Journal of Electrical Engineering, Vol. 74, No. 3, 2023, pp. 184-187

sciendo

PAPER

On adding security to RLL - LDPC CCSDS codes without additional redundancy

Peter Farkaš[1,2]

In this paper it is presented that security can be added to three RLL LDPC codes specified by the Consultative Committee for Space Data Systems codes without additional redundancy and if some round conditions are valid, this can be done without the necessity to change the encoding and decoding procedures. The concept for obtaining lightweight security could be of interest for space communication, for communication in IoT or other networks in which the nodes are resource constrained.

Keywords: security, keyspace, LDPC codes, CCSDS codes, RLL codes, space communication, IoT

1. Introduction

Security and trustworthiness are enablers for most applications in space and IoT communication [1-3]. In space and IoT the energy, computational and hardware resources, which can be used for secure communication implementation are constrained. Therefore, at the present time the so-called lightweight security is considered as a possible solution. In lightweight security algorithms there has to be a tradeoff between complexity and robustness.

Numerous review publications are dealing with lightweight security for IoT [3-5]. Therefore, it will be not repeated and the interested reader can consult [1-5].

Recently in [6] it was shown that RLL properties can be implemented into the CCSDS-LDPC codes specified in [7]. This can be done without additional redundancy and in some circumstances without the need to change the decoding procedures for the original error correcting codes [8]. The approach presented in [6] was based on adding a so-called modifier to the encoded codewords with reordered symbols at the transmitting side. The influence of these changes can be eliminated at the receiver in the cases presented in [8].

The question arises if light security can also be implemented similarly into these CCSDS-LDPC codes. In this manuscript it is shown that the answer is positive.

2. Standard CCSDS-LDPC codes

The LDPC codes specified in [7] are binary linear block codes - with the following basic parameters: (128, 64), (256, 128), (512,128). In the pair (n, k), n and k denote codeword length and dimensionality of the code respectively. The codes are specified using the following parity check matrices:

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^7 & \mathbf{\Phi}^2 & \mathbf{\Phi}^{14} & \mathbf{\Phi}^6 & \mathbf{0}_M & \mathbf{\Phi}^0 & \mathbf{\Phi}^{13} & \mathbf{I}_M \\ \mathbf{\Phi}^6 & \mathbf{I}_M \oplus \mathbf{\Phi}^{15} & \mathbf{\Phi}^0 & \mathbf{\Phi}^1 & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^0 & \mathbf{\Phi}^7 \\ \mathbf{\Phi}^4 & \mathbf{\Phi}^1 & \mathbf{I}_M \oplus \mathbf{\Phi}^{15} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^{11} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^3 \\ \mathbf{\Phi}^0 & \mathbf{\Phi}^1 & \mathbf{\Phi}^9 & \mathbf{I}_M \oplus \mathbf{\Phi}^{13} & \mathbf{\Phi}^{14} & \mathbf{\Phi}^1 & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix} \quad (1)$$

$$\mathbf{H}_2 = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^{31} & \mathbf{\Phi}^{15} & \mathbf{\Phi}^{25} & \mathbf{\Phi}^0 & \mathbf{0}_M & \mathbf{\Phi}^{20} & \mathbf{\Phi}^{12} & \mathbf{I}_M \\ \mathbf{\Phi}^{28} & \mathbf{I}_M \oplus \mathbf{\Phi}^{30} & \mathbf{\Phi}^{29} & \mathbf{\Phi}^{24} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^1 & \mathbf{\Phi}^{20} \\ \mathbf{\Phi}^8 & \mathbf{\Phi}^0 & \mathbf{I}_M \oplus \mathbf{\Phi}^{28} & \mathbf{\Phi}^1 & \mathbf{\Phi}^{29} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{21} \\ \mathbf{\Phi}^{18} & \mathbf{\Phi}^{30} & \mathbf{\Phi}^0 & \mathbf{I}_M \oplus \mathbf{\Phi}^{30} & \mathbf{\Phi}^{25} & \mathbf{\Phi}^{26} & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix} \quad (2)$$

