Khulna University of Engineering & Technology

CSE 4120: Technical Writing and Seminar

Paper Title

ANS-based compression and encryption with 128-bit security

Authors

Seyit Camtepe, Jarek Duda, Arash Mahboubi, Paweł Morawiecki, Surya Nepal, Marcin Pawłowski, Josef Pieprzyk

Source

SPRINGER IJIS , Date of publication: July 13, 2022
DOI: https://doi.org/10.1007/s10207-022-00597-4

Submitted By-

Partho Choudhury Shoumya
Roll No: 1807021

Date: 05 July ,2023

Abstract

A significant portion of interactions on the Internet involve repetitive information and require robust security measures. To address this, a typical approach involves compressing the data stream to eliminate redundancy and then encrypting it using authenticated encryption protocols. While this approach is versatile and effective with various compression and encryption algorithms, it has one drawback: the algorithms are developed independently. It would be advantageous to have a single algorithm, known as compcrypt, that integrates compression and encryption, offering improved efficiency and security. This study explores the design of a compcrypt algorithm using ANS entropy coding. Initially, the fundamental properties of ANS are analyzed and it is demonstrated that a straightforward implementation of ANS with a concealed encoding table can be vulnerable to statistical attacks. Subsequently, investigation is done on the behavior of ANS when its states are randomly selected. Based on these insights, proposed compcrypt algorithm is developed, employing ANS with randomized state jumps and a sponge MonkeyDuplex encryption. The security and efficiency aspects of the proposed design is thoroughly discussed, which provides a high level of both confidentiality and integrity/authentication, ensuring 128-bit security. Implementation experiments reveal that this compcrypt algorithm achieves symbol processing rates of up to 269 MB/s (with a slight impact on compression rate) and 178 MB/s.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a noisy and unreliable communication channel there remains a problem of data transmission which was observed by Shannon[1]. It was shown that errors during transmission can be corrected if data are encoded with enough redundancy. To deal with the errors precisely, error correcting codes are developed to provide a carefully designed redundancy so the original data can be recovered even if some bits have been corrupted during transmission. However the reverse problem still persists. The main focus of theory of entropy coding is how to remove redundancy from transmitted data. This issue is extremely important for growing internet applications, where almost all data transmitted are highly redundant. Surely compression of data simply saves time and bandwidth. Additionaly, origianl data can be easily recovered by running a decompression algorithm.

Some entropy coding algorithm applies some well known techniques such as Huffman coding. However, this is suboptimal. The arithmetic/range coding offers a significant improvement as it allows to compress symbols for an arbitrary probability of distribution. Its main weakness, however, is a heavy computation overhead.

Asymmetric Numeral Systems (ANS) are a new and promising invention that provides both efficient and close to optimal entropy coding. To ensure confidentiality, integrity and authentication in the ever growing trend for people to turn to the internet for entertainment and work teleconferencing, a solution can be to use compression followed by encryption. However, a better option is to design a joint compression and encryption (called compcrypt).
A research problem that is used in this work is the design of a compcrypt algorithm such that:

$$\text{cost(compcrypt)} \ll cost(compression) + cost(encryption)$$

$$\text{security(compcrypt)} = \text{security(encryption)}$$

$$\text{comp\_rate(compcrypt)} \approx comp\_rate(compression)$$

Chapter 2

Background

As mentioned earlier, Huffman coding[7] is applied by the first entropy coding algorithm. It offers optimal compression for symbols that follow a probability distribution that are integer powers of 1/2. It is suboptimal as an interseting analysis of the code can be found in [6]. It is shown in [8], [9] and [11] the arithmetic/range coding offers a significant improvement as it allows to compress symbols for an arbitrary probability distribution. But it has a significant weakness, a heavy computation overhead. ANS provides promise of both efficient and close to optimal entropy coding as shown in [3], [5] and [10]. Duda and Niemiec consider a plain ANS as a compcrypt algorithm in their work [4]. The compcrypt algorithm holds up well against ciphertext-only adversries. But it is completely defenceless against integrity attacks as shown in the work [2]. The work proposes three versions of compcrypt with a better security level. However they apply a s little cryptography as possible.

