Joint Video Compression and Encryption using Arithmetic Coding and Chaos

Amit Pande
Department of Computer Science,
University of California,
Davis, CA, USA
Email: amit@cs.ucdavis.edu

Joseph Zambreno
Electrical and Computer Engineering,
Iowa State University,
Ames, IA, USA
Email: zambreno@iastate.edu

Prasant Mohapatra
Department of Computer Science,
University of California,
Davis, CA, USA
Email: prasant@cs.ucdavis.edu

***Abstract*—Joint Video Compression and Encryption (JVCE) has gained increased attention in the past couple of years to reduce the computational complexity of video compression, as well as provide encryption of multimedia content for web services. In this paper, we present a JVCE framework based on Binary Arithmetic Coding (BAC). We first present an interpretation of BAC in terms of a skewed binary map and then describe 7 other possible chaotic maps which give similar Shannon optimal performance as BAC. We then propose a modification of BAC in which the overall length within the range [0,1) allocated to each symbol is preserved, but the choice of map used to encode each symbol is based on a key. The encoder, referred to as Chaotic Binary Arithmetic Coder (CBAC), has the effect of scrambling the intervals without making any changes to the width of interval in which the codeword must lie, thereby allowing encryption without sacrificing any coding efficiency. We also present some some security enhancement features to show how they can alleviate the limitations of our technique against known cryptanalysis on BAC-based encryption schemes.**

***Index Terms*—Chaos, Encryption, Skewed Tent Map, Arithmetic Coding, Cryptography, Data compression.**

I. Introduction

Joint Video Compression and Encryption (JVCE) has gained increased attention in the past couple of years to reduce the computational complexity of video compression, as well as provide encryption of multimedia content for web services.

The video encryption algorithms must efficiently cater to the characteristics of video bitstream which are different from traditional digital data stream: Video content is larger in size and structured in different ways to enable different applications and network communications. Moreover, multimedia web services (such as video-on-demand, multimedia messages, and video conferencing) over heterogeneous architectures such as mobile phones and laptops need scalable media transmission schemes which have low decryption and re-encryption cost overhead, shorter delays and QoS requirements (such as real-time playback and low delay). Future video search engines must be able to search for a particular person or object from secure video libraries. The increased trend towards embedded devices require such algorithms to have low computational overhead.

It is difficult to cater the multiple requirements of low computational overhead, high security levels and other needs of compressed multimedia bitstreams [1] by the traditional compress-then-encrypt paradigm. Generic encryption algorithms are thus disadvantageous for multimedia security. Selective encryption schemes have been developed but most of them have proven to be weak against cryptanalysis [2], [3]. JVCE schemes come to the rescue by providing encryption at low overheads and without changing the structure of multimedia content. In this paper, a JVCE scheme based on Arithmetic Coding is presented and security enhancements are proposed to make this scheme secure against cryptanalysis.

Arithmetic coding [4] involves associating a sequence of symbols with a position in the range [0,1), and is usually implemented recursively. It typically enables very high coding efficiency as multiple symbols are coded jointly. It has been adopted for use in image compression standards, including JPEG2000 and H.264 to provide lossless entropy coding.

Owing to large volumes of multimedia data and useful properties of compressed bitstream such as rate-adaptive transmission, scalability, DC-image extraction for content searching etc generic encryption algorithms such as AES, DES etc have disadvantages for multimedia security. Many multimedia-specific algorithms have been proposed in research literature.

Recently some efforts have been made towards joint design of encryption and compression modules (particularly the entropy coding techniques such as arithmetic coding) to provide a more flexible encryption of data. These techniques allow encryption at little/ no computational overhead and (in most cases) preserve the format compatibility of compressed bitstream. For example- Liu et al. [5] presented a system using table-based bit sequence substitutions to enable the arithmetic coding stage to be simultaneously used for encryption. The authors in [6], [7] associate a fixed length index to each variable length codeword to encrypt the indexes. However, all these approaches suffer from compression inefficiency while the latter also leads to generation of emulated markers.

