Entropy of the Internal State of an FCSR in Galois Representation

Andrea Röck

Team SECRET, INRIA Paris-Rocquencourt, France
http://www-rocq.inria.fr/secret/
andrea.roeck@inria.fr

Abstract. Feedback with Carry Shift Registers (FCSRs) are primitives that are used in multiple areas like cryptography or generation of pseudorandom sequences. In both cases, we do not want that an attacker can easily guess the content of the register. This requires a high entropy of the inner state. We consider the case of a binary FCSR in Galois representation. In this article, we show that we already lose after one iteration a lot of entropy. The entropy reduces until the moment where the FCSR reaches a periodic behavior. We present an algorithm which computes the final entropy of an FCSR and which also allows us to show that the entropy never decreases under the size of the main register.

1 Introduction

FCSRs are finite state machines which were independently introduced by Goresky and Klapper [KG93,KG97], Marsaglia and Zamand [MZ91], and Couture and L'Ecuyer [CL94]. They are similar to Linear Feedback Shift Registers (LFSRs). However, they use an additional register to store the carry information and their transition function is non linear, more precisely quadratic [AB05b]. Goresky and Klapper [GK02] distinguish between FCSRs in Fibonacci and Galois representation. In this article, we consider binary FCSRs in Galois architecture.

An application of FCSRs is cryptography, as for example the stream cipher presented by Arnaut and Berger in [AB05b]. In this area, the entropy is an important parameter. It represents the minimal number of binary questions an attacker has to ask on average to obtain an unknown value. Therefore, we are always interested in a high entropy. In the following, we study the entropy of the inner state of an FCSR. First we give some notations. Subsequently, we will present the structure of an FCSR in the Galois representation and explain exactly the meaning of the state entropy. In the end, we give an overview of this article.

1.1 Notations

In this article, we are going to use the following notations to describe the behavior of an FCSR.

n: Let n denote the size of the main register M in bits.

m: We mean by m the 2-adic description of the current state of M: $m = \sum_{i=0}^{n-1} m_i\, 2^i$, where $m_0, \ldots, m_{n-1}$ are the bits in M. Thus, $0 \leq m < 2^n$.

d: Let d be an integer with $2^{n-1} \leq d < 2^n$. We will use it to determine at which positions we have a feedback with carry. We mean by d_i the i'th bit of the binary representation of d. We define $d_i = 0$ for all $i < 0$.

$I_d = \{0 \leq i \leq n-2 | d_i = 1\}$: The set I_d contains all feedback positions.

$d^* = d - 2^{n-1}$.

$\ell = wt(d^*)$: We denote with ℓ the number of feedback branches, where $wt(d^*)$ means the hamming weight of d^*, *i.e.* the number of 1's in its binary representation.

$c = \sum_{i \in I_d} c_i\, 2^i$: The value c is the 2-adic description of the carry bits.

$(m(t), c(t))$: The pair $(m(t), c(t))$ represents the actual value of the state after t iterations.

$q = 1 - 2d$: We denote by q the divisor in the 2-adic description of the produced bit string. It holds that $q < 0$.

$p = m + 2c$: The value p represents the corresponding dividend which can be in the range of $0 \leq p \leq |q|$.

s: Let s is the binary representation of the output sequence of the generator.

$H(1)$: Let $H(1)$ denote the entropy of the state after one iteration.

H^f: As soon as the FCSR obtains a periodic behavior, the state entropy does not change any more. We denote this final entropy by H^f.

Furthermore, we need two special sums in our calculations:

$$S_1(k) = \sum_{x=2^{k-1}+1}^{2^k} x \log_2(x),$$

$$S_2(k) = \sum_{x=1}^{2^k-1} x \log_2(x).$$

1.2 FCSR in Galois

A binary FCSR in Galois representation is built like a LFSR with a main register and several feedback branches. However, in the case of the FCSRs we have an additional carry bit at each feedback position. An example for such an FCSR is presented in Fig. 1.

To change the state from $(m(t), c(t))$ to $(m(t+1), c(t+1))$, we use the following equations:

$$
\begin{aligned}
m_{n-1}(t+1) &= m_0(t), && (1)\\
i \in I_d : m_i(t+1) &= (m_0(t) + c_i(t) + m_{i+1}(t)) \bmod 2, && (2)\\
c_i(t+1) &= (m_0(t) + c_i(t) + m_{i+1}(t)) \div 2, && (3)\\
i \notin I_d : m_i(t+1) &= m_{i+1}(t), && (4)
\end{aligned}
$$

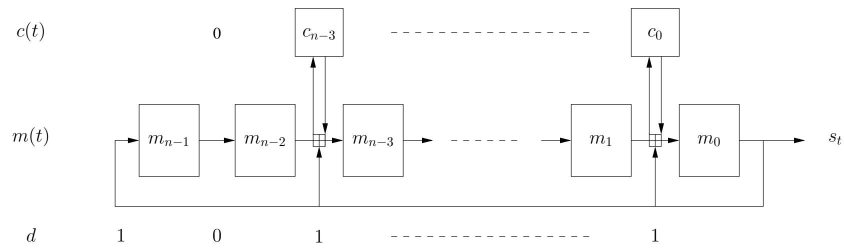

Fig. 1. Model of an FCSR.

where $x \div 2$ means the integer division $\lfloor x/2 \rfloor$. The bit $m_0(t)$ is directly shifted to position $m_{n-1}(t+1)$. In the cases without a feedback, *i.e.* $i \notin I_d$, the bit gets simply shifted one position to the left. Otherwise, we add the bit in the main register at position $i+1$ with the carry bit and the bit from position 0. This sum modulo 2 is put into the main register. The value of the sum divided by 2 is given back to the carry bit. The equations (2) and (3) can also be written together as:

$$m_i(t+1) + 2c_i(t+1) = m_0(t) + c_i(t) + m_{i+1}(t). \tag{5}$$

Remark 1. The + in all these equations represents the normal integer addition (and not an addition modulo 2) even if we are only using bit values for $m_i(t)$ and $c_i(t)$.

The output sequence of the FCSR can be easily described by means of the feedback positions (determined by d) and the initial state (m, c). Let q and p be defined as $q = 1 - 2d$ and $p = m + 2c$. This means that $q < 0$ and $0 \leq p \leq |q|$. In this case, the output sequence of the FCSR is $s = \frac{p}{q}$ [GK02]. Another property shown in [AB05a] is: Let $(m(t+1), c(t+1))$ be the state produced by $(m(t), c(t))$ after one iteration. Let $p(t) = m(t) + 2c(t)$ and $p(t+1) = m(t+1) + 2c(t+1)$ be the corresponding values of p. Then it holds that:

$$2p(t+1) = p(t) \pmod{q}. \tag{6}$$

where all bits are 0 or 1 respectively.