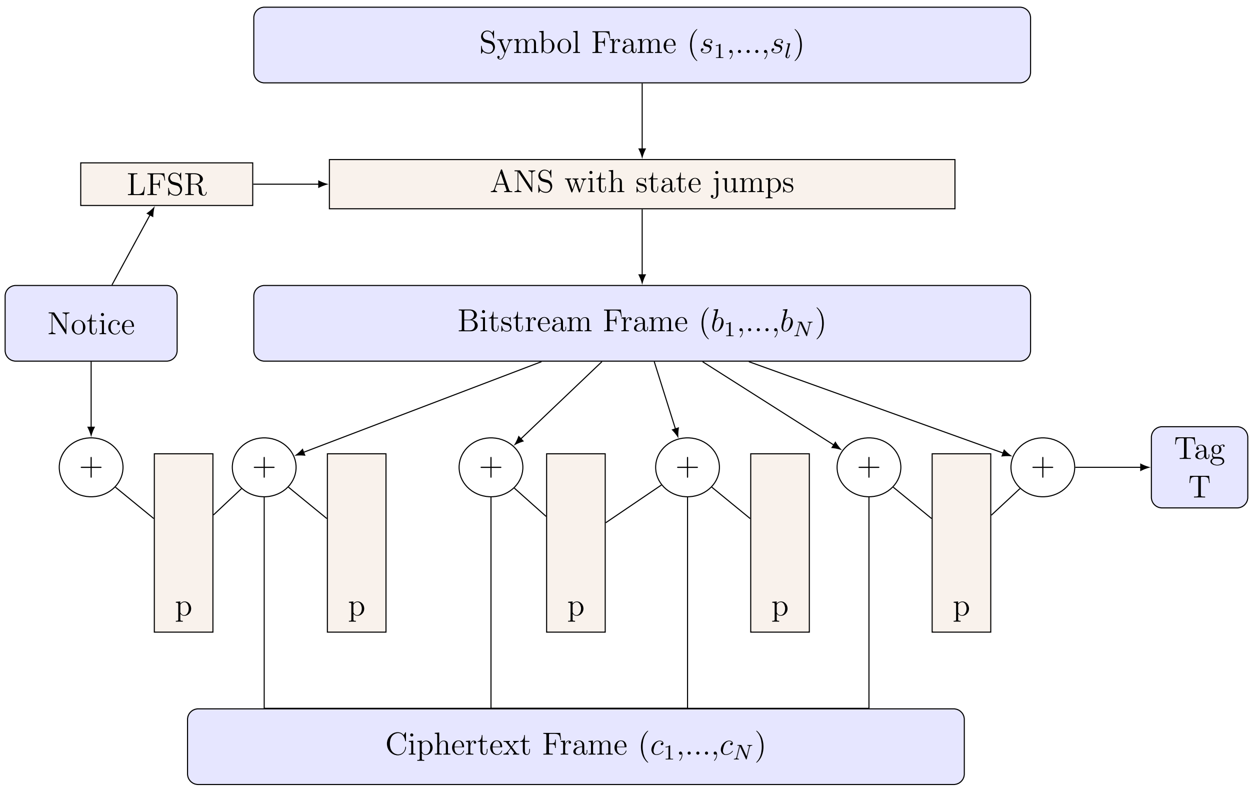
The previous works has lots of shortcomings. So in this work, the authors tried to overcome these thinigs as efficiently as possible. Mainly authors tried to:

- Analysis of a plain ANS compression rate: Authors exploit Markov chains to determine precisely probability distribution of ANS states.
- Statistical attacks on a plain ANS: It is shown that adaptive-statistics adversary is able to apply a divide-and-conquer algorithm to determine an encoding table much faster than the exhaustive search.
- Development of ANS variants: Here ANS are either fully randomized (as shown in [2]) or partially randomized.
- Design a 128-bit secure compcrypt algorithm. The algorithm uses a sponge MonkeyDuplex technique.

Chapter 3

Methodology

By utilizing the advantages of ANS techniques, this paper proposes a new algorithm which does compression and encryption in a single operation while maintaining the desired level of speed and security. The process is depicted in the flowchart below:

