Adaptive Self-Correcting Floating Point Source Coding Methodology for a Genomic Encryption Protocol

Harry C. Shaw
Comm. Systems Branch
NASA/GSFC
Greenbelt, MD

Sayed Hussein
ECE Dept.
George Washington Univ.
Washington, DC

Hermann Helgert
ECE Dept.
George Washington Univ
Washington, DC

ABSTRACT
We address the problem of creating an adaptive source coding algorithm for a genomic encryption protocol using a small alphabet such as the nucleotide bases represented in the genetic code. For codewords derived from an alphabet of *N* plaintext with probability of occurrence, *p*, we describe a mapping into a floating point representation of the codewords which are translated into genomic codewords derived from a novel modification of the Shannon-Fano-Elias coding process. Errors in the reverse decoding process are processed through an adaptive, self-correcting codebook to determine the best fit codeword decoding solution. A genetic algorithmic approach to error correction within the source coding is also summarized.

General Terms
Data Confidentiality and Network Authentication

Keywords
Source coding, genetic algorithms, probability mass functions. Shannon-Fano-Elias.

1. INTRODUCTION
Genomic encryption protocols are being widely studied for implementation in advanced information security [1], [2]. In this paper, we present a source coding system for subsequent encryption via a system that emulates the mechanisms of regulation of gene expression [3]. However, utilization of such a protocol requires an efficient source coding scheme that is optimized for the requirements of the electronic domain (bandwidth and channel efficiency, error detection and correction, signal recovery in the presence of noise and interferers, etc.) In this paper, we address the mapping of a plaintext source code alphabet into genomic codes using the matrix cofactors of a solution of linear equations. The transmitted data content is a series of floating point matrix cofactors. At the receiving end, the receiver applies a decoding algorithm to recover and invert the cofactor matrix and correct the rounding and floating point errors via an adaptive source codebook. A genetic algorithm provides an efficient method to determine to correct errors in received codewords based upon the fitness of approximated codewords.

Codeword lengths are adaptable based upon the entropy of user selected source. This source could be a user plaintext, the selected genome of one or more species, or other sources as required. The genomic alphabet can consist of the four most commonly found nucleotides (adenine, cytosine, guanine and thymine. It can be expanded to include epigenetic marking (methyl-cytosine) [4], mutagenic base modifications (xanthine, hypoxanthine) [5], the RNA base uracil, and so forth. The method is extensible to the proteome and other domains within the space of gene expression products.

A large variety of methods have been published to utilize DNA transcription and translation in cryptographic systems. DNA cryptography using the central dogma of biology has been proposed for mobile ad hoc networks [6]. It takes plaintext through a process of DNA→RNA→Amino Acid coding. A combination of DNA computing and Elliptic Curve Cryptography has been described [7] for a powerful form of DNA encryption. It permits encrypted traffic over communication links which may not be secure. A symmetric key block cipher approach using DNA transcription and translation has been demonstrated by [8]. Other forms of DNA encryption include:

- Image compression – encryption using a DNA-based alphabet and a genetic algorithm based compression scheme [9].
- DNA encryption utilizing gel electrophoresis images and a molecular checksum [10].
- Steganographic approach using DNA as a natural template for hidden messages [11].
- DNA watermarks to identify genetically modified organisms utilizing the DNA-Crypt algorithm permitting a user to insert encrypted data into a genome of choice [12], [13].

2. THE METHOD

2.1 High-level description of the transmitter source coding process
Consider a memoryless source generating letters from an alphabet $A_1 = \{a_1, a_2, a_n\}$ with a source taken from a probability mass function $P = \{p_1, p_2, p_n\}$. Let the source generate a message X such that: $X=x_1x_2...x_i \in A \ \forall i$ where i represents the word order of the message. The message X is serialized and subdivided into character blocks of size r, and r-sized blocks are arranged into k sized word blocks in a set L as shown in Figure 1.

