, *the vector X will have the same value in all considered time instances.*

Assume for a moment that the key bits used to determine the quasigroups in the second part are known. With in total $u + v$ values in the vectors X and Y, we consider the $(u + v)/2$ known keystream symbols that are directly below X and Y, see Figure 4 again. Using the knowledge of these keystream symbols, the number of possible combinations of the two vectors X, Y will be reduced from 4^{u+v} to roughly 2^{u+v}. The idea is to choose u and v such that $v > u$. This means that not all X vectors will be possible in the set of possible X, Y pairs. Thus, the outcome of this part is a set Γ_k such that

$$\Gamma_k = \{X : \text{ there exists } (X, Y) \text{ matching } z_{t+kP_B}, z_{t+kP_B+2}, \ldots z_{t+kP_B+u+v}\}.$$

Finally, we combine this with the fact that the vector $X = (x_1, x_2, \ldots, x_v)$ will be the same at time instances t and $t+kP_B$. This means that X must appear in all sets Γ_k and hence in the intersection of them. The procedure should now be clear.

For each choice of the $2u + 2$ key bits used to define the quasigroups in the lower part, we determine the sets Γ_k, for $k = 0, 1, 2, \ldots$. We take the intersection between the sets obtained so far, and continue until the intersection is empty. If we eventually receive an empty intersection, the chosen value of the key bits

is discarded. On the other hand, if at the end there is only one vector X in the intersection, then we assume that we found the correct key bits. The number of key bits that are guessed in this attack is $2u + 2$. When we know these key bits, the remaining part of the key could be exhaustively searched.

5 A More Detailed Analysis of the Attack

In this section we give a more detailed analysis of the different parts and parameters used in the attack. The parameters that will be covered are

- Guessing the correct period P_B.
- The length of the vectors X and Y.
- The number of time instances that has to be considered in order to discard a wrong key candidate.

5.1 The Period P_B

As stated in (2), the period of the sequence after B e-transformers have the form $P_B = 2^{\mu_1}3^{\mu_2}$ for some μ_1, μ_2. It is clear that the X vector will repeat the same values if the distance between two matrices as described in Figure 4 is a multiple of the period. So we will assume a distance P'_B and the repetition of the value for the X vectors will be true if the actual period is a factor, i.e., if $P_B | P'_B$. We denote the probability that $P_B | P'_B$ by $\alpha_{P'_B}$. This value is, according to [4], approximately calculated as

$$\alpha_{P'_B} = \int_0^{P'_B} f_{P_{2m}}(s)\, ds, \tag{3}$$

where $f_{P_{2m}}(s)$ is defined in (1).

Recall that X_i denoted the factor by which e-transformer $*_i$ increases the period. In Section 3 we saw that the probability distribution for X_i converges to the distribution

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 \\ \frac{1}{4} & \frac{1}{4} & \frac{11}{32} & \frac{5}{32} \end{pmatrix}.$$

This gives us a rough idea of the expected period. For example, if $B = 64$ we can expect around 16 of the factors being 1, around the same number being 2, around 22 factors being 3, and around 10 factors being 4. So for $B = 64$ we can set $P'_B = 2^{36} \cdot 3^{22}$ and there is a fairly large chance that $P_B | P'_B$. The actual probability for different values of the period deviated slightly from the above since the probabilities are not as the asymptotic ones for low values of i. However, it can all be computed numerically.

5.2 The Length of Vectors X and Y

Assuming that we have chosen a value $B = 80 - u - 1$ and an assumed period P'_B such that $P_B | P'_B$, we now consider the choice of v. In order to create a set Γ_k where not all X vectors appear we need to choose $v > u$. We denote the difference by d, hence

$$v = u + d.$$

The simplest approach is then to start at time t and move forward. We assign all 4^2 possible values to y_1, y_2. We can then calculate everything below these positions in Figure 4. As we already know the value of z_{t+2}, only 4 of the possible candidates for y_1, y_2 will survive. For each surviving value of y_1, y_2, we assign all possible values for y_3, y_4, compute the values below and check against the known value of z_{t+4}. We will have 16 possibilities for the (y_1, y_2, y_3, y_4) vector. After finishing the Y vector we just continue in this fashion with $x_i, i = 1, \ldots x_v$. The set of possible assignments of (Y, X) is then 2^{u+v}. The complexity of calculating this set in this basic way is then roughly 2^{u+v}. Finally, the Y values are stripped off and the result is the set Γ_k. In an actual implementation we can make the constant factor in the algorithm very small. This will be described in more detail in Section 6.