In [8], a chaos-based adaptive arithmetic coding technique was proposed. The arithmetic coder's statistical model is made varying in nature according to a pseudo-random bitstream generated by coupled chaotic systems. Many other techniques based on varying the statistical model of entropy coders have been proposed in literature, however these techniques suffer from losses in compression efficiency that result from changes in entropy model statistics and are weak against

978-1-4244-7932-0/10/$26.00

known attacks [9].

Recently, Grangetto et al. [10] presented a randomized arithmetic coding (RAC) scheme which achieves encryption by inserting some randomization in the arithmetic coding procedure at no expense in terms of coding efficiency. RAC needs a key of length 1-bit per encoded symbol. Wen and Kim et al. [11], [12] presented a generalization of this procedure, called as key-Splitting Arithmetic Coding (KSAC) where a key is used to split the intervals of an arithmetic coder. Some limitations and features of KSAC are presented next giving the motivation to improve the security performance of arithmetic coding:

1) KSAC introduces losses in coding efficiency which are later restricted to a small value by putting some constraints on the keyspace [11].
2) The KSAC encoder may have to work with multiple subintervals and needs to compute one shortest representation arithmetic code for each subinterval, thereby increasing the computational cost of encoder significantly.
3) The memory requirements of KSAC coder are atleast double that of BAC.
4) Although the key length can be arbitrary, the authors restrict it to 2 bits per encoded symbol [11].
5) Successful attacks have been demonstrated against KSAC scheme [9].

This motivates us to present an arithmetic encoder which can encode data without any compression inefficiency and promises better security than existing schemes.

In this paper we first present arithmetic coding as a chaotic iteration on skewed binary map and then use this model to propose a more robust Chaotic Binary Arithmetic Coder (CBAC) scheme which is similar to RAC in that it incurs no computational or memory overhead but provides more possibilities of interval ordering (leading to keys of 3 bits per symbol). A security enhancement is proposed to counter the known attack against arithmetic coding based schemes.

II. Arithmetic Coding as a Non-Linear Chaotic System

The work by Nagraj et al. [13] derives the equivalence of arithmetic coding and chaotic piece-wise linear maps (which they refer to as GLS coding). They develop a theory for skewed maps (skewed with a non-linearization parameter a to be used for image encryption) to be used for encryption and compression purposes. The above approach has two main disadvantages, making the scheme prone to cryptographic attacks:

1) A wrong guess in value of the skew parameter a may lead to imperfect reconstruction and not necessarily to completely random output. The first few symbols of binary string may be correctly guessed by a wrong, but closely related value of a.
2) It is possible to iteratively guess the value of a by launching known plaintext attack. The closer the value of a gets to the original a value, the more symbols will be reconstructed properly.

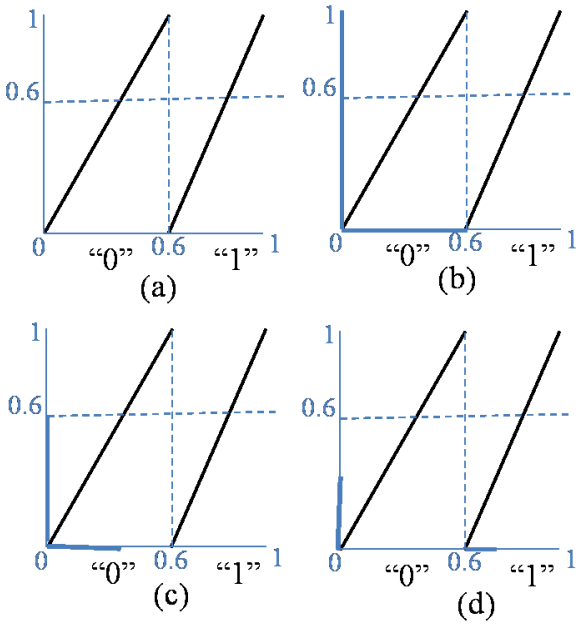

Fig. 1. (a) The binary map with p=0.6 (probability for symbol '0' is 0.6), Back-iterating the chaotic map to get the code word for encoding '100' with this map. (b) mapping the symbol '0', (c) mapping '00' and (d) mapping '100' to get the code-interval

The equivalence of chaotic maps to arithmetic coding is explained briefly with an example.

