Digital Arithmetic Coding with AES Algorithm

Archek Praveen Kumar
Lecturer,
Dept of Electronics & Communication
Amity University, Rajasthan

Deepika Bansal
Lecturer,
Dept of Electronics & Communication
Amity University, Rajasthan

ABSTRACT
The paper presents the security and compression of data by Digital Arithmetic coding with AES (Advanced Encryption Standard) algorithm. basic research is to arithmetically encode the data first and then encrypt it by using AES algorithm then transmits the code. At the receiving end the data is decrypted and decoded to produce the user data. The paper has an advantage of encoding/decoding and compression of data at a time. The input data size is of 128 bits or 256 bits in AES, so the idea is to compress the data before encryption by arithmetic coding. For encoding/decoding use digital arithmetic coding and for encryption/decryption we use AES algorithm. The arithmetic coding is similar to Huffman coding they both achieve their compression by reducing the average number of bits required to represent the symbol. Arithmetic coding stands out in terms of elegance, effectiveness and versatility, since it is able to work most efficiently in largest number of circumstances and purposes. AES is advanced encryption standard process where deals with substation and permutation of data for proper secure.

General Terms
Arithmetic encoding, AES encryption, AES decryption, Arithmetic Decoding.

1. INTRODUCTION

There are so many techniques for the transmission of data with proper encryption and decryption but the idea is to transfer the data over perfect security and compression at a time. Encoding transforms data into another format using a scheme that is publicly available without any key so that it can easily be reversed. Encryption is for maintaining data confidentiality and requires the use of a secret key in order to return to plaintext. Encryption does not expand the data except for a few bytes of padding at the end of the last block. The resulting data are not compressible at any rate because they are basically random, no algorithm is able to effectively compress them, so best method is to compress the data first, then encrypt them. The data is encoded with arithmetic coding where the result is arithmetically encoded according to probability and then the result is encrypted with AES algorithm. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. AES is based on a design principle known as a substitution-permutation network. AES is the first publicly accessible and open cipher approved by the National Security Agency (NSA) for top secret information.

2. GENERAL BLOCK DIAGRAM

The input data to be transmitted can be characters or numerical values. The input characters are divided in to sub groups and that group of data is encoded by using arithmetic coding and final range is encoded to hexadecimal or to binary numbers. Then by using AES algorithm that data is encrypted and transmitted through wired or wireless. At the receiving end the encrypted data is decrypted and then it is decoded by using arithmetic decoding technique and the original data is retrieved.

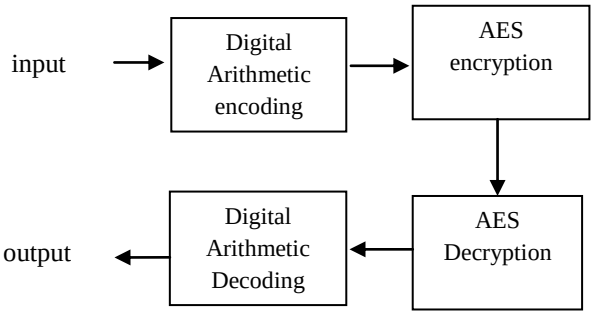

Fig 1: Block diagram of the arithmetic coding with AES

3. BLOCK DESCRIPTION

3.1 Digital Arithmetic Coding
Digital arithmetic coding is the advanced technique of the arithmetic coding. Firstly the arithmetic coding deals with probability theory. Basically all the characters, special characters and numerical are shared in a range from 0 to 1. Arithmetic coding is divided in two parts.

- Arithmetic coding is very simple. Its basic properties are used in the computational techniques required for a practical implementation.
- Considering practical implementation aspects, including arithmetic operations with low precision, the subdivision of coding and modeling, the realization of adaptive encoders and also analyze the arithmetic coding computational complexity, and techniques to reduce it.