5.3 The Number of Intersections Needed to Discard a Key Candidate

The total number of possible X vectors is 4^v. However, in the algorithm, using the knowledge of the keystream z_t, the vector X can only take 2^{u+v} values. Thus, using $v = u + d$, only a fraction $1/2^d$ of all values will be possible. Actually, in practice it is slightly less because some X vectors may appear twice (for different Y vectors). If we put

$$4^v \cdot \left(\frac{1}{2^d}\right)^K \approx 1,$$

we see that we need about $K \approx \frac{2v}{d}$ sets $\Gamma_k, k = 0, 1, \ldots, K-1$ to get an empty intersection. At least, the average number is around $2v/d$. As an example, for the choice $v = u + 2$ $(d = 2)$ there can be at most 25% of all the X vectors in Γ_k. Since the number of possible X vectors is 4^v we expect that we do not need much more than v sets.

In general, a higher value of d will increase the computational complexity but since the reduction of possible X values in an intersection is much higher, it will lead to a smaller number of required intersections and hence a shorter required keystream length.

5.4 Computational Complexity

Let us summarize the computational complexity of the attack. We assume first a value $B = 80 - u - 1$ and P'_B such that $P_B | P'_B$. There is an error probability, $1 - \alpha_{P'_B}$ that this assumption is not true.

Then we guess $2u+2$ key bits corresponding to the last $u+1$ quasigroups used. For each such key the complexity of checking it is then roughly $2^{u+v} \cdot K$. Since $v = u + d$ this results in a total complexity of about

$$2^{4u+d+3} \cdot \frac{u+d}{d}.$$

After recovering $2u+2$ key bits one can either reconstruct the sequence after B e-transformers and apply the same attack again, now with much less complexity; or simply do an exhaustive key search on the remaining key bits.

6 Algorithmic Aspects

In this section we describe some algorithmic aspects of the attack and show that the complexity is based on very simple operations, much faster than the operation of verifying a key candidate in exhaustive key search. The considerations here relate to the part of the attack that calculates the Γ_k sets.

Let $(a_{B+1,t}, a_{B+2,t}, \ldots, a_{79,t})$ be the state of the lower part of Edon80 at time t and denoted S_t. In Figure 4 this corresponds to a column starting below an x_i value.

In a straight forward algorithm we save all possible states S_t and the corresponding X vector. Each time a new $a_{79,t}$ (t even) is introduced, each state S_t with corresponding X vector will produce 4 new states. Each new state will have a corresponding X vector with 2 additional entries. Thus, at the end of the algorithm, we will have 2^{u+v} possible states and X vectors. We can note that the last step is the most expensive step. It will cost $C \cdot 2^{u+v}$ where C is the cost for making $2u+2$ table lookups. This constant can be significantly reduced by using a slightly different algorithm.

We can take advantage of the following observation. Since the length of the state vector S_t is $u+1$ there are in general 4^{u+1} possible values for the state of the lower part of Edon80 at any time. However, looking at the attack as illustrated in Figure 4, where we have a given keystream sequence $z_t, z_{t+2}, \ldots$, we observe that at any time instance (with a received keystream symbol) only $2^{2\lceil u/2 \rceil}$ different states of the second part of Edon80 are possible.

This property comes from the fact that we know every second of the values $z_t = a_{79,t}$. Knowing $a_{79,t}$ and $a_{79,t+2}$ and allowing 4 possible values for $a_{79,t+1}$ will give 4 possibilities for the pair $(a_{77,t+2}, a_{78,t+2})$. Knowing $a_{79,t}$, $a_{79,t+2}$ and $a_{79,t+4}$ and allowing 16 possibilities for $(a_{79,t+1}, a_{79,t+3})$ gives 16 possibilities for the vector $(a_{75,t+4}, a_{76,t+4}, a_{77,t+4}, a_{78,t+4})$ etc.