We can use the following property of an FCSR to determine the period of a sequence: if q is odd, p and q are coprime and $s = p/q$ then the period of s is the order of 2 modulo q, this means the smallest t such that $2^t = 1 \pmod{q}$. If p and q are not coprime, we divide them both by their greatest common divisor. It is easy to see that the maximal period is $|q| - 1$. There are always two fix points $(0, 0)$ and $(2^n - 1, d^*)$ with an period of length 1, which represents the cases where all bits are respectively 0 or 1. It is known [GK02] that if 2 is primitive modulo q, which means it has order $|q| - 1$, then all the other periodic states are obtained by the state $(1, 0)$ by iterating the FCSR. Since q is odd, this implies that, except for the two fix points, we have an FCSR with one single cycle of maximal length $|q| - 1$. Such a sequence is called ℓ–sequence.

1.3 State Entropy

For any discrete probability distribution $P = \{p_1, \ldots, p_Z\}$, Shannon's entropy is defined by:

$$H = \sum_{j=1}^{Z} p_j \log_2 \left(\frac{1}{p_j} \right) . \qquad (7)$$

If $p_j = 0$, we use the classical convention in the computation of the entropy: $0 \log_2(\frac{1}{0}) = 0$. This can be done, since a zero probability has no influence in the computation of the entropy. The state of our FCSR consists of $n + \ell$ bits. Let us assume that each initial state is chosen with the same probability $p_{(m(0),c(0))} = 2^{-n-\ell}$. Then by using (7) we know that the initial entropy is $n + \ell$ bits.

Let us consider the update function of an FCSR. It consists of trees of different length, where each root of a tree is a node in a cycle. This is the same for any finite function which is not a permutation. An example of such a graph can be found in Fig. 2. Each node in the graph represents a possible state of

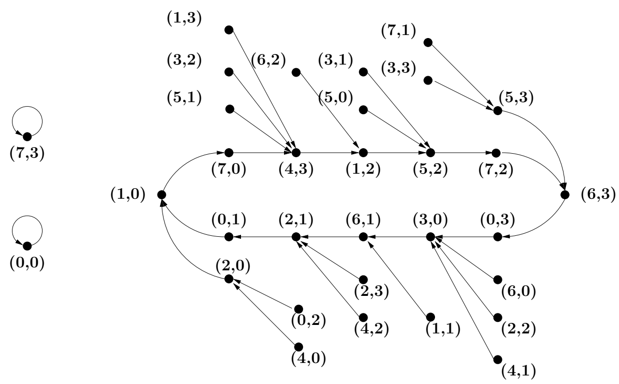

Fig. 2. Functional graph of the FCSR with $n = 3$ and $q = -13$.

the FCSR and each arrow represents an update from one state to another. Each time when a state is produced by more than one state, the number of possible states, and therefore the entropy of the state, reduces. As soon as we reached a point on the cycle from any possible starting points, the FCSR behaves like a permutation and the entropy stays constant. We will denote this value by the final entropy H^f.

We need the probability of each state to compute the entropy after some iterations of the update function. If a state is produced by exactly r other states after k iterations, then its probability is $r\ 2^{-n-\ell}$. Using this probability, we can compute the entropy applying (7).

Remark 2. We consider the case where all the $2^{n+\ell}$ different values for the initial state can be chosen, like in the first version of the F-FCSR-8 [ABL05]. This is not always the case, *e.g.* for later versions of the F-FCSR [ABL06], the carry bits of the initial value are always set to 0. This implies that at the beginning this stream cipher has only n bits of entropy, however, it will not lose any more entropy, as we will see later.

1.4 Outline

In Section 2, we compute the state entropy after 1 iteration. Subsequently, in Section 3, we present an algorithm which computes the final entropy for any arbitrary FCSR. This algorithm uses some sums which gets difficult to compute for large ℓ. However, we give a method to compute very close upper and lower bounds of the entropy. The same algorithm is used in Section 4 to prove that the final entropy is always larger than n. We conclude the article in Section 5.

2 Entropy after one Iteration

If the initial value of the state is chosen uniformly, we have an initial state entropy of $n + \ell$ bits. We are interested in how many bits of entropy we already lose after one iteration.

Let us take an arbitrary initial state $(m(0), c(0))$ which produces the state $(m(1), c(1))$ after one iteration. To compute the probability of $(m(1), c(1))$, we want to know by how many other initial states $(m'(0), c'(0)) \neq (m(0), c(0))$ it is produced. From (1), we see that for such an initial state $m'_0(0) = m_{n-1}(1) = m_0(0)$ is fixed. In the same way, we see from (4) that for all $i \notin I_d$ the values $m'_{i+1}(0) = m_i(1) = m_{i+1}(0)$ are already determined. By using (5) and the previous remarks, we can write for $i \in I_d$:

$$m_i(1) + 2c_i(1) = m_0(0) + c'_i(0) + m'_{i+1}(0)$$
$$m_i(1) + 2c_i(1) = m_0(0) + c_i(0) + m_{i+1}(0)$$

and thus,

$$c'_i(0) + m'_{i+1}(0) = c_i(0) + m_{i+1}(0).$$

If $m_{i+1}(0) = c_i(0)$ it must hold that $m_{i+1}(0) = c'_i(0) = m'_{i+1}(0)$ since each value can only be either 0 or 1. Therefore, the only possibility for $(m'(0), c'(0))$ to differ from $(m(0), c(0))$ is that there is a position $i \in I_d$ with $m_{i+1}(0) \neq c_i(0)$.

Let j be the number of positions in the initial state where $i \in I_d$ and $c_i(0) + m_{i+1}(0) = 1$. Then, there are exactly $2^j - 1$ other initial states which produce the same state after one iteration. Thus, $(m(1), c(1))$ has a probability of $\frac{2^j}{2^{n+\ell}}$. We look now how many states $(m(1), c(1))$ have this probability. Such a state must be created by an initial state $(m(0), c(0))$ which has j positions $i \in I_d$ with $c_i(0) + m_{i+1}(0) = 1$. There are $\binom{\ell}{j}$ possibilities to choose these positions. At the remaining $\ell - j$ positions with $i \in I_d$, we have $m_{i+1}(0) = c_i(0) \in \{0, 1\}$. In the same way, we can choose between 0 and 1 for the remaining $n-\ell$ positions. There exists exactly $2^{n-j}\binom{\ell}{j}$ different states $(m(1), c(1))$ with a probability of $2^{j-n-\ell}$. Using (7), $\sum_{j=0}^{\ell}\binom{\ell}{j} = 2^\ell$ and $\sum_{j=0}^{\ell} j\binom{\ell}{j} = \ell 2^{\ell-1}$ we can write the entropy after one iterations as:

$$\begin{aligned} H(1) &= \sum_{j=0}^{\ell} 2^{n-j}\binom{\ell}{j} 2^{j-n-\ell}(n+\ell-j) \\ &= n + \frac{\ell}{2}. \end{aligned}$$

We have shown that the entropy after one iteration is:

$$H(1) = n + \frac{\ell}{2} \tag{8}$$

which is already $\ell/2$ bits smaller than the initial entropy.