$$\mathbf{H}_3 = \begin{bmatrix} \mathbf{I}_M \oplus \mathbf{\Phi}^{63} & \mathbf{\Phi}^{30} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{25} & \mathbf{0}_M & \mathbf{\Phi}^{43} & \mathbf{\Phi}^{62} & \mathbf{I}_M \\ \mathbf{\Phi}^{56} & \mathbf{I}_M \oplus \mathbf{\Phi}^{61} & \mathbf{\Phi}^{50} & \mathbf{\Phi}^{23} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{37} & \mathbf{\Phi}^{26} \\ \mathbf{\Phi}^{16} & \mathbf{\Phi}^0 & \mathbf{I}_M \oplus \mathbf{\Phi}^{55} & \mathbf{\Phi}^{27} & \mathbf{\Phi}^{56} & \mathbf{I}_M & \mathbf{0}_M & \mathbf{\Phi}^{43} \\ \mathbf{\Phi}^{35} & \mathbf{\Phi}^{56} & \mathbf{\Phi}^{62} & \mathbf{I}_M \oplus \mathbf{\Phi}^{11} & \mathbf{\Phi}^{58} & \mathbf{\Phi}^3 & \mathbf{I}_M & \mathbf{0}_M \end{bmatrix}, \quad (3)$$

where $\mathbf{I}_M$, $\mathbf{0}_M$, $\mathbf{\Phi}^{\varepsilon}$ are $M \times M$ identity, all zero and ε-th right circular shifts of the identity matrix respectively and M=n/8. (In $\mathbf{H}_1$, $\mathbf{H}_2$ and $\mathbf{H}_3$, the values of M are 16, 32 and 64 respectively.) Operator $\oplus$ denotes modulo two additions (further denoted as mod 2).

We will call the four rows in the matrices (1)-(3) macro-rows. For example, the first macro-row from top in (1) is in Fig. 1.

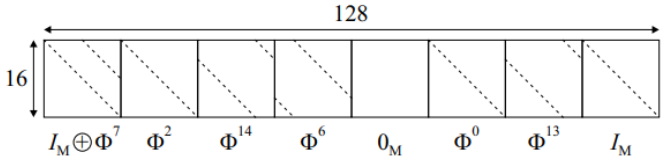

Fig. 1. Scatter chart of first macrorow from top in (1). The dotted lines correspond to ones in particular 16x16 circulants.

[1] Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Institute of Multimedia Information and Communication Technologies, Bratislava, Slovakia
[2] Pan-European University, Institute of Applied Informatics/Faculty of Informatics, Bratislava, Slovakia
p.farkas@ieee.org

https://doi.org/10.2478/jee-2023-0024, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

The LDPC codes could be encoded using the generator matrices in systematic form:

$$\mathbf{G} = [\mathbf{I}_{4M} \vdots \mathbf{W}] . \quad (4)$$

Submatrix $\mathbf{W}$ could be obtained as follows [7]:

$$\mathbf{W} = [\mathbf{P}^{-1} \vdots \mathbf{Q}] , \quad (5)$$

where $\mathbf{P}$ and $\mathbf{Q}$ are submatrices composed of the last and first $4M$ columns of the corresponding control matrices given by Eqns. (1-3).

3. RLL-LDPC codes obtained from standard CCSDS-LDPS

In [6] it was shown that RLL properties can be implemented into the standard codes [7] without additional redundancy and without the need to change the decoding procedures if some round conditions are valid [8]. The method is based on adding vectors called modifiers to encoded codewords with reordered symbols. It is illustrated in Fig. 2. The addition of vectors is in the finite field *GF*(2).

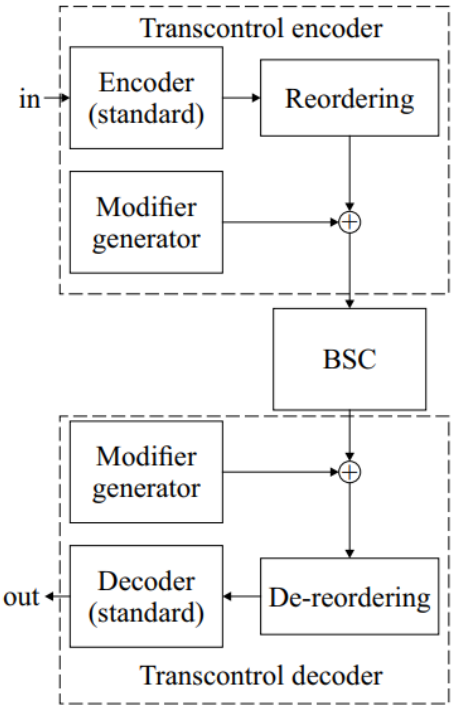

Fig. 2. The method for adding RLL properties into LDPC codes specified in [7] illustrated using a Binary Symmetric Channel (BSC) channel.