We can from the known keystream compute all 2^u possible states for times $t+u+1, t+u+3, \ldots$. We can then obtain a trellis by including all possible state transitions from time $t+u+1$ to $t+u+3$ and so on. A state transition from time $t+u+1$ to $t+u+3$ can be labelled by the values of (x_1, x_2) giving rise to that transition. This way of modelling the lower part of Edon80 is useful when we implement the algorithm for computing the Γ_k sets for a given choice of key bits.

We can divide the X vector in two equally sized parts, $X = (X_1, X_2)$, where $X_1 = (x_1, x_2, \ldots, x_{v/2})$ and $X_2 = (x_{v/2+1}, x_{v/2+2}, \ldots, x_v)$. We first assign Y and compute possible values of Y as before. This is actually equivalent to computing the state of the second part of Edon80 at time $t + u$, so when we continue we do not keep the value of Y but instead we keep the state S_t at the time we are considering. We continue as before, but only over the X_1 vector. This results in a set of possible X_1 vectors and their ending states $S_{t+u+v/2}$. The complexity of calculating this set is then $C \cdot 2^{u+v/2}$. Next, for every choice of the 2^u possible states $S_{t+u+v/2}$ at time $t + u + v/2$, we assign all possible values for $x_{v/2+1}, x_{v/2+2}, \ldots$, and create a second set of all possible X_2 vectors and their starting states $S_{t+u+v/2}$. The complexity of calculating this second set is also $C \cdot 2^{u+v/2}$. Thus, calculating the two sets is much faster than finding Γ_k in the straight forward algorithm.

The bottle neck in this algorithm is to create Γ_k from the two sets. This is done by selecting all possible combinations of X_1 and X_2 where the ending state of X_1 and the starting state of X_2 are the same. With the two sets sorted according to the states $S_{t+u+v/2}$, the set Γ_k is easily obtained. Since the size of Γ_k is about 2^{u+v} this does not change the asymptotic complexity but the constant term in the complexity is very small. Each operation consists of just concatenating X_1 and X_2, a very simple operation.

The memory requirement in the algorithm is moderately small. We need about 2^{u+v} words, where each word represents an X vector.

7 Simulation Results

In order to verify the attack, it has been simulated on a reduced version of Edon80. We have produced a keystream exactly as in Edon80 with the modification that only 24 e-transformers was used, i.e., a variant logically denoted Edon24. We have investigated the case when the assumed period P'_B is such that $P_B | P'_B$. The simulations target the number of possible values for the vector X that are still possible after intersecting the k' sets Γ_k, $k = 0, 1, \ldots, k'$. Table 1 shows the average number of remaining elements for different values of k' when $v = u + 2$, i.e., when $d = 2$. As stated in Section 5.3 we expect that we need about v intersections of sets Γ_k. For all simulated values of u we have in average only 0.1 possible value for the X vector left in Γ_k after $v = u + 2$ intersections. This verifies our claim. Table 2 shows the average number of remaining elements when $d = 6$. As expected, the intersections produce an empty set with much fewer sets Γ_k than in the case with $d = 2$.

Moreover, our implementation also always found the correct key and discarded all false key candidates using our algorithm.

8 Estimating the Attack Complexity

As explained before, we have several parameters that we can choose, giving different parameters for the attack. Basically, there is a trade-off between the

Table 1. The average number of possible values for X left in the intersection of $\Gamma_k, k = 0, 1, \ldots, k'$ sets for different choice of u, when $d = 2$.

	$\lvert Y\rvert = u$					
k'	4	5	6	7	8	9
0	909.3	3597.7	14534.2	57953.3	232281.4	927796.6
1	201.6	788.9	3226.0	12823.0	51486.9	205105.3
2	45.8	172.3	716.5	2837.0	11407.7	45379.9
3	10.1	37.7	159.0	626.2	2526.2	10033.2
4	2.3	8.3	35.2	138.4	558.9	2223.5
5	0.5	1.9	7.8	30.6	124.2	493.0
6	0.1	0.4	1.7	6.8	27.7	109.2
7	0.0	0.1	0.4	1.5	6.1	23.9
8	0.0	0.0	0.1	0.3	1.3	5.4
9	0.0	0.0	0.0	0.1	0.3	1.1
10	0.0	0.0	0.0	0.0	0.1	0.2
11	0.0	0.0	0.0	0.0	0.0	0.1
12	0.0	0.0	0.0	0.0	0.0	0.0

required length of the received keystream and computational complexity of the key recovery part. For example, choose $d = 2$ and $u = 9$ as simulated above,

Table 2. The average number of possible values for X left in the intersection of $\Gamma_k, k = 0, 1, \ldots, k'$ sets for different choice of u, when $d = 6$.