3 Final State Entropy

We have shown that after one iteration the entropy has already decreased by $\ell/2$ bits. We are now interested in down to which value the entropy decreases after several iterations, *i.e.* the final entropy H^f. For the computation of H^f, we need to know how many initial states arrive at the same cycle point after the same number of iterations. We are going to use the following proposition.

Proposition 1. *[ABM08, Prop. 5] Two states (m, c) and (m', c') are equivalent,* i.e. *$m + 2c = m' + 2c' = p$, if and only if they eventually converge to the same state after the same number of iterations.*

Let us assume that we have iterated the FSCR sufficiently many times that we are on the cycle of the functional graph. In this case, we do not have any more collisions. If a state in the cycle is reached by x other states, it has a probability of $x/2^{n+\ell}$. After one iteration, all probabilities shift one position in the direction of the cycle, which corresponds to a permutation of the probabilities. However, the definition of the entropy is invariant to such a permutation. Let $v(p)$ denote the number of states which produce the same p. From Proposition 1, we know that we find a corresponding state in the cycle, which is reached by $v(p)$ states and has a probability of $v(p)/2^{n+\ell}$. We can write the final entropy by means of equation (7):

$$H^f = \sum_{p=0}^{|q|} \frac{v(p)}{2^{n+\ell}} \log_2\left(\frac{2^{n+\ell}}{v(p)}\right). \tag{9}$$

Remark 3. Let us have a look at an FCSR as mentioned in Remark 2, which always sets $c(0) = 0$. For each $0 \le p < 2^n$ there is only one possibility to form $p = m$. This means that there are no collision and the final entropy is the same as the initial entropy, namely n bits.

The numerator p can take any value between 0 and $2^n - 1 + 2(d - 2^{n-1}) = 2d - 1 = |q|$. We look at the binary representations of $m, 2c$ and p to find all possible pairs (m, c) which correspond to a given p. We study bit per bit which values are possible. This idea is presented in Fig. 3, where each box represents a

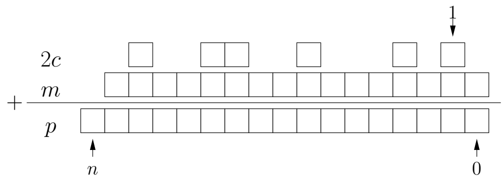

Fig. 3. Evaluation of $p = m + 2c$ bit per bit.

possible position of a bit. We mean by $2c$ the carry bits of the value c all shifted one position to the left.

Remark 4. Due to this shift, we will normally consider the index $i-1$ for d or c.

3.1 Notations

Before continuing we give some additional notations:

$ca(j) = m_j + c_{j-1} + ca(j-1) \pmod 2$: In the following, we consider only the addition $m+2c$, thus, if we talk of a carry we mean the carry of this integer addition. We mean by $ca(j)$ the carry which is produced by adding the bits m_j, c_{j-1} and the carry of the previous position. *E.g.* if we have $m = 13$ and $c = 2$ we have $ca(1) = 0$ and $ca(2) = ca(3) = 1$. The value $ca(0)$ is always 0 since we only have m_0 for the sum.

$i := \lfloor \log_2(p) \rfloor$: For $1 \leq p \leq |q|$, let i be the index of the most significant bit in p which is not equal to 0.

$\ell' = \#\{j \leq i | d_{j-1} = 1\}$: We define by ℓ' the number of indices smaller or equal to i for which $d_{j-1} = 1$.

$r(p) = max\{j < i | d_{j-1} = 0, p_j = 1\}$: For a given p, let $r(p)$ be the highest index smaller than i such that $d_{j-1} = 0$ and $p_j = 1$. In this case, the carry of the integer addition $m + 2c$ cannot be forwarded over the index $r(p)$. If there is no index j with $d_{j-1} = 0$ and $p_j = 1$, we set $r(p) = -1$. For the case $i < n$ and $d_{i-1} = 0$, we get a range of $-1 \leq r(p) < i$. However, we will see that for $2^n \leq p \leq |q|$, the value $r(p)$ is only possible for $-1 \leq r(p) < log_2(d^*) + 1$. For simplicity reasons, we sometimes write only r if it is clear which p we are meaning or if there are multiple p's with the same value for $r(p)$.

$f_1(r)$: This is a helping function which is needed in the further computations. It is defined as:

$$f_1(r) = \begin{cases} 2^r & \text{for } r \geq 0 \\ 1 & \text{for } r = -1. \end{cases}$$

$\ell''(r) = \#\{j < r | d_{j-1} = 1\}$: For a given r, we define ℓ'' as the number of indices strictly smaller than r for which $d_{j-1} = 1$. Again, we use sometimes only ℓ'' if it is clear to which r we refer.

$v(p) = \#\{(m, c) | m + 2c = p\}$: Let $v(p)$ denote the number of pairs (m, c) which create $p = m + 2c$.

In Case 2 in Section 3.2, we will see that it is sufficient to consider the indices j with $r(p) < j < i$ for knowing if there is a carry at position $i - 1$. To facilitate the computation, we use the following notations:

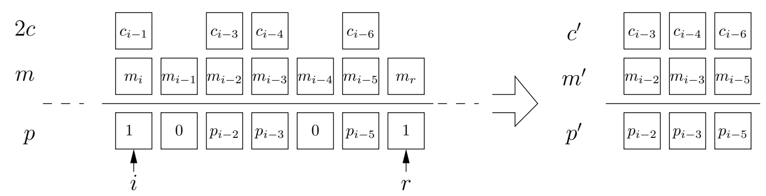

Fig. 4. Reduction of $p, m, 2c$ to p', m', c'.

p',m' and c': We mean with p', m' and c' the bit strings $p, m, 2c$ reduced to the positions j with $r < j < i$ and $d_{j-1} = 1$. An example can be seen in Fig. 4.

Remark 5. In the case of c', we do not consider c but $2c$. Let j_1 be an index $r < j_1 < i$ with $d_{j_1-1} = 1$ and j_2 its corresponding index in the reduction, *i.e.* $m'_{j_2} = m_{j_1}$. Then we use $c'_{j_2} = c_{j_1-1}$. Furthermore, the value of c' is a continuous bit string, since we are only interested in positions where there is a feedback register.

The length of p', m' and c' is $\ell' - \ell'' - 1$ bits.

$0p'$ and $1p'$: We obtain $0p'$ and $1p'$ by concatenating respectively 0 and 1 to the left of the bit string p'.

$X(p')$: We denote by $X(p')$ the number of possibilities for m' and c' such that $1p' = m' + c'$, which means that we have a carry at the position of the most significant bit of m'.