The words are lexicographically coded in the format of $\alpha.\beta_1\beta_2...\beta_k$ where α is the Huffman decimal code for the first letter and β_k are the subsequent Huffman decimal codes for remaining letters. Clearly, if the character blocks are long enough, precision and accuracy of subsequent floating point computations would be a concern. Therefore, the character block size is made adaptable to the floating point capabilities of the transmitting and receiving system.

$$\text{words}: x_1 x_2 x_3 x_4, \ldots x_i$$
$$\downarrow$$

$(r\ x\ k\ x\ n)$	1	$(a_1,a_2,..a_r),(a_1,a_2,..a_r),\ldots,(a_1,a_2,..a_r)$
blocks	2	$(a_1,a_2,..a_r),(a_1,a_2,..a_r),\ldots,(a_1,a_2,..a_r)$
	n	$(a_1,a_2,..a_r),(a_1,a_2,..a_r),\ldots,(a_1,a_2,..a_r)$
		1 2 … k

Figure 1. Organization of words for the source coding protocol. Words are divided into equal blocks *r* characters long. The new blocks are coded in groups of *k* blocks at a time resulting in *r x k x n* organization to begin the source coding process.

A pilot channel link between transmitter and receiver can be used to establish the optimal character block size based upon current channel state information. The source coding can be implemented in conjunction with a subsequent channel coding algorithm.

Let $R=\{R_1, R_2,\ldots R_n\}=\{(a_1, a_2,\ldots,a_r)_1, (a_1, a_2,\ldots,a_r)_2\ldots (a_1, a_2,\ldots,a_r)_n$ then:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \ldots & R_{1k} \\ R_{1k} & R_{11} & R_{12} & \ldots & R_{1k-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ R_{12} & R_{13} & \ldots & R_{1k} & R_{11} \end{bmatrix}_n \begin{bmatrix} q_1 \\ q_2 \\ \ldots \\ \ldots \\ q_k \end{bmatrix}_n = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ \ldots \\ y_k \end{bmatrix}_n \quad (1)$$

And Q_n is defined as:

$$Q_n = \begin{bmatrix} q_1 & q_2 & \ldots & q_k \\ q_k & q_1 & \ldots & q_{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ q_2 & q_3 & \ldots & q_1 \end{bmatrix}_n \quad (2)$$

This is transformed into a matrix of cofactors by:

$$Q_n^{-1} = \frac{1}{Det\, Q_n^T} \begin{bmatrix} C_1 & C_2 & \ldots & C_k \\ C_k & C_1 & \ldots & C_{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ C_2 & C_3 & \ldots & C_1 \end{bmatrix}_n \quad (3)$$

Let $C = \{C_{11}, C_{12}, \ldots C_{nk}\}$ which code the entire set of the original words in *X*. Treating *C* as a set of symbols from an alphabet of base 10 characters and sign characters $A_2 = \{+,-,0,1,2,3,4,5,6,7,8,9\}$ the entropy in bits of code *C* can be derived from the standard definition

$$H(C) = -\sum_{i=1}^{12} p_i \log_2(p_i) \quad (4)$$

Every unique plaintext message will have a unique distribution of symbols for each set *C*. The entropy, *H*, represents the lower bound on symbol length. However, the goal of coding set *C* is not minimum symbol length but a prefix-free code with symbol error correction capability at the decoder codebook. Therefore, a modification to the Shannon-Fano-Elias source coding algorithm has been developed for this purpose. Following Shannon-Fano-Elias, assume $p(x) > 0$ for all *x*, the cumulative distribution function, $F(x)$ is:

$$F(x) = \sum_{a \le x} p(a) \quad (5)$$

Shannon-Fano-Elias modifies the CDF as [14]:

$$\bar{F}(x) = F(x) + \frac{1}{2} p(x) \quad (6)$$

We define instead

$$F'(x) = F(x) - \frac{1}{2} p(x) \quad (7)$$

and the binary code length, λ, remains as

$$\lambda(x) = \left\lceil \log\left(\frac{1}{p(x)}\right) \right\rceil + 1 \quad (8)$$