The purpose of the reordering is to obtain consecutive intervals of ones in each row from the macro-row and at the same time to minimize the gaps between them. In Fig. 3 the first macro-row from Fig. 1 is illustrated after the reordering.

If the rows in which are these intervals have even Hamming weight it is possible to invert an uneven number of symbols in them with the effect that not all symbols in these intervals in codewords will be the same [9]. Therefore, the modifier must have an uneven number of ones in the coordinates corresponding to these intervals. Consequently, after the modifier is added to the reordered codewords, the resulting binary vector will have RLL properties.

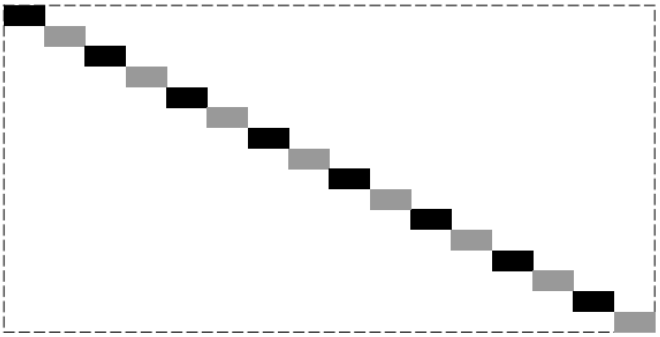

Fig. 3. Scatter chart of the first macrorow from top in (1). Bold lines correspond intervals containing 8 ones each and shadow lines correspond to the gaps in which the other symbols are positioned. Each gap corresponds to 8 binary symbols.

Observing Fig. 1 and Fig. 3 one can see that there are 8 intervals containing 8 ones each in which the modifier has to have an uneven number of ones and 8 intervals corresponding to gaps in which this restriction is not valid for the modifier. Therefore, the worst case run length of identical symbols in the RLL-LDPC code defined by (1) is in this case 22. The same value is valid also for codes given by (2) and (3). The reason is that in codes (2) and (3) there are 16 rows and 32 rows with 8 consecutive ones respectively. Consequently, in codes (2) and (3) there are 16 gaps and 32 gaps with 8 symbols respectively.

4. Concept for adding lightweight security to three RLL-LDPC CCDS

There is a broad spectrum of requirements which have to be fulfilled if secure communication has to be achieved. They correspond to different and distinct secure communication topics such as authentication, authorization, certification, copy resistance, covertness, integrity, ownership protection, secrecy and others. However, there are some overlaps between them and the central role in many applications plays the topics of classical cryptography [10].

In classical cryptography encryption functions are usually classified into two kinds. The first one is the block encryption function and the second one is the stream encryption function. In the first case the plaintext is divided into blocks of ν symbols – so called message blocks denoted as $\mathbf{x}$. Each message block is mapped into the so called ciphertext block $\mathbf{y}$.

$$\mathbf{y} = f_{\kappa}(\mathbf{x}) \tag{6}$$

It is supposed here that the vectors $\mathbf{x}$ and $\mathbf{y}$ will have coordinates from a $GF(2)$ and both will have length ν.

In (6) the index κ denotes a key which is an element from a set K denoted as keyspace.

In the case of stream ciphering the message has unspecified length. However, in practice the stream encryptor in order to be practically useable, has to have limited internal memory.

We will restrict our attention further only to so called additive stream ciphers. In these schemes the keystream is generated by a finite-state machine. It should resemble a random stream with equiprobable binary symbols. However, because the keystream is generated by finite state machine it is deterministic and can be predicted if the initial seed is known. Therefore, it is denoted as a pseudo random sequence (PN-sequence). The encryption function in this case is a mod 2 addition of the keystream to the datastream.