	$\lvert Y\rvert = u$			
k'	4	5	6	7
0	16265.1	64310.8	260222.9	1040318.8
1	253.0	983.8	4036.7	16164.6
2	3.8	15.2	62.9	250.0
3	0.1	0.2	0.9	4.1
4	0.0	0.0	0.0	0.1
5	0.0	0.0	0.0	0.0

i.e. $B = 70$ in the Edon80 case, and an assumed period of $P'_B = 2^{40} \cdot 3^{24}$. Then the computational complexity is low, roughly 2^{44} but the required keystream is large, roughly $2^{78} \cdot 11$, where the factor 11 comes from the fact that we need to intersect at most $11 + 1$ sets Γ_k. With the low computational complexity we can of course increase the d parameter and reduce the required keystream to roughly 2^{78}. Finally, we must include the error probability. An error occurs if P'_B is not a multiple of the true period P_B. We simply use (3) to estimate this probability. A numerical calculation gives that the period is below 2^{78} with probability more than $1/2$. There may be some possible periods below 2^{78} which does not divide

P'_B. On the other hand, we can try out different (the most probable) forms of P'_B in our attack with only a slight increase in complexity. So here we can assume that the error probability is about $1 - \alpha_{P'_B} \approx 1/2$.

Clearly, such a long received keystream sequence as 2^{78} is not desirable, even if the computational complexity is low. We also see that allowing the error probability to be quite close to 1 might be beneficial. We will then repeat the attack $\alpha_{P'_B}^{-1}$ times and the requirement is now to receive $\alpha_{P'_B}^{-1}$ different keystreams (obtained from different IV values). The computational complexity, T, grows to

$$T = \alpha_{P'_B}^{-1} \cdot 2^{4u+d+3} \cdot \frac{u+d}{d}.$$

Though in average we only need slightly less than K intersections, there will be key candidates that need more intersections before they can be discarded. On the other hand, it is not crucial that *all* wrong key candidates are discarded. If we end up with a set up possible keys then these keys can be tested individually at the end. This will not affect the computational complexity. With $P'_B \cdot K$ keystream bits, we will have $K+1$ sets Γ_k, $k = 0, 1, \ldots, K$ to intersect. This will keep the probability of false alarm low. Thus, the number of keystream bits, D_{IV}, that are needed from each IV is given by

$$D_{IV} = P'_B \cdot \frac{2u+2d}{d}.$$

The total number of keystream bits, D_{tot}, is given by

$$D_{tot} = \alpha_{P'_B}^{-1} \cdot P'_B \cdot \frac{2u+2d}{d}.$$

The trade-off parameters in the attack are u, d and P'_B. The attack complexities are all functions of these values. We consider two cases.

I There is no restriction on the amount keystream that can be generated by one key/IV pair.
II We respect the limitation given in [7], i.e., only 2^{48} keystream bits can be generated before reinitialization with a new IV.

In Table 3 we tabulate some possible values of T, D_{IV} and D_{tot} for the two different cases. With no restriction on the keystream per key/IV pair the parameter choice $u = 13$, $d = 4$ and $P'_B = 2^{58}$ gives about 2^{69} for both computational complexity and total amount of keystream. We conclude that we have an attack requiring a total of 2^{69} received keystream bits and 2^{69} simple operations to recover the key.

If we respect the 2^{48} limit, choosing parameters $u = 9$, $d = 6$ and $P'_B = 2^{45}$ will allow us to recover the key with in total $2^{72.4}$ keystream bits and $2^{71.4}$ simple operations. In many situations it is difficult to argue that we can have a computational complexity that is lower than the number of keystream bits. An adversary observing the keystream is likely to need at least one operation per observed keystream bit. On the other hand, only very few keystream bits

Table 3. Attack complexity for various parameter choices.