3.2 Final Entropy Case by Case

We cannot write the sum (9) for the final entropy directly in a closed form. However, we partition the set of all possible $0 \leq p \leq |q|$ in four cases. For each case, we will evaluate the value $\frac{v(p)}{2^{n+\ell}} \log_2 \left(\frac{2^{n+\ell}}{v(p)} \right)$ for all its p's. We obtain the final sum of the entropy by summing up all these values.

Case 1: $1 \leq i < n$ and $d_{i-1} = 0$

To create $p_i = 1$ at position i, we have to add $m_i + ca(i-1)$. For each value of $ca(i-1)$ there exists exactly one possibility for m_i. For each position j with a feedback bit, $d_{j-1}=1$, we have two possibilities to create the value of p_j. In this case, we can write for each p:

$$v(p) = 2^{\ell'}.$$

For each i, all p's within the range $[2^i, 2^{i+1}[$ are possible. So we must add:

$$H_1(n,i,\ell,\ell') = 2^i\, 2^{\ell'-n-\ell}(n+\ell-\ell') \tag{10}$$

to the entropy for each $1 \leq i \leq n$ with $d_{i-1}=0$.

Case 2: $1 \leq i < n$ and $d_{i-1}=1$:

For a given p we know from the definition of $r(p)$ that for all j's with $r(p) < j < i$, if $d_{j-1}=0$, then $p_j=0$. In the case of $(d_{j-1}=0,\ p_j=0)$, a carry is always forwarded. This means that for $m+2c$, if we have $ca(j-1)=1$, m_j must be 1 and we have $ca(j)=1$. However, with $ca(j-1)=0$ we have $m_j=0$, and so we have $ca(j)=0$ as well. It is sufficient to consider the $\ell'-\ell''-1$ positions j with $i>j>r(p)$ for which $d_{j-1}=1$, to know if we have a carry at index $i-1$.

From the definition of this case, we know that $p_i=1$ and $d_{i-1}=1$. If we have $ca(i-1)=0$ we have two possibilities for (m_i,d_{i-1}), namely $(1,0)$ and $(0,1)$, to generate the $p_i=1$. Otherwise, we only have one possibility $(0,0)$. For a given p' we have:

$$X(p') + 2(2^{\ell'-\ell''-1} - X(p')) = 2^{\ell'-\ell''} - X(p') \tag{11}$$

possibilities to choose m',c' and (m_i,d_{i-1}). The following lemma helps us to compute $X(p')$. Its proof is given in Appendix B.

Lemma 1. *Let p', m' and c' be three bit strings of length K. For all $0 \leq x \leq 2^K-1$, there exists* exactly one *p' with $X(p')=x$,* i.e. *there exists exactly one p' such that for x different pairs (m',c') we can write $1p' = m'+c'$.*

In our case: $K=\ell'-\ell''-1$. The next question is, how many p's have the same reduction p'. If $0 \leq r < i$ we have 2^r possibilities, in the case $r=-1$ we have only one. We consider this behavior by using the helping function $f_1(r)$.

By combining Lemma 1 and (11), we obtain that for each $2^{\ell'-\ell''-1}+1 \leq y \leq 2^{\ell'-\ell''}$ there is exactly one p' which is generated by y different combinations of m' and c'. Each p which corresponds to such a p', is generated by $y\ 2^{\ell''}$ different pairs (m,c), since at each position $j<r$ with $d_{j-1}=1$ we have two possibilities to create p. This means that this p has a probability of $\frac{y\,2^{\ell''}}{2^{n+\ell}}$.

For fixed values of i, r, ℓ' and ℓ'' we have to add the following value to the entropy:

$$\begin{aligned}
&H_2(n,r,\ell,\ell',\ell'') \\
&= f_1(r) \sum_{y=2^{\ell'-\ell''-1}+1}^{2^{\ell'-\ell''}} \frac{y 2^{\ell''}}{2^{n+\ell}} \log_2\left(\frac{2^{n+\ell}}{y 2^{\ell''}}\right) \\
&= f_1(r) 2^{-n-\ell}\left[2^{\ell'-2}\left(3\, 2^{\ell'-\ell''-1}+1\right)(n+\ell-\ell'') - 2^{\ell''} S_1(\ell'-\ell'')\right].
\end{aligned}$$

Thus, in this case, we have to add for every $1 \leq i \leq n-1$ with $d_{i-1} = 1$, and every $-1 \leq r < i$ with $d_{i-1} = 0$ the value:

$$H_2(n,r,\ell,\ell',\ell'') = f_1(r)2^{-n-\ell}\Big[2^{\ell'-2}\Big(3\,2^{\ell'-\ell''-1} + 1\Big)(n+\ell-\ell'') - 2^{\ell''}S_1(\ell'-\ell'')\Big] \tag{12}$$

where $\ell' = \ell'(i)$ and $\ell'' = \ell''(r)$.

Case 3: $i = n$, $2^n \leq p \leq |q|$:

In this case, we always need a carry $ca(n-1)$ to create $p_n = 1$. Like in the previous case, we are going to use $r(p)$, ℓ'', p', (m', c') and $X(p')$. However, this time we have $i = n$ and $\ell = \ell'$.

For $p = |q|$, which means that (m,c) consists of only 1's, it holds that for all $n > j > \log_2(d^*)+1$, we have $p_j = 0$. If we would have a $r(p) \geq \log_2(d^*)+1$, then p would be greater than $|q|$ which is not allowed. Therefore, r must be in the range of $-1 \leq r < \log_2(d^*)+1$.

From Lemma 1, we know that for all $1 \leq x \leq 2^{\ell-\ell''} - 1$ there exists exactly one p' with $X(p') = x$, *i.e.* there are x pairs of (m', c') with $1p' = m' + c'$. We exclude the case $X(p') = 0$, because we are only interested in p's that are able to create a carry. For each p', there are $f_1(r)$ possible values of p which are reduced to p'. If p' is created by x different pairs of (m', c'), then each of its corresponding values of p is created by $x\ 2^{\ell''}$ pairs of (m,c) and has a probability of $\frac{x\ 2^{\ell''}}{2^{n+\ell}}$.

For a given r and ℓ'' the corresponding summand of the entropy is:

$$\begin{aligned} H_3(n,r,\ell,\ell'') &= f_1(r) \sum_{x=1}^{2^{\ell-\ell''}-1} \frac{x\ 2^{\ell''}}{2^{n+\ell}} \log_2\left(\frac{2^{n+\ell}}{x\ 2^{\ell''}}\right) \\ &= f_1(r)2^{-n}\left[2^{-1}\left(2^{\ell-\ell''}-1\right)(n+\ell-\ell'') - 2^{\ell''-\ell}S_2(\ell-\ell'')\right]. \end{aligned}$$

In this case, we have to add for each value of $-1 \leq r < \log_2(d^*)+1$ with $d_{r-1} = 0$ and $\ell'' = \ell''(r)$:

$$H_3(n,r,\ell,\ell'') = f_1(r)2^{-n}\left[2^{-1}\left(2^{\ell-\ell''}-1\right)(n+\ell-\ell'') - 2^{\ell''-\ell}S_2(\ell-\ell'')\right]. \tag{13}$$

Case 4: $0 \leq p \leq 1$

If $p = 0$ or $p = 1$, there exists only one pair (m,c) which can produce the corresponding p. Thus, for each of these p's we have to add:

$$H_4(n,\ell) = 2^{-n-\ell}(n+\ell) \tag{14}$$

to the sum of the entropy.