Consider encrypting the binary message '100' of length 3 bits. For binary arithmetic coding, let p (probability for occurance of symbol '0') be preselected to 0.6. For arithmetic coding, we will choose the interval [0.6,1) in first iteration to encode '1', [0.6, 0.84) to encode the '10' and [0.6, 0.744) to encode the '100'. A number lying in the interval [0.6,0.744) represents the arithmetic code for binary message '100'.

Now, let us define the skewed binary map with the following equations:

$$y = \left\{ \begin{array}{c} x/p \text{ when } x \leq p \\ (x-p)/(1-p) \text{ when } x > p \end{array} \right\} \quad (1)$$

Iteration on this chaotic map (Figure II(a)) is equivalent to decoding the arithmetic code. The symbol decoded is defined as follows:

$$\text{Decode} \left\{ \begin{array}{c} \text{'0' when } x \leq p \\ \text{'1' when } x > p \end{array} \right\} \quad (2)$$

Back iteration on this map is equivalent to the arithmetic encoding procedure, defined by the following equation:

$$x = \left\{ \begin{array}{c} py \text{ when '0'} \\ (1-p)y + p \text{ when '1'} \end{array} \right\} \quad (3)$$

To encode '100' with the p value precomputed to 0.6. we substitute the p value to get the chaotic map as shown in Figure II(a). In this case, however the encoding is done in reverse order: from end of sequence to the beginning. We first encode '0' and get the interval [0,0.6) (see figure II(b)), next

we encode '00' to get the interval [0,0.36) (see figure II(c)) and finally we encode '1' to get the interval corresponding to '100' as [0.6, 0.744). So we have reached the same interval as the arithmetic code.

The decoding part is also simple. Let us say that the transmitted code was 0.625. Iteration on the chaotic map as in Equation 1 gives the values [0.625, 0.0625, 0.1041] on two iterations. Thresholding it using the rule in Equation 2 gives us the original sequence [1 0 0] or '100'.

III. Chaotic Binary Arithmetic Coder

In the previous section we explained how arithmetic coding can be viewed as re-iteration on skewed binary map. There are, however, eight equivalent modes of skewed binary maps which can be used for iteration. They are shown in Figure 2. These modes differ from each other in the way input is mapped into the chaotic orbit. The maps differ in the interval in which the arithmetic code must lie for a symbol '0' or '1' but the width of interval remains the same. Hence, all the modes are equivalent as far as compression efficiency is achieved.

Let us define the generalized skewed binary map with the following equations:

$$y = \left\{ \begin{array}{l} n_1 x + c_1 \text{ when } x \leq k \\ n_2 x + c_2 \text{ when } x > k \end{array} \right\} \tag{4}$$

$$\text{Decode} \left\{ \begin{array}{l} \text{'0' when } x \in [i1, i2] \\ \text{'1' when } x \in [i3, i4] \end{array} \right\} \tag{5}$$

Then, the back iteration on skewed binary map is defined by the following equations:

$$x = \left\{ \begin{array}{l} m_1 y + b_1 \text{ when '0'} \\ m_2 y + b_2 \text{ when '1'} \end{array} \right\} \tag{6}$$

where m,n,b and c values determine the slope of chaotic map. These values can be precomputed to a tabular form for direct lookup at time of implementation as shown in the table. $n_1 = N1(i)$, $n_2 = N2(i)$, $c_1 = C1(i)$, $c_2 = C2(i)$, $m_1 = M1(i)$, $m_2 = M2(i)$, $b_1 = B1(i)$, $b_2 = B2(i)$, $m_1 = M1(i)$, $m_2 = M2(i)$, $i_1 = I1(i)$, $i_2 = I2(i)$, $i_3 = I3(i)$, and $i_4 = I4(i)$ and i varies from 1 to 8 depending on the choice of chaotic map. Table I gives the value of these parameters for all eight chaotic maps.