with brackets indicate rounding to the next higher integer. Let

$$\nu(x) = \left\lceil \frac{1}{1 - F'(x)} + V(x) \right\rceil \quad (9)$$

where $V(x)$ is an offset value that shifts the decimal value of ν into a desired range between adjacent values of F(x). The codeword is:

$$J(x) = binary(\nu(x)) \mid \lambda(x) \quad (10)$$

$J(x)$ is the binary codeword truncated to $\lambda(x)$ bits. Table 1 illustrates an example.

Table 1. Modified Shannon-Fano-Elias Coding

Symbol	p(x)	F`(x)	λ	V	ν	Codeword
0	0.2924	0.1462	3	0	2	010
1	0.0903	0.3376	5	0	2	00010
7	0.0832	0.4243	5	13	15	01111
9	0.0744	0.5031	5	14	17	10001
3	0.0743	0.5774	5	21	24	11000
6	0.0675	0.6483	6	0	3	000011
8	0.0672	0.7157	6	7	11	001011
4	0.0671	0.7829	6	20	25	011001
2	0.0670	0.8499	6	21	28	011100
5	0.0666	0.9167	6	28	41	101001
+	0.0259	0.9630	7	0	28	0011100
-	0.0241	0.9880	7	42	126	1111110

The expected length of this code versus the entropy is < *H(x) + 2*, as in Shannon-Elias-Fano. This construction produces a prefix-free code, which, as expected, satisfies the Kraft inequality,

$$\sum_i D^{-\lambda_i} \le 1 \tag{11}$$

The next step is to concatenate the binary code words for each *k*-sized block of codewords. Each *k*-sized block may be preceded with a prefix-free preamble code that is not a member of the codebook. The resulting series is labeled as X_T, where the subscript *T* refers to the transmitter

$$X_T = C_{11} \| C_{12} \| .. \| C_{nk} \tag{12}$$

Two additional sequences are brought into the scheme. K_T and P_T. K_T is a binary sequence representing a unique symmetric encryption key. Ostensibly for this application it is the binary translation of gene sequence from a genomic alphabet as described in the introduction, or it could be any user specified binary sequence satisfying the requirements of a symmetric encryption key. P_T is a binary sequence representing a message authentication code that is a pre-shared secret between transmitter and receiver. For this application it is the binary translation of gene sequence from a genomic alphabet but it could also be any user specified binary sequence satisfying the requirements of keyed message authentication code. The final four steps are as follows:

$$\begin{aligned} F_T &= X_T \oplus K_T \\ G_T &= F_T \oplus P_T \\ M_T &= G_T \,||\, \mathrm{P}_T \\ M_T &\rightarrow M(DNA)_T \end{aligned} \tag{13}$$

Where the DNA alphabet can consist of symbols from a genomic alphabet such as:

$$A_D = \{A,T,C,G,MeC,H,X\} \tag{14}$$

One possible coding scheme for this alphabet using the previously described procedure is shown in table 2.

Table 2. DNA Base Source Coding

Symbol	p(x)	F`(x)	λ	V	ν	Codeword
A	0.2100	0.1050	4	0	2	0010
G	0.2100	0.3150	4	4	6	0110
C	0.2100	0.5250	4	6	9	1001
T	0.2100	0.7350	4	8	12	1100
MeC	0.0675	0.8738	5	12	20	10100
H	0.0672	0.9411	6	0	17	010001
X	0.0253	0.9874	8	0	80	01010000