The Algorithm 1 shows how we can compute the final entropy for an FCSR defined by n and d using the summands of the individual cases.

Algorithm 1 Final entropy

$H^f \leftarrow 0$
$\ell' \leftarrow 0$
$\ell \leftarrow wt(d) - 1$
$H^f \leftarrow H^f + 2H_4(n, \ell)$ $\{p = 0$ and $p = 1\}$
for $i = 1$ to $n - 1$ **do**
  if $d_{i-1} = 0$ **then**
    $H^f \leftarrow H^f + H_1(n, i, \ell, \ell')$
  else $\{d_{i-1} = 1\}$
    $\ell' \leftarrow \ell' + 1$
    $\ell'' \leftarrow 0$
    for $r = -1$ to $i - 1$ **do**
      if $d_{r-1} = 0$ **then**
        $H^f \leftarrow H^f + H_2(n, r, \ell, \ell', \ell'')$
      else $\{d_{r-1} = 1\}$
        $\ell'' \leftarrow \ell'' + 1$
      end if
    end for
  end if
end for
$\ell'' \leftarrow 0$
for $r = -1$ to $log_2(d - 2^{n-1})$ **do** $\{2^n \leq p \leq 2d - 1\}$
  if $d_{r-1} = 0$ **then**
    $H^f \leftarrow H^f + H_3(n, r, \ell, \ell'')$
  else $\{d_{r-1} = 1\}$
    $\ell'' \leftarrow \ell'' + 1$
  end if
end for

3.3 Complexity of the Computation

The exact computation of the entropy requires to evaluate the sums $S_1(k) = \sum_{x=1}^{2^k-1} x \log_2(x)$ and $S_2(k) = \sum_{x=2^{k-1}+1}^{2^k} x \log_2(x)$. If we have stored the values $S_1(k)$ and $S_1(k)$ for $1 \leq k \leq \ell$, we are able to compute the final entropy in $O(n^2)$. We need $O\left(2^\ell\right)$ steps to evaluate both sums, which is impractical for large ℓ. However, by using the bounds (20)-(21) for larger k's, we can easily compute a lower and upper bound of those sums. In Table 1, we compare for different FCSRs the exact computation with those, using upper and lower bounds. The values of d were randomly chosen. However, the accuracy of the estimation of the sums can be shown anyway.

We mean by H^f the exact computation of the the final entropy. The values lb/ub H_f mean the lower and the upper bound obtained by using the approximation of S_1 and S_2. The last two columns, lb/ub H_f, $k > 5$, we gain by using the approximations only for $k > 5$. This last approximation is as close that we do not see any difference between the lower and the upper bound in the first 8 decimal places.

n	d	ℓ	H^f	lb H^f	ub H^f	lb H^f , $k > 5$	ub H^f, $k > 5$
8	0xAE	4	8.3039849	8.283642	8.3146356	8.3039849	8.3039849
16	0xA45E	7	16.270332	16.237686	16.287598	16.270332	16.270332
24	0xA59B4E	12	24.273305	24.241851	24.289814	24.273304	24.273305
32	0xA54B7C5E	17		32.241192	32.289476	32.272834	32.272834

Table 1. Comparison of the exact computation of the final state entropy, with upper and lower bounds.

4 Lower Bound of the Entropy

We can use the algorithm of the previous section and induction to give a lower bound of the final state entropy. For a given n and ℓ we first compute the entropy for an FCSR where all the carry bits are the ℓ least significant bits. Subsequently, we show that by moving a feedback position to the left, the direction of the most significant bit, we increase the final entropy. In both steps, we study all $0 \leq p \leq |q|$ case by case and use the summands of the entropy H_1, H_2, H_3 and H_4 as presented in Section 3.

4.1 Basis of Induction

For a fixed n and ℓ we study the final state entropy of an FCSR with

$$d = 2^{n-1} + 2^{\ell} - 1.$$

This represents an FCSR which has all its recurrent positions grouped together at the least significant bits (see Fig. 5). Like in the previous section, we are going

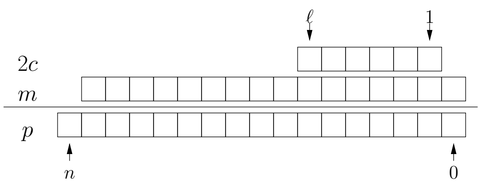

Fig. 5. FCSR with $d = 2^{\ell} - 1 + 2^{n-1}$

to compute the entropy case by case.

- $p = 0$ and $p = 1$.
- $1 \leq i \leq \ell$: Here we have $d_{i-1} = 1$, $\ell' = i$, $r = -1$ and 0 and thus $\ell'' = 0$.
- $\ell < i < n$: We have $d_{i-1} = 0$ and $\ell' = \ell$.
- $2^n \leq p \leq |q|$: Since it must hold that $r \leq \log_2(d^*) = \log_2(2^{\ell} - 1)$ we see that the only possible values for r are -1 and 0 and therefore $\ell'' = 0$.

So in this case, the final entropy is:

$$\begin{aligned}
H^f(n,d) = \quad & 2H_4(n,\ell) \\
& + \sum_{i=1}^{\ell} \left(H_2(n,-1,\ell,i,0) + H_2(n,0,\ell,i,0)\right) \\
& + \sum_{i=\ell+1}^{n-1} H_1(n,i,\ell,\ell) \\
& + H_3(n,-1,\ell,0) + H_3(n,0,\ell,0) \\
= \quad & n + \ell\left(2^{-n+\ell+1} - 2^{-n+1}\right) - 2^{-n-\ell+2} S_2(\ell) \ .
\end{aligned}$$

By using the lower bound (21) for $S_2(\ell)$ we can write:

$$H^f(n,d) \geq n + \frac{2^{-n+\ell+2}}{12\ln(2)}\left(3 - (4+\ell)2^{-2\ell} - 2^{-3\ell} + 2^{1-4\ell}\right) \ .$$

Let us examine the function $g(\ell) = 3 - (4+\ell)2^{-2\ell} - 2^{-3\ell} + 2^{1-4\ell}$. It is easy to verify that $g(\ell+1) - g(\ell) > 0$ for all $\ell \geq 1$. Thus, we can write $g(\ell) \geq g(1) = 7/4$ for all $\ell \geq 1$ and finally:

$$H^f(n,d) \geq n + 2^{-n+\ell}\frac{7}{12\ln(2)} \geq n \ . \tag{15}$$

4.2 Induction Step

We show that by moving one feedback position one position to the left, which means in direction of the most significant bit, the final state entropy increases. To prove this, we compare two cases $\mathcal{A}$ and $\mathcal{B}$. In Case $\mathcal{A}$ we choose an s such