Taking this into account one can see an obvious similarity with the method for obtaining RLL properties in LDPC codes described earlier and illustrated in Fig. 2. Particularly the modifier is a binary vector and it is added mod 2 to the reordered codewords of the LDPC code. The restriction that in the "bold" intervals (denoted further as b-intervals) the modifier has to have an uneven number of ones makes it necessary to adapt the ciphering method to this requirement. In other words, the common additive stream cipher cannot be used directly. It must be adapted to the blocks of codewords and also to some subblocks named intervals as will be seen further.

In the following text, notes and concepts will be discussed heuristically as to how the modifier can be adapted in order that after its addition the transmitted words will have security properties. At the same time an estimation of the upper bounds for the key space size will be made using analytical approach for particular LDPC codes.

The LDPC codes defined by Eqns. (1-3) have b-intervals of the same length 8 [6]. Therefore, we can first estimate the number of combinations in which an uneven number of ones can be placed into the positions in the modifier corresponding to only one b-interval denoted N

$$N = \binom{8}{1} + \binom{8}{3} + \binom{8}{5} + \binom{8}{7} = 128 \tag{7}$$

The value of N is very small and therefore the fixed pattern of ones in each b-interval of the modifier would not provide any significant security. This implies that it is necessary to pseudo-randomly change the pattern for each b-interval. This can be combined with pseudorandom patterns for the positions of gaps (denoted as g-intervals below) which are between the b-intervals. There are 256 combinations of how the modifier can be designed in the positions corresponding to one g-interval.

We can now estimate the upper bounds on the sizes (cardinality) of key space for particular LDPC codes and give some more details on the constraints which have to be considered in future research when a concrete hardware or software implementation will be designed following the concept presented in this paper.

A. *Security obtained by only addition of the modifier*

First it will be supposed that only the addition of the modifier will be used to achieve lightweight security. In the LDPC code defined by Eqn. (1) there are altogether 8 b-intervals and 8 g-intervals. Consequently, the key space for this code is

$$N_1 = (N)^8 .2^{64} = 2^{120} \cong 1.33 \times 10^{36} \tag{8}$$

For the other two LDPC codes we get the following formulae

$$N_2 = (N)^{16} .2^{128} = 2^{240} \cong 1.77 \times 10^{72} \tag{9}$$

$$N_3 = (N)^{32} .2^{256} = 2^{480} \cong 3.12 \times 10^{144} \tag{10}$$

B. *Security obtained by only interleaving the positions of symbols in the codewords*

The numbers obtained using Eqns. (8-10) are relatively small and therefore it is desirable to increase the size of the keyspace further. One option is to use specific reordering (interleaving) of codeword symbols for each codeword of the LDPC code. Each reordering has to fulfill the condition that the 8 ones will be consecutive (in other words form the b-interval) in each uneven row of the first macro-row from the top in Eqns. (1-3). It is also necessary that the b-intervals and g-intervals alternate, each having 8 bits. However, it is possible to distinguish 8 shifts by one bit of this alternating structure of intervals.

In one b-interval there are $8!$ possibilities how the positions could be permutated. The positions of b-intervals inside a codeword by themselves could be also permutated in $8!$ ways. Consequently, the overall numbers of position permutations in particular codes are

$$P_1 = 8.(8!)^2 .64! \cong 2^{314} \cong 1.65 \times 10^{99} \tag{11}$$

$$P_2 = 8.(8!)^2 .128! \cong 2^{734} \cong 5.02 \times 10^{225} \tag{12}$$

$$P_3 = 8.(8!)^2 .256! \cong 2^{1702} \cong 2.25 \times 10^{512} \tag{13}$$

C. *Security obtained by combining A and B*

In case that the lightweight security obtained by approach *A* and approach *B* has to be increased further, it is possible to

combine them. The cardinality of the keyspace can be increased significantly by combining permutations (cardinality given by Eqns. (11-13) with all possible modifiers b-intervals (cardinalities given by Eqns. (8-10).

The modifier can be generated by obtaining appropriate blocks or vectors from a PN generator. These must have uneven Hamming weights in subblocks corresponding to b-intervals.