A, G, C, and T represent the four main nucleotide bases adenine, guanine, cytosine, and thymine. MeC represents 5-Methylcytosine, an important epigenetic marker, H represents hypoxanthine and X represents xanthine. H and X are mutagenic deaminations of DNA bases that occasionally occur in gene sequences. MeC operates as an epigenetic marker by altering the pattern of gene expression without changing the basic sequence. Subsequent encryption steps can utilize all of the bases represented in this alphabet for creating different types of encrypted codes. The entropy of the DNA bases in a genomic sequence is also a source of potential encryption coding by skewing the code lengths of DNA based source code sequences. Certain genomes have A-T or G-C base pair contents the deviate significantly from a uniform distribution. The genome of *Mycoplasma genitalium G37* (National Center for Biotechnology Information accession number NC_000908.2) has a low G+C content of 34% [15]. Utilization of a genomic sequence with a high concentration of CpG (cytosine-phosphate-guanine) islands can be used to alter the source code sequence lengths for each base from what would be expected in a uniform distribution of the four main bases (A-T, C-G).

The resulting message is designated M_T. M_T contains the coded message contents and the required hash code necessary for the receiver to authenticate the transmitted message. M_T represents the basic, unencrypted message unit that is to be subjected to higher level encryption at the transmitter.

2.2 High-level description of the receiver source decoding process

The receiver receives the message, and creates a bit stream labeled M_R, which represents the best estimate at the receiver of the transmitted message. M_R is decrypted and the receiver computes the P_R pre-shared secret message authentication code and determines that it matches the P_T. Then M_R is sent to the receiver source decoder. The description of the process resumes at this point.

The final four steps of the transmission source coding are reversed in decoding (the subscript *R* refers to processes at the receiver:

$$\begin{aligned} M(DNA)_R &\rightarrow M_R \\ M_R &= G_R \,||\, \mathrm{P}_R \\ F_R &= G_R \oplus P_R \\ X_R &= F_R \oplus K_R \end{aligned} \tag{15}$$

Extending from the previous definition at the transmitter:

$$X_R = C_{11,R} \| C_{12,R} \| .. \| C_{nk,R} \tag{16}$$

Using linear algebra, the cofactor matrix is assembled and the inverse yields the original lexicographic codes. Summarizing these steps yields:

$$Q_{n,R} = \begin{bmatrix} C_1 & C_2 & \dots & C_k \\ C_k & C_1 & \dots & C_{k-1} \\ \dots & \dots & \dots & \dots \\ C_2 & C_3 & \dots & C_1 \end{bmatrix}_n^{-1} \tag{17}$$

$$R_n = \begin{bmatrix} R_{11} & R_{12} & \dots & R_{1k} \\ R_{1k} & R_{11} & R_{12} & R_{1k-1} \\ \dots & \dots & \dots & \dots \\ R_{12} & R_{13} & \dots & R_{11} \end{bmatrix}_n \tag{18}$$

The *R* coefficients then map back to the original words in set

$X.\ R = \{R_1, R_2, \ldots R_n\} \rightarrow \{(a_1, a_2, \ldots, a_r)_1, (a_1, a_2, \ldots, a_r)_2 \ldots (a_1, a_2, \ldots, a_r)_n \rightarrow X = x_1 x_2 \ldots x_i \in A \ \forall i$

3. Example

An example is taken from a snippet from the script of the first line of Shakespeare's Hamlet soliloquy: 'HAMLET To be or not to be that is the question Whether'. We compare the effects of dividing the plaintext phrase into 3 word blocks, 6 characters per block, versus 3 word blocks and 4 characters per block. The lexicographic codes and the uncorrected plaintext recovery are shown in table 3. Computations were performed on a 32-bit HP Pavilion dv4 PC under Windows 7 using Microsoft Visual Basic 2010 and Microsoft Excel 2007. The remaining errors in the recovered 6 character block codes are easily corrected at the source codebook level.