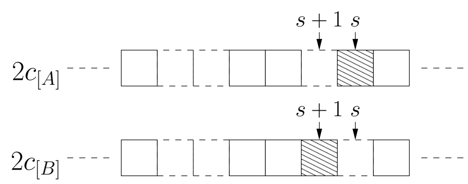

Fig. 6. Moving the feedback position.

that $d_{s-1} = 1$ and $d_s = 0$. It must hold that:

$$1 \leq s \leq n-2, \tag{16}$$

$$3 \leq n \ . \tag{17}$$

To create Case $\mathcal{B}$, we move the feedback at index s one position to the left. This is equivalent to:

$$d_{\mathcal{B}} = d_{\mathcal{A}} - 2^{s-1} + 2^s .$$

In Fig. 6, we display the two values $2c_{\mathcal{A}}$ and $2c_{\mathcal{B}}$ which we are going to use to compute the different finale entropies. Let

$$L = \#\{j < s | d_{j-1} = 1\}$$

be the number of feedback positions smaller than s. We mean by

$$I^r_{<s} = \{-1 \leq j < s : d_{i-1} = 0\}$$

the set of all indices smaller than s where there is no feedback and which are thus possible values for r. It must hold that $|I^r_{<s}| = s - L$.

We want to show that:

$$H^f_{\mathcal{B}} - H^f_{\mathcal{A}} \geq 0 \ . \tag{18}$$

To do this, we study the different cases of p. Each time, we examine if the summands from the algorithm in Section 3 are different or not. In the end we sum up the differences.

- $p = 0$ or $p = 1$: The summands of the entropy in $\mathcal{A}$ and $\mathcal{B}$ are the same.
- $i < s$: The summands of the entropy in $\mathcal{A}$ and $\mathcal{B}$ are the same.
- $n > i > s + 1$:
  - $d_{i-1} = 0$: In this case, i and ℓ' are the same for $\mathcal{A}$ and $\mathcal{B}$ and thus the summands of the entropy as well.
  - $d_{i-1} = 1$: We have:
    $$L + 2 \leq \ell' \leq \ell \ .$$
    - $r < s$: In this case, i, r, ℓ' and ℓ'' are the same and thus the summands of the entropy as well.
    - $s + 1 < r$: In this case, i, r, ℓ' and ℓ'' are the same and thus the summands of the entropy as well.
    - $s \leq r \leq s + 1$:
      - $\mathcal{A}$: Since for r it must hold that $d_{r-1} = 0$, we get $r = s + 1$ and thus $\ell'' = L + 1$. In this case, we have to count:
        $$H_2(n, s + 1, \ell, \ell', L + 1) \ .$$
      - $\mathcal{B}$: In this case, we have $r = s$ and $\ell'' = L$ and therefore the term:
        $$H_2(n, s, \ell, \ell', L) \ .$$
- $2^n \leq p \leq |q|$:
  - $r < s$: In this case, i, r and ℓ' are the same and thus the summands of the entropy as well.
  - $s + 1 < r$: In this case, i, r and ℓ' are the same and thus the summands of the entropy as well.
  - $s \leq r \leq s + 1$:
    - $\mathcal{A}$: We have $r = s + 1$ and $\ell'' = L + 1$. Therefore, the summand of the entropy in this case is:
      $$H_3(n, s + 1, \ell, L + 1) \ .$$

* $\mathcal{B}$: We have to consider $r = s$ and $\ell'' = L$ and therefore:

$$H_3(n, s, \ell, L) \ .$$

– $s \leq i \leq s+1$: We are going to use $\ell''(r)$ to denote the value of ℓ'' corresponding to a specific r.

- $i = s$:
  * $\mathcal{A}$: In this case, we have $d_{i-1} = 1$ and $\ell' = L+1$. Thus we have to count for each $r \in I^r_{<s}$:

  $$H_2(n, r, \ell, L+1, \ell''(r)) \ .$$

  * $\mathcal{B}$: Since $d_{i-1} = 0$ and $\ell' = L$ we get:

  $$H_1(n, s, \ell, L) \ .$$

- $i = s+1$:
  * $\mathcal{A}$: In this case, we have $\ell' = L+1$ and $d_{i-1} = 0$, thus we need to consider:

  $$H_1(n, s+1, \ell, L+1) \ .$$

  * $\mathcal{B}$: This time, we have $d_{i-1} = 1$. For $\ell' = L+1$, $r \in I^r_{<s}$ and $r = s$ we get:

  $$H_2(n, r, \ell, L+1, \ell''(r)) \ .$$

  In the case of $r = s$, we can write $\ell'' = L$.

By combining all these results, we get a difference of the final entropies of:

$$\begin{aligned}
H^f_{\mathcal{B}} - H^f_{\mathcal{A}} = & \sum_{\ell'=L+2}^{\ell} \left(H_2(n, s, \ell, \ell', L) - H_2(n, s+1, \ell, \ell', L+1) \right) \\
& + H_3(n, s, \ell, L) - H_3(n, s+1, \ell, L+1) \\
& + H_1(n, s, \ell, L) - \sum_{r \in I^r_{<s}} H_2(n, r, \ell, L+1, \ell''(r)) \\
& + \sum_{r \in I^r_{<s}} H_2(n, r, \ell, L+1, \ell''(r)) + H_2(n, s, \ell, L+1, L) \\
& - H_1(n, s+1, \ell, L+1) \\
& = 2^{\ell-1} \left(4\ell - 4L - 2 \right) + 2^{2\ell - L} + 2^{L+1} \left(3S_2(\ell - L - 1) - S_1(\ell - L) \right) \ .
\end{aligned}$$

If we use the lower bound (21) for $S_2(\ell - L - 1)$ and the upper bound (20) for $S_1(\ell - L)$, we can write:

$$H^f_{\mathcal{B}} - H^f_{\mathcal{A}} \geq 2^L \left((\ell - L) \frac{1}{4\ln(2)} + \frac{7}{12\ln(2)} + 2^{-(\ell-L)} \frac{14 - 12 \, 2^{-(\ell-L)}}{12\ln(2)} \right) \ .$$

From $\ell > L$, it follows directly (18), which means that the difference of the final entropies is greater or equal to 0.

Every FCSR with ℓ feedback positions can be build by starting with the FCSR described in Section 4.1 and successively moving one feedback position to the left. Thus, by combining (15) and (18) we write the following theorem.