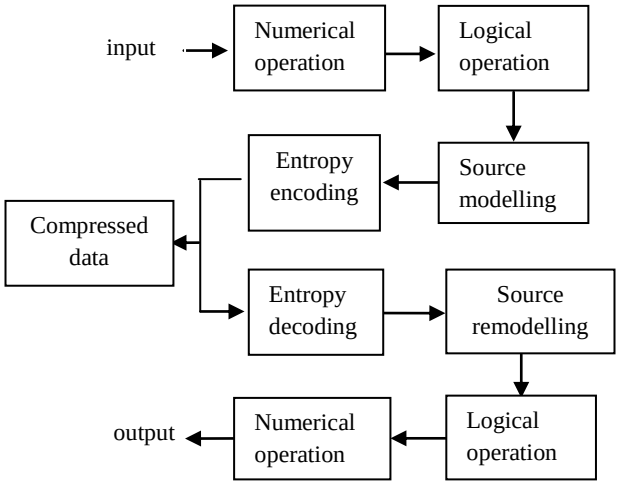

Fig 1: Flow of encoding and decoding processes for data compression

Transmission considers all the different entropy-coding methods, and their possible applications in compression applications, *arithmetic coding* stands out in terms of elegance, effectiveness and versatility, since it is able to work most efficiently in the largest number of circumstances and purposes. These are most desirable features.

- The compression of each symbol is provably optimal, when applied to independent and identically distributed sources.
- The same arithmetic coding implementation can effectively code all the diverse data created by the different processes as shown in Figure 2, such as modeling parameters, transform coefficients, signaling, etc. It is effective in a wide range of situations and compression ratios.
- It simplifies automatic modeling of complex sources, yielding near-optimal or significantly improved compression for sources that are not independent and identical.

Its main process is arithmetic, which is supported with ever-increasing efficiency by all general-purpose or digital signal processors (CPUs, DSPs). It is suited for use as a compression black-box. Compression applications employ a wide variety of techniques, have quite different degrees of complexity, but share some common processes. Figure 2 shows a diagram with typical processes used for data compression. These processes depend on the data type. As mentioned the input data is divided in to sub groups and that group of data is numerically processed. Where numerically processing like predictive coding and linear transforms, is normally used for waveform signals, like images and audio. The output data is logically processed, where Logical processing consists of changing the data to a form more suited for compression, like run-lengths, zero-trees, set-partitioning information, and dictionary entries. The next stage, source modeling, is used to account for variations in the statistical properties of the data. It is responsible for gathering Statistics and identifying data contexts that make the source models more accurate and reliable. Final process is entropy coding, which is the process of representing information in the most compact form. It may be responsible for doing most of the compression work, or it may just complement that has been accomplished by previous stages.

In arithmetic coding, a message is encoded as a real number in an interval from one to zero. Arithmetic coding typically has a better compression ratio than Huffman coding, as it produces a single symbol rather than several separate code words. The idea behind arithmetic coding is to have a probability line, (0-1), and assigned to every symbol a range in this line based on its probability, the higher range which assigns to it. Once we have defined the range and the probability line, start to encode symbols, every symbol defines where the output floating point number lands here is the arithmetic coding algorithm, with an example to aid understanding.

Start with an interval [0, 1), divided into subintervals of all possible symbols to appear within a message. Make the size of each subinterval proportional to the frequency at which it appears in the message.

Symbol	**Probability**	**Interval**
A	0.2	[0.0, 0.2)
E	0.3	[0.2, 0.5)
I	0.1	[0.5, 0.6)
O	0.2	[0.6, 0.8)
U	0.2	[0.8,1.0)

When encoding a symbol, "zoom" into the current interval, and divide it into subintervals like in step one with the new range. Example: suppose we want to encode "AIU". We "zoom" into the interval corresponding to "A", and divide up that interval into smaller subintervals like before. We now use this new interval as the basis of the next symbol encoding step.