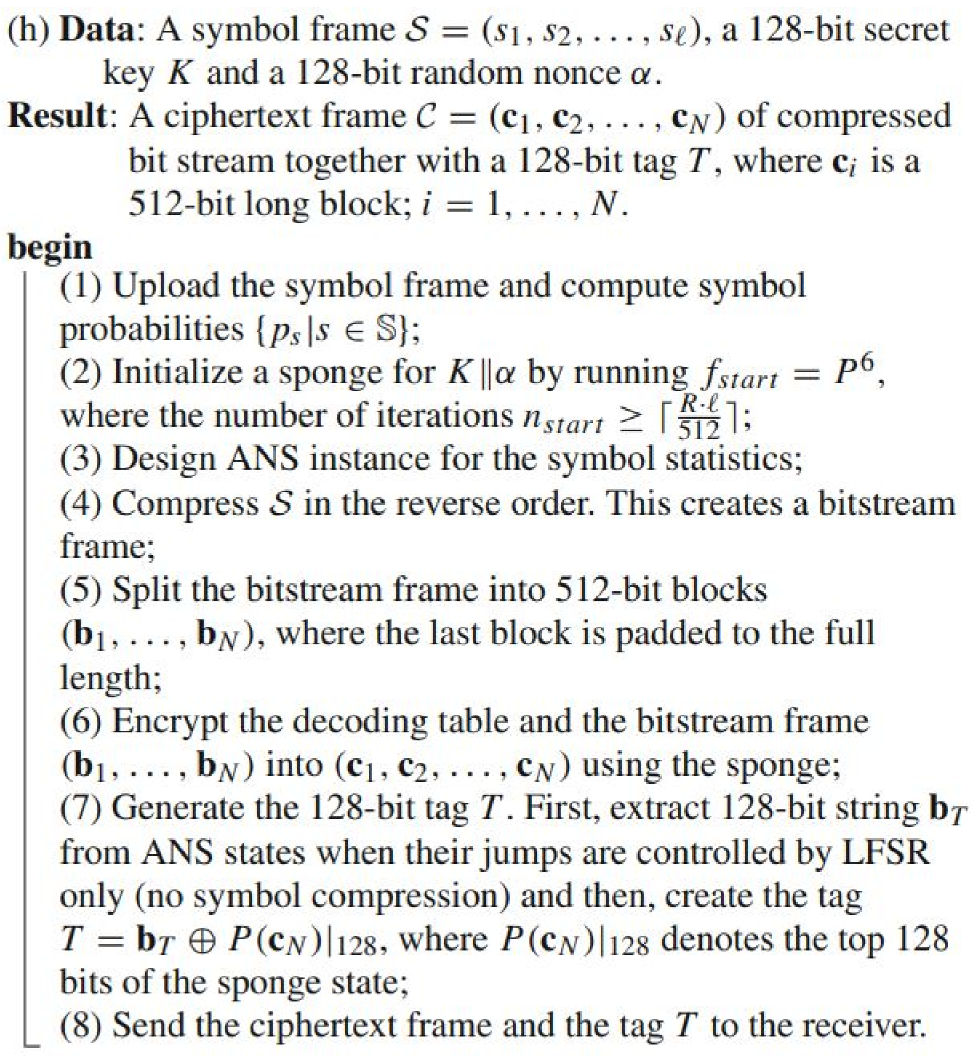


Symbol Frame (s_1,...,s_l)
LFSR
ANS with state jumps
Notice
Bitstream Frame (b_1,...,b_N)
+
p
Tag T
Ciphertext Frame (c_1,...,c_N)

Fig: Flowchart of data flow of compcrypt algorithm

3.1 Proposed Algorithm

The data flow of the proposed compcrypt algorithm is shown above. Now the algorithm steps are shown below:



```
(h) Data: A symbol frame $\mathcal{S} = (s_1, s_2, \ldots, s_\ell)$, a 128-bit secret key $K$ and a 128-bit random nonce $\alpha$.
Result: A ciphertext frame $\mathcal{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N)$ of compressed bit stream together with a 128-bit tag $T$, where $\mathbf{c}_i$ is a 512-bit long block; $i = 1, \ldots, N$.
begin
    (1) Upload the symbol frame and compute symbol probabilities $\{p_s | s \in \mathbb{S}\}$;
    (2) Initialize a sponge for $K \| \alpha$ by running $f_{start} = P^6$, where the number of iterations $n_{start} \geq \lceil \frac{R \cdot \ell}{512} \rceil$;
    (3) Design ANS instance for the symbol statistics;
    (4) Compress $\mathcal{S}$ in the reverse order. This creates a bitstream frame;
    (5) Split the bitstream frame into 512-bit blocks $(\mathbf{b}_1, \ldots, \mathbf{b}_N)$, where the last block is padded to the full length;
    (6) Encrypt the decoding table and the bitstream frame $(\mathbf{b}_1, \ldots, \mathbf{b}_N)$ into $(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N)$ using the sponge;
    (7) Generate the 128-bit tag $T$. First, extract 128-bit string $\mathbf{b}_T$ from ANS states when their jumps are controlled by LFSR only (no symbol compression) and then, create the tag $T = \mathbf{b}_T \oplus P(\mathbf{c}_N)|_{128}$, where $P(\mathbf{c}_N)|_{128}$ denotes the top 128 bits of the sponge state;
    (8) Send the ciphertext frame and the tag $T$ to the receiver.
```