Theorem 1. *An FCSR in Galois architecture, with given values for n and ℓ, has at least $n + 2^{\ell-n}\frac{7}{12\ln(2)} \geq n$ bits of entropy if all $2^{n+\ell}$ initial states appear with the same probability.*

5 Conclusion

If we allow all initial states of the FCSR with the same probability $2^{-n-\ell}$, we have an initial entropy of the state of $n + \ell$ bits. We showed in this article that already after one iteration, the entropy is reduced to $n + \ell/2$ bits.

As soon as the FCSR has reached its periodic behavior, the entropy does not reduce any more. In this article, we presented an algorithm which computes this final entropy in $O\left(n^2\right)$ steps. The algorithm is exact if the results of the sums $S_1(k) = \sum_{x=2^{k-1}+1}^{2^k} x\log_2(x)$ and $S_2(k) = \sum_{x=1}^{2^k-1} x\log_2(x)$ are known for $k \leq \ell$. For large values of ℓ, the same algorithm allows us to give close upper and lower bounds for the entropy by using approximations of $S_1(k)$ and $S_2(k)$.

In the end, we used the same algorithm to prove that the final state entropy never drops under n bits. One might argue that this is evident, since there are $|q|$ different values of p and $\log_2(|q|) \approx n$. However, it would be possible that the probabilities are not very regular distributed and that we would have a lower entropy. With our bound, it is sure that the entropy cannot drop under n bits.

The entropy of an FCSR decreases quite fast. However, it stays always larger or equal to n bits.

References

[AB05a] F. Arnault and T.P. Berger. Design and properties of a new pseudorandom generator based on a filtered fcsr automaton. *IEEE Transactions on Computers*, 54(11):1374–1383, 2005.

[AB05b] F. Arnault and T.P. Berger. F-FCSR: Design of a new class of stream ciphers. In Henri Gilbert and Helena Handschuh, editors, *FSE*, Lecture Notes in Computer Science, pages 83–97. Springer, 2005.

[ABL05] F. Arnault, T.P. Berger, and C. Lauradoux. F-FCSR. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/008, 2005. `http://www.ecrypt.eu.org/stream`.

[ABL06] F. Arnault, T.P. Berger, and C. Lauradoux. Update on F-FCSR stream cipher. In *SASC, State of the Art of Stream Ciphers Workshop*, pages 267–277, Leuven, Belgium, February 2006. ECRYPT Network of Excellence in Cryptology.

[ABM08] F. Arnault, T.P. Berger, and M. Minier. Some results on fcsr automata with applications to the security of fcsr-based pseudorandom generators. *IEEE Transactions on Information Theory*, 54(2):836–840, February 2008.

[CL94] R. Couture and P. L'Ecuyer. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Math. Comput.*, 62(206):799–808, 1994.

[GK02] M. Goresky and A. Klapper. Fibonacci and galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, 48(11):2826–2836, 2002.

[KG93] A. Klapper and M. Goresky. 2-adic shift registers. In *Fast Software Encryption - FSE'93*, Lecture Notes in Computer Science 809, pages 174–178. Springer Verlag, 1993.

[KG97] A. Klapper and M. Goresky. Feedback shift registers, 2-adic span, and combiners with memory. *J. Cryptology*, 10(2):111–147, 1997.

[MZ91] G Marsaglia and A Zaman. A new class of random number generators. *Annals of Appl. Prob.*, 1(3):462–480, 1991.

A Bounds for the Sums

In this section, we prove the following lower and upper bounds:

$$\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) \geq 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right)+2^{k-2}(k+1)+\frac{1-2^{-k+1}}{24\ln(2)}\ , \tag{19}$$

$$\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) \leq 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right)+2^{k-2}(k+1)+\frac{1-2^{-k}+3\,2^{1-2k}}{12\ln(2)}\ , \tag{20}$$

$$\sum_{x=1}^{2^k-1} x\log_2(x) \geq 2^{2k-1}\left(k-\frac{1}{2\ln(2)}\right)-2^{k-1}k+\frac{4+k+2^{-k+1}}{24\ln(2)}\ , \tag{21}$$

$$\sum_{x=1}^{2^k-1} x\log_2(x) \leq 2^{2k-1}\left(k-\frac{1}{2\ln(2)}\right)-k\,2^{k-1}+\frac{4+k+2^{-k}-2^{1-2k}}{12\ln(2)}\ . \tag{22}$$

The idea of this proof is that:

$$\frac{1}{2}\left(x\log_2(x)+(x+1)\log_2(x+1)\right) \approx \int_x^{x+1} y\log_2(y)\,dy\ ,$$

since $\log_2(x)$ increases much slower than x and, thus, $x\log_2(x)$ is almost a straight line. This integral can be directly computed by:

$$\begin{aligned}\int_x^{x+1} y\log_2(y)\,dy &= \frac{y^2}{2}\left(\log_2(y)-\frac{1}{2\ln(2)}\right)\Bigg|_{y=x}^{x+1} \\ &= \frac{1}{2}\left(x\log_2(x)+(x+1)\log_2(x+1)\right)-\frac{1}{4}\frac{2x+1}{\ln(2)} \\ &\quad +\log_2\left(1+\frac{1}{x}\right)\frac{x}{2}(x+1)\ .\end{aligned}$$

We use the approximation of the natural logarithm:

$$\frac{1}{\ln(2)}\left(\frac{1}{x}-\frac{1}{2x^2}+\frac{1}{3x^3}-\frac{1}{4x^4}\right) \leq \log_2\left(1+\frac{1}{x}\right) \leq \frac{1}{\ln(2)}\left(\frac{1}{x}-\frac{1}{2x^2}+\frac{1}{3x^3}\right)$$

for $x>0$ to get:

$$\frac{1+2x}{4\ln(2)}-\frac{\left(\frac{1}{3x}-\frac{1}{6x^2}+\frac{1}{2x^3}\right)}{4\ln(2)} \leq \frac{x}{2}(x+1)\log_2\left(1+\frac{1}{x}\right) \leq \frac{1+2x}{4\ln(2)}-\frac{\left(\frac{1}{3x}-\frac{2}{3x^2}\right)}{4\ln(2)}$$

and finally the bounds for the integral:

$$\int_x^{x+1} y\log_2(y)\,dy \geq \frac{1}{2}\left(x\log_2(x)+(x+1)\log_2(x+1)\right) - \frac{\left(\frac{1}{3x}-\frac{1}{6x^2}+\frac{1}{2x^3}\right)}{4\ln(2)} \quad (23)$$

$$\int_x^{x+1} y\log_2(y)\,dy \leq \frac{1}{2}\left(x\log_2(x)+(x+1)\log_2(x+1)\right) - \frac{\left(\frac{1}{3x}-\frac{2}{3x^2}\right)}{4\ln(2)}\ . \quad (24)$$