Symbol	**New "a" Interval**
A	[0.0, 0.04)
E	[0.04, 0.1)
I	[0.1, 0.12)
O	[0.12, 0.16)
U	[0.16,0.2)

Repeat the process until the maximum precision of the machine is reached, or all symbols are encoded. To encode the next character "I", we use the "a" interval created before, and zoom into the subinterval "I", and use that for the next step. This produces

Symbol	New "U" Interval
A	[0.10, 0.102)
E	[0.102, 0.11)
I	[0.11, 0.112)
O	[0.112, 0.116)
U	[0.116, 0.12)

And lastly for the final result repeat the process. The "I" interval created before, to encode the next character "U" This produces

Symbol	New "U" Interval
A	[0.116, 0.1168)
E	[0.1168, 0.118)
I	[0.118, 0.1184)
O	[0.118432, 0.1192)
U	[0.1192, 0.12)

Transmit some number within the latest interval to send the codeword. The number of symbols encoded will be stated in the protocol of the image format, so any number within [0.1192, 0.12) will be acceptable. The process of encoding/decoding can be done by taking percentage of each interval with same range or by using the algorithms

At the receiving end the user have the same probability matrix showed in table 1. The decoding process will be, suppose if 0.1194 is transmitted. Check the value 0.1194 where, it comes in the range "A" which is first letter. Next the A range is divided in to table 2 and see for the value 0.1194 and its in "I", so second letter is "I". Now I range is divided in to sub interval as table 3 then 0.1194 is in range "U", so U is selected the same is shown in the decoding algorithm.

3.2 AES(Advanced Encryption Standard)

The arithmetically encoded data is encrypted or decrypted by using AES algorithm. AES is based on substation and permutation process. AES use symmetric key for encryption and the reverse transformation or decryption. AES operates on a 4×4 column-major order matrix of bytes. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits. The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of cipher text. The numbers of cycles of repetition are as 10 cycles of repetition for 128 bit keys, 12 cycles of repetition for 192 bit keys, 14 cycles of repetition for 256 bit keys. There are several processing steps in each round which depends on the encryption key. By using the same encryption key, a set of reverse rounds are applied to transform cipher text back into the original plaintext. It consists of a number of rounds where each round makes a number of transformations on a state, and uses a round key derived from the encryption key. The number of rounds depends on the block and key size. An encryption of a block starts with a transformation AddRoundKey, this is followed by an odd number of regular rounds, and ends with a special final round. The reason the final round is different has nothing to do with security, but was done to makes it possible to reuse encryption code to do the decryption.

High-level description of the AES algorithm process

- Key Expansion takes place where round keys are derived from the cipher key.
- In the Initial Round, add round key operation is done in which each byte of the state is combined with the round key using bitwise xor.
- In future Rounds certain operations like SubBytes, shift rows, mixcoloumns, add round key. Where SubBytes is a non-linear substitution step where each byte is replaced with another according to a lookup table. ShiftRows is the transposition step where each row of the state is shifted cyclically a certain number of steps. MixColumns are mixing operation which operates on the columns of the state, combining the four bytes in each column. As explained AddRoundKey in which each byte of the state is combined with the round key using bitwise xor.
- In Final Round no MixColumns but SubBytes, ShiftRows, AddRoundKey.

SubBytes are substitution of each byte in the block independent of the position in the state. This is an S-box. It is a bijection on all possible byte values and therefore invertible. This is the non-linear transformation. The S-box used is proved to be optimal with regards to non-linearity. The S-box is based on arithmetic in GF (2^8) (galois field). Galois field is a group of polynomials in which some external operations are done to the polynomial for encryption/decryption. Generally for GF (2) polynomial is x^2 +1, for GF (2^2) or GF (4) polynomial is x^3+x+1. Similarly GF (2^8) polynomial is $x^8 + x^5 + x^3 + x^2 + 1$. The S-box used is derived from the multiplicative inverse over **GF** (2^8), known to have good non-linearity properties. In the SubBytes step, each byte in the *state* matrix is replaced with a SubByte using an 8-bit substitution box where this operation provides the non-linearity in the cipher. The S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points, and also any opposite fixed points.

ShiftRows is a cyclic shift of the bytes in the rows in the state and is clearly invertible.The ShiftRows is operation on the rows of the state where cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular by n-1 bytes. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively.

In the MixColumns , four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs, where each input byte affects all four output bytes, During this operation, each column is multiplied by the known matrix that for the 128 bit key. The multiplication operation is defined as, multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing xor with the initial un-shifted value. In more general sense, each column is treated as a polynomial over GF (2^8) and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from GF (2) [x]. The MixColumns step can also be viewed as a multiplication by a particular MDS matrix in a finite field.