Figure 3.1: Compcrypt Algorithm

Chapter 4

Result Analysis

The proposed compcrypt algorithm have been implemented on different platforms, namely PC, Raspberry Pi 3 and 4. Apart from Keccak , the authors also take into account two popular ciphers AES and ChaCha/Poly1305. Their experiments have been done for a source that contains 256 symbols, whose probabilities follow a geometric distribution with the parameter p = 0.5. Symbol frames are 32kB long.

The results of their experiments are shown in table below. The authors take efficiency of plain ANS as their baseline for comparison. ANS with state jumps (denoted as ANS) suffers a significant efficiency penalty. But uniform state distribution is essential for their compcrypt security claims. Their compcrypt ANS+Keccak has almost the same efficiency as ANS, which means that encryption comes at a very low cost. As a benefit the compcrypt claims 128-bit security (for both CPA and integrity).

Table 4.1: Comparison of Algorithms

Algorithm	**Efficiency PC**	**MB/s RP13**	**RP14**	**Efficiency PC**	**Drop RP13**	**RP14**	**128-bit Security**	Comp. Rate Loss
Plain ANS	220	17	60				x	0
ANS*	182	15	49	17%	12%	18%	x	$\approx 0.5\%$
ANS* + Keccak	178	13	48	19%	24%	20%	✓	$\approx 0.5\%$
ANS + Full_Keccak	135	9	27	39%	47%	55%	✓	0
ANS + AES_NI	152	8	29	31%	53%	52%	✓	0
ANS + AES	100	8	29	55%	53%	52%	✓	0
ANS + ChaCha20	95	6	16	57%	65%	73%	✓	0

Chapter 5

Discussion & Conclusion

In this study, a joint ANS-based compression and encryption with 128-bit security has been the focus of the work. Lots of shortcomings of previous works has been addressed and the authors tried to depict a more efficient algorithm. Based on the presentation of the work done in this paper we find some merits as below:

1. Plain ANS is susceptible to statistical attacks which may result in compromising of encoding table. But as here randomized ANS is used, divide-and-conquer attack has a little chance of success.

2. As 128-bit security is provided here, brute attack will surely be defended.

3. It is flexible enough to meet the current needs. If there is a need for a very fast compcrypt, then a designer can trade state-jump properties with the number of Keccak- f permutations P per round. If a higher than 128-bit security is required, then the proposed compcrypt can be re-designed after a careful security consideration.

There are also a few drawbacks of the work presented in the paper. They are:

1. The compression rate is slow as shown in the results. It could lead to bigger problems for large systems.

2. Compression loss is shown to be 0.5% which can lead to significant penalty if the algorithm is applied to large systems.

References

[1] Daniel J Bernstein. The poly1305-aes message-authentication code. In *International workshop on fast software encryption*, pages 32–49. Springer, 2005.

[2] Seyit Camtepe, Jarek Duda, Arash Mahboubi, Paweł Morawiecki, Surya Nepal, Marcin Pawłowski, and Josef Pieprzyk. Compcrypt–lightweight ans-based compression and encryption. *IEEE Transactions on Information Forensics and Security*, 16:3859–3873, 2021.

[3] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

[4] Jarek Duda and Marcin Niemiec. Lightweight compression with encryption based on asymmetric numeral systems. *arXiv preprint arXiv:1612.04662*, 2016.

[5] Jarek Duda, Khalid Tahboub, Neeraj J Gadgil, and Edward J Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In *2015 Picture Coding Symposium (PCS)*, pages 65–69. IEEE, 2015.

[6] David W Gillman, Mojdeh Mohtashemi, and Ronald L Rivest. On breaking a huffman code. *IEEE Transactions on Information theory*, 42(3):972–976, 1996.

[7] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[8] G Nigel N Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, volume 2, 1979.

[9] Alistair Moffat, Radford M Neal, and Ian H Witten. Arithmetic coding revisited. *ACM Transactions on Information Systems (TOIS)*, 16(3):256–294, 1998.

[10] Alistair Moffat and Matthias Petri. Large-alphabet semi-static entropy coding via asymmetric numeral systems. *ACM Transactions on Information Systems (TOIS)*, 38(4):1–33, 2020.

[11] Jorma J Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203, 1976.