Case	u	d	P'_B	$\alpha_{P'_B}$	D_{IV}	D_{tot}	T
I	9	6	2^{60}	$2^{-9.18}$	$2^{62.3}$	$2^{71.5}$	$2^{55.5}$
	13	2	2^{54}	$2^{-10.9}$	$2^{57.9}$	$2^{68.8}$	$2^{70.8}$
	13	4	2^{58}	$2^{-7.72}$	$2^{61.1}$	$2^{68.8}$	$2^{68.8}$
	15	2	2^{56}	$2^{-7.73}$	$2^{60.1}$	$2^{67.8}$	$2^{75.8}$
II	7	10	2^{46}	$2^{-26.1}$	$2^{47.8}$	$2^{73.9}$	$2^{67.9}$
	9	6	2^{45}	$2^{-25.1}$	$2^{47.3}$	$2^{72.4}$	$2^{71.4}$
	9	8	2^{45}	$2^{-25.1}$	$2^{47.1}$	$2^{72.2}$	$2^{73.2}$
	11	4	2^{45}	$2^{-22.7}$	$2^{47.9}$	$2^{70.6}$	$2^{75.6}$

are actually used in the attack. If the adversary can randomly access keystream bits, the computational complexity can be allowed to be much smaller than the keystream.

Comparing the attack to an exhaustive key search, we can note that an exhaustive key search would require computing the key/IV setup consisting of 160 cycles and then additionally 80 cycles to get the 80 first output bits. Every cycle must compute 80 quasigroup operations. So a software implementation would require $240 \cdot 80$ quasigroup operations, i.e., more than 2^{14} operations to test one key. Thus, our attack requiring roughly 2^{69} simple operations is about 2^{25} times faster than a software implemented exhaustive key search.

9 Conclusion

An attack on Edon80 has been presented. It takes advantage of the relatively short period inside the state of the cipher. By knowing that some values in the internal state will repeat with high probability after a certain amount of state updates, it was possible to determine several key bits used in the update of the last part of the state. The required number of keystream bits as well as the total complexity is around 2^{69}, if we allow each key/IV pair to generate about 2^{61} keystream bits. If we consider the restriction put by the designers i.e., only 2^{48} keystream bits can be produced by each key/IV pair, then the total complexity is about 2^{72} simple operations with about 2^{47} bits from each key/IV pair.

Adding just a few more quasigroup operations to the chain of 80 is not enough to counter the attack, but doubling this number to 160 would be sufficient to resist the attack. However, such a modification comes at the cost of doubling the hardware (and the gate count).

We do not exclude the possibility of improving this attack by for example finding more efficient ways of computing the intersection of Γ_k sets. Since we are guessing a lot of key bits, there might be a possibility to do something more

efficient. Some minor improvements to the described attack have already been found, and will be described in the full version of this paper.

References

1. Gligoroski, D., Markovski, S., Kocarev, L., Gusev, M.: Edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/007 (2005) *http://www.ecrypt.eu.org/stream.*
2. Hong, J.: Period of streamcipher Edon80. In Maitra, S., Madhavan, C.E.V., Venkatesan, R., eds.: Progress in Cryptology - INDOCRYPT 2005. Volume 3797/2005 of Lecture Notes in Computer Science., Springer-Verlag (2005) 23–34
3. Gligoroski, D., Markovski, S., Kocarev, L., Gusev, M.: Understanding periods in edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/054 (2005) *http://www.ecrypt.eu.org/stream.*
4. Gligoroski, D., Markovski, S., Knapskog, S.J.: On periods of Edon-(2m, 2k) family of stream ciphers. The State of the Art of Stream Ciphers, Workshop Record, SASC 2006, Leuven, Belgium (2006)
5. Kasper, M., Kumar, S., Lemke-Rust, K., Paar, C.: A note on algebraic properties of quasigroups in Edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/032 (2007) *http://www.ecrypt.eu.org/stream.*
6. Kasper, M., Kumar, S., Lemke-Rust, K., Paar, C.: A compact implementation of Edon80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/057 (2006) *http://www.ecrypt.eu.org/stream.*
7. Gligoroski, D., Markovski, S., Kocarev, L., Gusev, M.: Status of Edon80 in the second phase of eSTREAM. eSTREAM, ECRYPT Stream Cipher Project (2006) *http://www.ecrypt.eu.org/stream/p2ciphers/edon80/edon80_p2note.pdf.*