The generalization above includes the BAC and RAC schemes. An encoding/ decoding procedure using only $i = 1$ is equivalent to the BAC. If we only use two modes $i = 1$ and $i = 5$ instead of 8, the equivalent system is similar to the RAC scheme. KSAC can also be visualized in terms of chaotic maps by splitting the '0' interval in Figure 3(a) by a key into two and inserting the split to the right of the map.

A. Implementation Efficiency

For a normal binary arithmetic coder, at each iteration the starting interval $[I_s, I_e)$ is updated at one end. On encoding a '0' the final interval becomes $[I_s + p(I_e - I_s), I_e)$ while on encoding a '1' the final interval becomes $[I_s, I_s + p(I_e - I_s)$. Thus, every iteration requires one multiplication and two addition operations. The decoding procedure for a binary arithmetic coder involves updating the interval $[I_s, I_e)$ at one end depending on whether the last decoded symbol was a '0' or a '1'. Thus, every iteration again requires one multiplication and two addition operations.

For chaotic arithmetic encoder, both end of interval are updated at every iteration using a linear transformation $x = my+c$ thus requiring two multiplications and two additions for encoding. The decoding is simple as it involves iteration on the chaotic map according to the linear transformation $y = nx+c$ involving a multiplication and an addition operation. There are some additional table lookups involved in chaotic coding to choose the right chaotic map at every iteration which can be efficiently implemented in software/ hardware. Thus, CBAC encode requires more computations than BAC encode while CBAC decode requires less computations than BAC decode.

All eight maps give the same width of final interval for the code word to lie, the interval itself being different for different maps. For example: '01100' will be given the interval $[0.504, 0.53856)$ using map (a), $[0.36, 0.39456)$ using map (b), $[0.1824, 0.21696)$ using map (c), $[0.0384, 0.07296)$ using map (d), $[0.46144, 0.496)$ using map (e), $[0.92704, 0.9616)$ using map (f), $[0.78304, 0.8176)$ using map (g) and $[0.60544, 0.64)$ using map (h).

B. Applications

The CBAC can be used as a joint compression-cum-encryption technique for data encryption. It is particularly beneficial for data-intensive tasks such as multimedia encryption and compression and can be integrated into the standard video compression algorithms such as JPEG2000, JPEG, MPEG etc.

CBAC can be used for full or selective encryption of multimedia data. For full encryption, the entire volume of multimedia data is passed through CBAC encoder while in case of selective encryption only the important parts of data are passed through CBAC encoder. If we reveal the first K bits of the key publicly, then a part of the bitstream can be decoded correctly while decoding the entire bitstream will require knowledge of the entire key. Thus, CBAC can be used to provide conditional access to the multimedia content.

C. Secure Multicast

The CBAC schemes allows a given input string to be decoded by more than one key for correct reconstruction. This is attributed to the fact, that for a given bit - two of the eight maps will give the same output interval. For example - encoding '0' with either of the maps given in Figure 2(a) and Figure 2(b) will lead to same result and encoding '1' with maps given in Figure 2(a) and Figure 2(d) will give the same result.

Thus, we can infer that "*Of the 8 possible maps, two maps will give the same reconstruction for a single bit but the choice of bit depends on its value being '0' or '1'.*" For a N bit message, we have a pool of 2^N choices out of the total keyspace of 2^{3N}.