3.1 Genetic Algorithm (GA) for Source code error correction

Errors may occur at any position within a coefficient. Assume that an error is received in the DNA code and is propagated into and the binary code received at the receiver is decoded and subsequently into the set of cofactors $\{C_{n1\text{-}R}, C_{n2\text{-}R}, \ldots, C_{nk\text{-}R}\}$. The remaining source of error is in inversion of the cofactor matrix. The receiver does not know the precision of the original cofactors; therefore arbitrary truncation will produce uneven results. The original source code and the recovered source code can be represented as vectors having magnitude and phase with an error vector. Let b = codeword $C_{nk\text{-}R}$ and let a represent a candidate codeword for b. A genetic algorithm approach can be used to determine the fitness of a series of candidate codewords derived from the recovered codeword. The codeword with the highest fitness score is the best estimate of the recovered text. The highest fitness is derived by ordering a series of candidate codewords that minimize the distance, d, between the candidate codeword and the received codeword. Thus $d = |b - a| \rightarrow 0$ is the criteria for optimal candidate codeword selection and the most fit codeword possessing a zero distance between candidate and received codeword. Thus, the genetic algorithm examines the jointly typical sequences between sender and receiver codebook to determine the fittest candidate among received codewords. There exists a fitness threshold such that codewords with values beneath the threshold value are excluded from consideration. The code is prefix-free; therefore, candidate codewords can be generated from recovered codewords before the entire code word is received.

3.2 Error correction capability

The matrix of cofactors, C is k-tuply redundant. For n=1, k=3, $C_{nk} = \{C_{11}, C_{12}, C_{13}, C_{13}, C_{11}, C_{12}, C_{12}, C_{13}, C_{11}\}$. Assume C_{nk} are i.i.d in individual packets. For small k, a majority voting scheme would provide a good error correction performance against random and burst errors at sub-optimal E_b/N_0 conditions at the receiver. Let P_e = probability of a bit error in a packet. For a majority voting receiver and k-tuple redundancy, total probability of error, P_{ε}, equals:

$$P_{\mathcal{E}} = \sum_{j=\frac{k+1}{2}}^{k} \binom{k}{j} (P_e)^j (1-P_e)^{k-j} \qquad (19)$$

Table 3. Comparison of 6 character block coding and 4 character block coding recovery

6 Char block sourced	6 Char block Recovered	4 char block sourced	4 char block recovered
86.96818292761	86.968182927609999999999999513	86.968182	86.968181999999999999910656396
88.761766889592	88.761766889591999999999999506	92.76188761	92.761887609999999999905496473
88.7667638876777	88.766763887677699999999999499	766.889592	766.889591999999999999236230939
761.887617668895	761.887617668894999980259561 22	88.76676388	88.766763880000000000231521412
92.887618696761	92.887618696760999997608395 15	767.7667619	767.766761880000000001957806511
88.857628876186	88.857628876185999997715387 48	761.7668895	761.766889500000000001939766504
92.8876475892762	92.887647589276199997492579 24	92.8876186	92.887618600000000000098708718
761.857667678876	761.857667678875999979741578 64	96.7618885	96.761888500000000000107561576
86.927618692763	86.927618692762999997692170 57	762.8876186	762.887618600000000000825690688
88.8896969696	88.889696969599999999999969 20	92.88764758	92.887647579999999999943083348
96.9696969696	96.969696969599999999999966 47	92.76276185	92.762761849999999999942229210
96.9696969696	96.969696969599999999999966 47	766.7678876	766.767887559999999999513732920
		86.9276186	86.927618599999999999999995707
		92.7638888	92.763888799999999999999995480
		96.969696	96.969695999999999999999995438

In the case of 6 characters coded per block at the source, 12 coefficients are transmitted and the pre-corrected recovered text was: "hamlet to be or n77t to be that is the question 76hether" In the case of the 4 characters coded per block at the source, 15 coefficients are transmitted and the pre-corrected recovered text was: "hamlet to be or not to be that is the question whether". The longer the block at the source, the fewer number of coefficients are required to be transmitted, at the cost of greater error correction at the receiver. For 8 characters per block at the source, only 9 coefficients are required to code the block, but the number of error positions in the message requiring correction at the receiver increased to seven.