On the other hand, the reordering the codeword symbols (denoted as interleaving) must also preserve the property that the b-intervals and g-intervals alternate. We get the following estimations for the upper bound of key space cardinalities for particular LDPC codes if this combined approach is used

$$K_1 = 8.(N)^8 .2^{64}.(8!)^2.64! \cong 2^{434} \cong 2.19\times10^{135} \quad (12)$$

$$K_2 = 8.(N)^{16} .2^{128}.(8!)^2.128! \cong 2^{974} \cong 8.86\times10^{297} \quad (13)$$

$$K_3 = 8.(N)^{32} .2^{256}.(8!)^2 \times 256! \cong 2^{2182} \cong 7.04\times10^{656} \quad (14)$$

The obtained numerical results are summarized in Table I.

Table 1

RLL-LDPC	Cardinality upper bounds of keyspace (approximate)		
	Modifier	***Interleaving of symbols***	***Overall (combined)***
(1)	1.33×10^{36}	1.65×10^{99}	2.19×10^{135}
(2)	1.77×10^{72}	5.02×10^{225}	8.86×10^{297}
(3)	3.12×10^{144}	2.25×10^{512}	7.04×10^{656}

Note 1

It has to be noted that similarly as in [6] and [8] the argumentation is valid that in specific round conditions the influence of the reordering codeword symbols and the mod 2 addition of the modifier on the transmitting side can be eliminated on the receiving side. Consequently, the encoding and decoding procedures of the original CCSDS LDPC codes need not be modified if the proposed method for implementing RLL and security properties is used.

5. Conclusions

In this paper a concept allowing for adding RLL properties together with security properties into three LDPC codes specified by the CCSDS without additional redundancy was presented. The concrete structure and realizations of the software or hardware encryptor was not presented and it remains for further research to fill this gap. The encryptor in the first order should achieve robustness against different attacks. If it is designed for IoT applications or other networks in which the nodes have restricted resources it is desired that it will have limited energy consumption, computational complexity and memory requirements.

Acknowledgement

This work was supported by the Slovak Research and Development Agency under Contract no. APVV-19-0436 and it was also supported by the European Union (COST CA22168, 6G-PHYSEC) and by the Scientific Grant Agency of the Ministry of Education of the Slovak Republic and the Slovak Academy of Sciences (grant VEGA 1/0477/18).

References

[1] “Space Data Link Protocols-Sumary of Concepts and Rationale”, Informational Report CCSDS 130.3-G-1, Green Book, March 2008

[2] “Space Data Link Security Protocol.” CCSDS 355.0-B-2, Blue Book, July 2022

[3] M. N. Khan, A. Rao and S. Camtepe, "Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey," in *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4132-4156, 15 March 15, 2021, doi: 10.1109/JIOT.2020.3026493.

[4] S. Ganiev and Z. Khudoykulov, "Lightweight Cryptography Algorithms for IoT Devices: Open issues and challenges," *2021 International Conference on Information Science and Communications Technologies (ICISCT)*, 2021, pp. 01-04, doi: 10.1109/ICISCT52966.2021.9670281.

[5] S. Windarta, S. Suryadi, K. Ramli, B. Pranggono and T. S. Gunawan, "Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions," in *IEEE Access*, vol. 10, pp. 82272-82294, 2022, doi: 10.1109/ACCESS.2022.3195572.

[6] P. Farkaš and M. Rakús, “Adding RLL properties to four CCSDS LDPC codes without increasing their redundancy,” Computing and Informatics, vol. 42, no. 1, pp. 157-190 , 2023, doi: 10.31577/cai_2023_1_157

[7] "TC Synchronisation and Channel Coding," Blue Book, CCSDS 131.0-B-4, Washington, DC, USA, Recommended Standard, Issue 4, July 2021

[8] P. Farkaš and T. Páleník, "On Soft Decoding of Some Binary RLL-Transmission Codes in Systems with Coherent BPSK Modulation," *Cybernetics & Informatics (K&I)*, pp. 1-5, 2022, doi: 10.1109/KI55792.2022.9925949.

[9] P. Farkaš, F. Schindler, "Run length limited error control codes construction based on one control matrix," Journal of Electrical Engineering, vol. 68., no. 4, June 2017, pp. 322-324, doi: 10.1515/jee-2017-0046/

[10] R. E. Blahut, “Cryptography and secure communication,” Combridge University Press, UK, 2014.

Received 7 June 2023