This property can potentially be used for secure multicast of videos. We illustrate this with the example of a three bit

TABLE I
PARAMETER LIST FOR THE EIGHT POSSIBLE CHOICES OF CHAOTIC ENCODER

Parameter	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
M1	p	p	$-p$	$-p$	p	$-p$	$-p$	p
B1	0	0	p	p	$1-p$	1	1	$1-p$
M2	$1-p$	$p-1$	$p-1$	$1-p$	$1-p$	$1-p$	$p-1$	$p-1$
B2	p	1	1	p	0	0	$1-p$	$1-p$
N1	$1/p$	$1/p$	$-1/p$	$-1/p$	$1/(1-p)$	$1/(1-p)$	$-1/(1-p)$	$-1/(1-p)$
C1	0	0	1	1	0	0	1	1
N2	$1/(1-p)$	$-1/(1-p)$	$-1/(1-p)$	$1/(1-p)$	$1/p$	$-1/p$	$-1/p$	$1/p$
C2	$-p/(1-p)$	$1/(1-p)$	$1/(1-p)$	$-p/(1-p)$	$(p-1)/p$	$1/p$	$1/p$	$(p-1)/p$
I1	0	0	0	0	$(1-p)$	$(1-p)$	$(1-p)$	$(1-p)$
I2	p	p	p	p	1	1	1	1
I3	p	p	p	p	0	0	0	0
I4	1	1	1	1	$1-p$	$1-p$	$1-p$	$1-p$
K	p	p	p	p	$1-p$	$1-p$	$1-p$	$1-p$

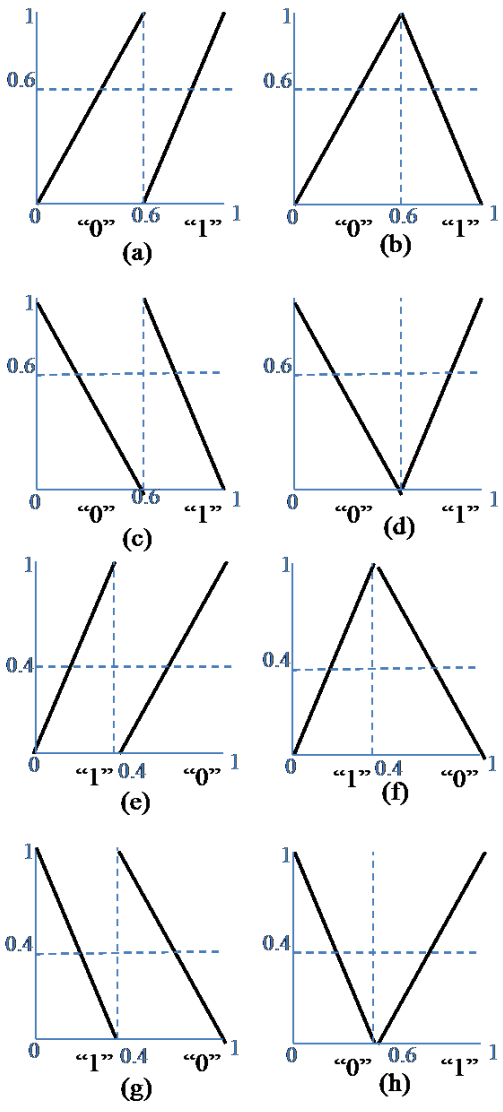

Fig. 2. (a-h) show the eight modes of the skewed binary map (p=0.6).

message '001' which is encoded with the maps '1','3' and '6' or simply '136'. Different users can download the same encoded message and decrypt them on their machine using different keys chosen from the pool of eight keys - '245','246', '235', '236', '135', '136', '145' and '146'. Thus, the same compressed (and encrypted) data can be sent to all the users in the multicast, while a unique key can be sent to each user. In this example (of three bits compression), the key space is $8^3 = 512$, of which only 8 keys are correct.

For a string of length 128, we have 2^{128} correct decryption keys in a keyspace of 2^{384}. Thus, the probability of guessing a right key is 2^{-256} and a exhaustive search of keyspace will not be helpful. On the other hand, we can distribute the same compressed data to different users (upto 2_{128}). Any illegal or unauthorized distribution of videos by the users can be tracked back to them by identifying the keys.