By combining the exact value of the integral:

$$\int_{2^{k-1}}^{2^k} y\log_2(y)\,dy = 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right)$$

with the lower bound:

$$\begin{aligned}
&\int_{2^{k-1}}^{2^k} y\log_2(y)\,dy \\
&\geq \frac{1}{2}\sum_{x=2^{k-1}}^{2^k-1} x\log_2(x) + \frac{1}{2}\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) - \frac{1}{4\ln(2)}\sum_{x=2^{k-1}}^{2^k-1}\left(\frac{1}{3x}-\frac{1}{6x^2}+\frac{1}{2x^3}\right) \\
&= \sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) - 2^{k-2}(k+1) - \frac{1}{4\ln(2)}\sum_{x=2^{k-1}}^{2^k-1}\left(\frac{1}{3x}-\frac{1}{6x^2}+\frac{1}{2x^3}\right)\ .
\end{aligned}$$

gained by means of (23), we receive the upper bound:

$$\begin{aligned}
\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) \leq\ & 2^{2k-3}\left(3k+1-\tfrac{3}{2\ln(2)}\right) + 2^{k-2}(k+1) \\
&+\tfrac{1}{4\ln(2)}\textstyle\sum_{x=2^{k-1}}^{2^k-1}\left(\tfrac{1}{3x}-\tfrac{1}{6x^2}+\tfrac{1}{2x^3}\right)\ .
\end{aligned} \quad (25)$$

In the same way, by using (24) we get:

$$\begin{aligned}
\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) \geq\ & 2^{2k-3}\left(3k+1-\tfrac{3}{2\ln(2)}\right) + 2^{k-2}(k+1) \\
&+\tfrac{1}{4\ln(2)}\textstyle\sum_{x=2^{k-1}}^{2^k-1}\left(\tfrac{1}{3x}-\tfrac{2}{3x^2}\right)\ .
\end{aligned} \quad (26)$$

Let us have a closer look at the two functions $g_1(x) = \frac{1}{3x} - \frac{1}{6x^2} + \frac{1}{2x^3}$ and $g_2(x) = \frac{1}{3x} - \frac{2}{3x^2}$. If we analyze their first derivatives, we see that $g_1(x)$ is decreasing for $x \geq 1$ and $g_2(x)$ is decreasing for $x \geq 4$. For the upper bound of the sum, we can write directly:

$$\begin{aligned}
\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) \leq\ & 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right) + 2^{k-2}(k+1) \\
&+\frac{1}{4\ln(2)}\sum_{x=2^{k-1}}^{2^k-1}\left(\frac{1}{3\ 2^{k-1}}-\frac{1}{6\ 2^{2k-2}}+\frac{1}{2\ 2^{3k-3}}\right) \\
&= 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right) + 2^{k-2}(k+1) + \frac{1-2^{-k}+3\ 2^{1-2k}}{12\ln(2)}
\end{aligned}$$

for all $k \geq 1$. In the case of the lower bound, we can write:

$$\begin{aligned}\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) &\geq 2^{2k-3}\left(3k+1-\frac{3}{2\ln(2)}\right)+2^{k-2}(k+1)\\&+\frac{1}{4\ln(2)}\sum_{x=2^{k-1}}^{2^k-1}\left(\frac{1}{3\;2^k}-\frac{2}{3\;2^{2k-2}}\right)\\&=2^{2k-1}\left(k-\frac{1}{2\ln(2)}\right)-2^{k-1}k+\frac{4+k+2^{-k+1}}{24\ln(2)}\end{aligned}$$

for $k \geq 3$. However, we can verify by numeric computation that the lower bound also holds in the cases $k=1$ and $k=2$. Thus, we have shown (19) and (20) for $k \geq 1$. Finally, by employing:

$$\sum_{x=1}^{2^K-1} x\log_2(x) = \sum_{k=1}^{K}\sum_{x=2^{k-1}+1}^{2^k} x\log_2(x) - K2^K$$

and the previous results, we receive the bounds (21) and (22).

B Proof of Lemma 1

Proof. Let p', m', c' be three bit strings of size k and let $X(p')$ be the number of possible pairs (m', c') such that $1p' = m' + c'$. We are going to use the following properties:

- For a given p', let (m', c') be such that $1p' = m'+c'$. We have two possibilities $10p' = 0m' + 1c' = 1m' + 0c'$ to create $10p'$. If (m', c') is such that $0p' = m' + c'$, we only have one possibility $10p' = 1m' + 1c'$ to build $10p'$. Thus, we can write:
$$X(0p') = 2\;X(p') + 1\;\left(2^k - X(p')\right). \tag{27}$$
- The only possibility to create $11p'$ is $11p' = 1m' + 1c'$ for a (m', c') such that $1p' = m' + c'$. Therefore, it holds that
$$X(1p') = X(p'). \tag{28}$$
- If the most significant bit of p' is a 0 followed by k times 1, we can only generate the carry in the last position. This is due to the fact that to create a carry and a 1 in the sum, we need three 1's which is not possible if we do not have a carry. So we have to create the carry in the highest position by $m'_k = c'_k = 1$. For the previous positions $0 \leq j < k$ we always have two possibilities for (m'_j, c'_j), respectively $(0, 1)$ and $(1, 0)$. Thus, in total we have 2^k choices for (m', c').
$$X(0\overbrace{1\ldots1}^{k}) = 2^k. \tag{29}$$

– As we have said above, it is not possible to create a carry with only 1's in p' and no previous carry, thus:

$$X(1\ldots1) = 0. \tag{30}$$

We are now going to use induction over the length k of the bit strings p', m' and c' to show that for all $0 \le x \le 2^k - 1$ there exists exactly one p' with $X(p') = x$.

Basis, $k = 1$:
From (29) and (30) we can easily see that:

$$X(1) = 0,$$
$$X(0) = 1.$$

Induction step, $k \to k+1$:
We assume that for every $0 \le x \le 2^k - 1$ there exists a p' of length k such that $X(p') = x$. We want to show that the same assumption holds for $k+1$.

– $x = 2^k$: From (29), we know that $X(p') = 2^k$ if $p' = 011\ldots1$ with k 1's.
– $0 \le x \le 2^k - 1$: From the assumption, we know that there exists a p' of length k with $X(p') = x$, thus by using (28) we can write $X(1p') = x$.
– $2^k < x \le 2^{k+1} - 1$: In this case $0 \le x - 2^k \le 2^k - 1$ and due to the assumption, we know that there exists a p' such that $X(p') = x - 2^k$. By using (27) we get:

$$\begin{aligned} X(0p') &= 2X(p') + \left(2^k - X(p')\right) \\ &= 2(x - 2^k) + (2^k - x + 2^k) \\ &= x. \end{aligned}$$

We have proven that for all $0 \le x \le 2^k - 1$ there exists a p' of length k, such that $X(p') = x$, *i.e.* there are exactly x pairs (m', c') of length k bits with $1p' = m' + c'$. Since in total there are only 2^k possible values of p' of length k, we see that there exists exactly one.