4. CONCLUSIONS

A source coding protocol has been presented for the generation of genomic code representations suitable for later encryption by gene expression encryption protocols. Such a protocol ingests plaintext and outputs into a genomic DNA code sequence with code lengths based upon the composition of a user selected source gene. Subsequent encryption directly

converts the raw DNA output of the protocol into gene sequences with properly coded control regions for subsequent transcription and translation. This source coding scheme can be used by any application which converts plaintext input to a genomic sequence for transmission and then recovers the plaintext from the genomic sequence at a receiver.

5. ACKNOWLEDGMENTS

Thanks to NASA HQ Space Communications and Navigations division and the NASA/Goddard Space Flight Center Space Network project. This paper is developed from a dissertation submitted to The George Washington University in partial fulfillment of the requirements for the Ph.D. degree.

6. REFERENCES

[1] H. Shaw and S. Hussein, "A DNA-Inspired Encryption Methodology for Secure, Mobile Ad-Hoc Networks (MANET), Proceedings of the First International Conference on Biomedical Electronics and Devices, BIOSIGNALS 2008, Funchal, Madeira, Portugal, vol. 2, pp. 472-477, January 28-31, 2008

[2] A. Gehani, T. LaBean, and J. Reif, "DNA-based Cryptography, Aspects of Molecular Computing", Springer-Verlag Lecture Notes in Computer Science, vol. 2950, pp. 167-188, 2004.

[3] H. Shaw, S. Hussein, and H. Helgert, "Genomics-based Security Protocols: From Plaintext to Cipherprotein", International Journal on Advances in Security, vol. 4 no 1 & 2, 2011

[4] M. Ehrlich and R.Y. Wang, "5-Methylcytosine in eukaryotic DNA", Science 19 June 1981: Vol. 212 no. 4501 pp. 1350-1357

[5] J. M. Berg, J. L. Tymoczko, and L. Stryer, Biochemistry. 5th edition (online ed,), New York: W H Freeman; 2002, sec. 27.6.1

[6] H. Singh, K. Chugh, H. Dhaka and A. K. Verma, "DNA based Cryptography: an Approach to Secure Mobile Networks", International Journal of Computer Applications 1(1):77–80, February 2010.

[7] P. Vijayakumar, V. Vijayalakshmi and G. Zayaraz. "DNA Computing based Elliptic Curve Cryptography", International Journal of Computer Applications 36(4):18-21, December 2011

[8] S. Sadeg, M. Gougache, N. Mansouri, and H. Drias, "An encryption algorithm inspired from DNA", Machine and Web Intelligence (ICMWI), 2010 International Conference on, 3-5 Oct. 2010, pp. 344 - 349

[9] N.G. Bourbakis, "Image Data Compression-Encryption Using G-Scan Patterns", Systems, Man, and Cybernetics, IEEE International Conference on Computational Cybernetics and Simulation, vol. 2, pp. 1117—1120, October 1997

[10] A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with DNA binary strands", BioSystems, vol. 57, issue 1, pp. 13-22, June 2000

[11] C.T. Clelland, V. Risca, and C. Bancroft, "Hiding Messages in DNA microdots", Nature, vol. 399, pp. 533—534, June 1999

[12] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm", BMC Bioinformatics, vol. 8, pp. 176, May 2007

[13] D. Heider and A. Barnekow, "DNA watermarks: A proof of concept", BMC Molecular Biology 2008, vol. 9, p, 40

[14] T. M. Cover and J. A. Thomas, 2006, Elements of Information Theory, 2nd Ed., Wiley Interscience

[15] C. M. Fraser, J. D. Gocayne, O. White, M.D Adams, R. A. Clayton, R. D. Fleischmann, et al., "The Minimal Gene Complement of *Mycoplasma genitalium*", Science, vol. 270, No. 5235, pp. 397-403, Oct. 20, 1995