D. Security

The following issues have greatly reduced the efficiency of RAC and KSAC schemes.

1) The RAC scheme has been known to be insecure against cryptanalysis using known-plaintext attacks. If an input corresponding to an all-zero sequence is revealed in RAC, an attacker then knows all information of intervals [12].
2) RAC needs 1 pseudo-random bit (or key bit) per 1-bit of input data which is inefficient as compared to the use of standard stream ciphers to encrypt the compressed stream (need 1 pseudo-random bit per 1-bit of compressed data). The authors in KSAC need multiple bits per 1-bit of input data but they propose reuse of keys and restrict the split locations per 1-input bit to 4 requiring only 2-bit key information per 1-bit of input data. However, this reduces the secrecy of the scheme. The authors in [9] present a cryptanalysis of this scheme where they reveal that a key of length 2000 bits can be broken with as few as 50000 plaintexts.

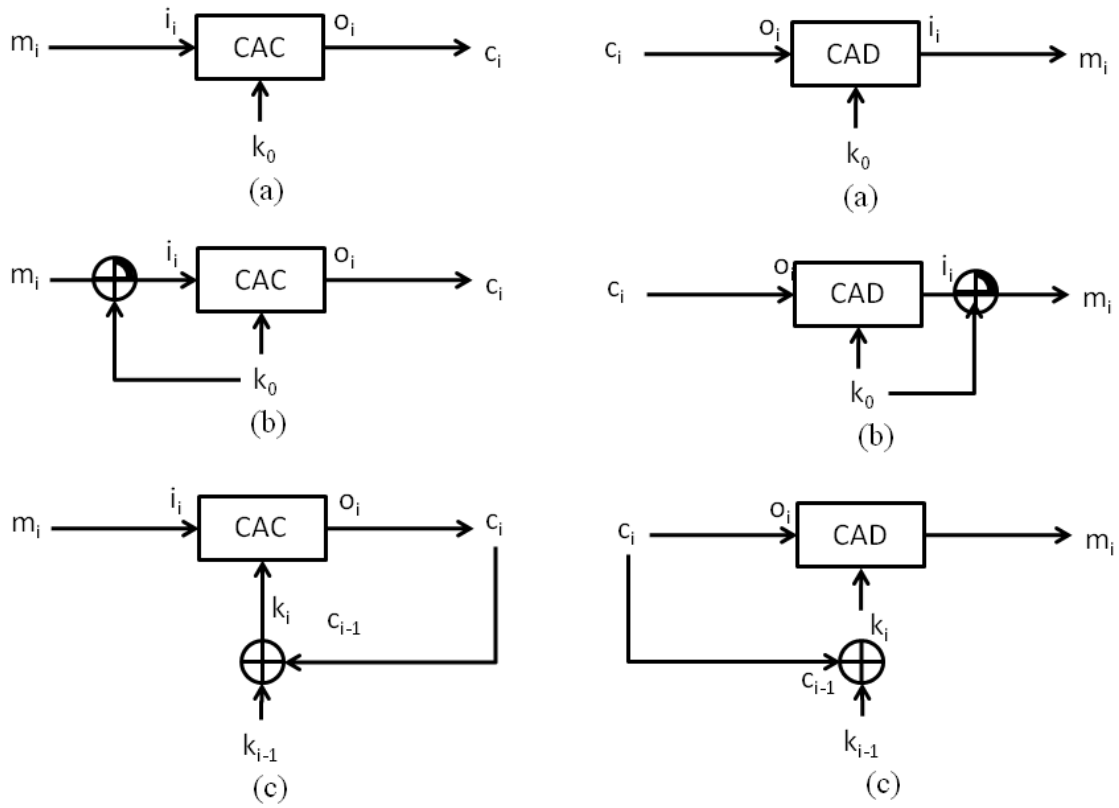

Fig. 3. Block diagram of (a) Chaotic Binary Arithmetic Coder (CBAC), (b) CBAC with Plaintext Modification (CBAC+PM), (c) CBAC with Key Update (CBAC+KU) (d) Chaotic Arithmetic Decoder (CAD), (b) CAD with Plaintext Modification reverted (CAD+PM), (c) CAD with Key Update (CBAC+KU)

These restrictions apply to CBAC scheme also. We next present a simple scheme to alleviate both the above mentioned limitations. Since more than one key value can refer to the same interval length, it is not possible to reverse engineer the key from known-plaintext or chosen plaintext based attacks.