The AddRoundKey step where the subkey is combined with the state. A subkey is derived from main key and each subkey is the same size as the state for each round. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR. The Roundkeys are made by expanding the encryption key into an array holding the RoundKeys one after another. The expansion works on words of four bytes. Mk is a constant defined as the number of four bytes words in the key. The encryption key is filled into the first Mk words and the rest of the key material is defined recursively from proceeding words. The word in position i, W[i], except the first word of a RoundKey, is defined as the XOR between the proceeding word, W[i−1], and W[i−Mk]. The first word of each RoundKey, W[i] (where i mod Mk == 0), is defined as the XOR of a transformation on the proceeding word, T(W[i − 1]) and W[i − Mk]. The transformation T on a word, w, is w rotated to the left by one byte, XOR'ed by a round constant and with each byte substituted by the S-box.

As the data encoded from arithmetic encoding is encrypted by using AES algorithm and transmitted. At the receiving end by using the same keys the data is decrypted and it's decoded by using arithmetic decoding process and data is retrieved.

4. RESULT

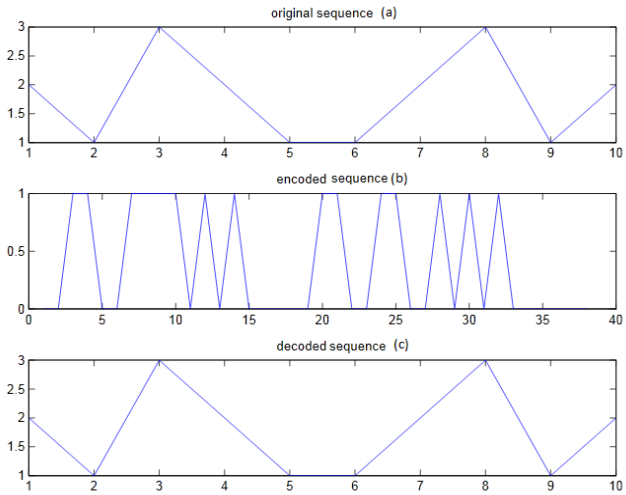

Fig 1: Output of MATLAB coding

Firstly a Random input sequence is generated which is shown by the original sequence. After the original sequence is generated, it is encoded arithmetically by some functions which are shown as encoded sequence in the graph. Now AES place vital role for encryption and decryption of the encoded sequence. The decrypted code is decoded arithmetically to produce original sequence which is shown as decoded sequence in the graph.

5. CONCLUSION

The input data will be either 128 bits or 256 bits in AES. By using MATLAB the sequence is encoded, encrypted, decrypted and decoded sequentially. Before encryption compression of data has been done by arithmetic coding in MATLAB. Error detection and correction capabilities are proved to be high. Given the paper with the final result, there are many future improvements we would pursue having more time. Much more efficient compressing techniques can be used before encryption. For every protocol this can be applicable

6. ACKNOWLEDGMENT

First and foremost we thank Almighty God whose grace was there throughout the course of the paper. We would like to thank our parents for their endless support. Finally like to express our deep sense of gratitude to our colleagues, Dept. of Electronics & Communication, Amity University for their enduring support and encouragement.

7. REFERENCES

[1] M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems.

[2] Practical Implementations of Arithmetic Coding, Paul G. Howard and Je_rey Scott Vitter, Brown University, Department of Computer Science, Technical Report No. 92{18 Revised version, April 1992 (Formerly Technical Report No. CS{91{45) James A. Storer, ed., Kluwer Academic Publishers, Norwell, MA, 1992.

[3] Introduction to coding theory by J.H. VAN LINT, 3rd edition, springer publications.

[4] Information theory by STEVES RAMEN, 1st edition, springer publications.

[5] Cryptography and network security by William stalings, 4th edition.

[6] http://www.codemiles.com/java/advanced-encryption-standard-aes-example-cipher-step1-t182.html

[7] http://www.zipworld.com.au/~isanta/uni/arithmetic.htm

[8] http://www.arturocampos.com/ac_arithmetic.html

[9] http://en.wikipedia.org/wiki/Arithmetic_coding