We XOR the first few P bits (say P=10) of sequence or message with the first P bits of the key. This is referred to as Plaintext Modification mode (CBAC+PM). This modification may lead to slight decrease in coding efficiency but coding loss can be reduced significantly by choosing $N >> P$. This is shown in Figure 3(c) and the corresponding changes in decoder are shown in Figure 3(d).

We propose another modification, called as Key Update (CBAC+KU) where we update the keys of the CBAC encoder every iteration by XORing the output bits to the key. The CBAC encoder encodes input messages (of length N bits each) using a key of length $3M$ ($3M < N$) such that only the first $M-1$ symbols are encoded with a unique key. The last three bits of key are used to encode the remaining (N-M+1) bits of the message. We XOR the key with first 3M bits of encoded output. This leads to use of different key for every iteration without the need of any pseudo-random generator. In case, the encoded output length is less than $3M$ bits, we XOR the last bit to the extra bits of key. The problem of choosing the first value of encoded output (to be XORed to the key) is resolved by using all zeros for first run of CBAC coder. Thus, the input key is $3M$ bits long. For N=1000, M=100 we have effectively a 300 bits long key space. These modifications shown in Figure 3(e)-(f) greatly enhances the security performance against known attacks without compromising the coding efficiency.

These modifications address the two limitations of BAC-based encryption systems. They can be used alone or both can be applied together to a system (CBAC+PM+KU).

E. Compression

The performance of CBAC coder is also believed to be the same as the normal arithmetic coder because the interval width for arithmetic code and our scheme are supposed to be the same. We ran an implementation of chaotic binary arithmetic coder over Matlab 7.8.0 (R2009a) and used variable precision arithmetic (vpa) tools in the Symbolic Mathematics Toolbox to run simulations for N=100,1000 etc. The simulation results show similar compression performance for CBAC and Binary arithmetic coder (BAC) and are presented in Table II. For the simulations, we generated random bitstreams of variable lengths (N=10, 100 or 1000) and variable probability values (p-3/5, 6/7 and 10/11). The mean length of the output bitstream is reported in the tables. It must be noted that although, in an individual case, the outputs of CBAC and BAC coders may be different, the average over a last sample must converge to same value.

TABLE II
PERFORMANCE OF CBAC FOR VARIOUS LENGTH STRINGS

	$N = 10$		$N = 100$		$N = 1000$	
	BAC	CBAC	BAC	CBAC	BAC	CBAC
$p = 3/5$	8.7876	8.749	96.016	96.563	970.09	969.39
$p = 6/7$	5.3252	5.2222	57.779	57.576	593.17	593.17
$p = 10/11$	4.182	4.18	42.55	42.54	593.17	593.16

TABLE III
COMPRESSION EFFICIENCY OF CBAC WITH SECURITY ENHANCEMENTS

	BAC	CBAC	CBAC+KU	CBAC+PM
$p = 3/5$	970.30 ± 8.07	969.84 ± 8.25	969.72 ± 8.34	978.2 ± 9.02
$p = 5/6$	647.60 ± 30.69	647.58 ± 30.65	647.48 ± 30.91	650.51 ± 31.9
$p = 10/11$	439.99 ± 32.66	439.5 ± 32.74	439.51 ± 32.6	452.1 ± 33.01

The results in Table III indicate that although BAC, CBAC and CBAC+KU have similar compression performance while CBAC+PM leads to some reduction in compression efficiency. The experiments were run for string of length N=1000, and averaged over 1000 simulations. The results for CBAC+PM are expected to be the same as CBAC or BAC because over large number of samples, the effect of key changes will be averaged out.

For CBAC+PM, the loss in compression is significant even for small values of P (P=10). This is attributed to the modification in first P bits of message (with probability p) with P bits of key ($p_{new} \approx 0.5$). The first P values have a significant impact in the width of intervals, hence a significant compression loss is observed.

IV. CONCLUSIONS

This paper presents a JVCE scheme which reduces the computational costs involved with traditional approach of encryption after compression. The proposed CBAC scheme uses an interpretation of Arithmetic Coding using chaotic camps. The compressed data itself is not encrypted, making it easier to preserve properties of video data for indexing, search, network communications and other operations. The proposed security enhancements allow us to build a video encryption scheme resistant to known attacks.

The compression performance of CBAC scheme and the security enhancement with key update (CBAC+KU) is the same as Binary Arithmetic Coder. The simple CBAC approach should be able to use context-adaptive models for Chaotic Binary Arithmetic Coder to replace the Context-Adaptive Binary Arithmetic Coders (CABAC) coders used in H.264/MPEG-4 video coders. CABAC coder has high computational costs which makes it available only for main and higher profiles. CBAC scheme has higher computational costs on encoder side but reduced computational requirements on decoder (than BAC) with added security feature, which makes it favorable for server-client video distribution model which require easy decoding on client side and extended computations on server side.

The secure multicast idea presented in this paper can be further developed to be deployed in modern systems. This approach is just the beginning of the efforts to build a non-conventional cryptosystem for internet multimedia applications by generating a keyspace from compression parameters instead of processing the compressed bitstream with the goal of *low-cost, low-overhead, property preserving* encryption of videos. One direction for future work is to investigate the vulnerability of proposed scheme against the attacks not discussed above and develop efficient solutions for the same.

REFERENCES

[1] Y. Mao and M. Wu, "A joint signal processing and cryptographic approach to multimedia encryption," *IEEE Trans. Image Processing,*, vol. 15, no. 7, pp. 2061–2075, July 2006.

[2] C.-P. Wu and C.-C. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 828–839, Oct. 2005.

[3] "A survey of video encryption algorithms," *Computers and Security*, vol. In Press, Corrected Proof, 2009.

[4] G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Communications*, vol. 29, no. 6, pp. 858–867, Jun 1981.

[5] X. Liu, P. G. Farrell, and C. Boyd, "A unified code," in *IMA Int. Conf. Cryptography and Coding*, 1999, pp. 84–93.

[6] M. L. Jiangtao Wen and M. Severa, "Access control of standard video bitstreams," in *Proc. IEEE Intl. Conf. Media Future*, 2001.

[7] M. Wu and Y. Mao, "Communication-friendly encryption of multimedia," in *IEEE Workshop on Multimedia Signal Processing*, 2002, pp. 292–295.

[8] R. Bose and S. Pathak, "A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system," *IEEE Trans. Circuits and Systems I*, vol. 53, no. 4, pp. 848–857, April 2006.

[9] G. Jakimoski and K. Subbalakshmi, "Cryptanalysis of some multimedia encryption schemes," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 330–338, April 2008.

[10] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 905–917, Oct. 2006.

[11] J. Wen, H. Kim, and J. Villasenor, "Binary arithmetic coding with key-based interval splitting," *IEEE Trans. Signal Processing Letters*, vol. 13, no. 2, pp. 69–72, Feb. 2006.

[12] H. Kim, J. Wen, and J. Villasenor, "Secure arithmetic coding," *IEEE Trans. Signal Processing*, vol. 55, no. 5, pp. 2263–2272, May 2007.

[13] N. Nagaraj and P. G. Vaidya, "One-time pad, arithmetic coding and logic gates: An unifying theme using dynamical systems," *CoRR*, vol. abs/0803.0046, 2008.