1-1-2000

Secure compression using adaptive Huffman coding

C. Kailasananathan
University of Wollongong

Reihaneh Safavi-Naini
University of Wollongong, rei@uow.edu.au

Philip Ogunbona
University of Wollongong, philipo@uow.edu.au

Secure compression using adaptive Huffman coding

Abstract

Recent developments in the Internet and Web based technologies require faster communication of multimedia data in a secure form. Standard compression algorithms such arithmetic coding schemes, and propose methods of protecting against these attacks. In the next section we review DHC, and in Section 3 describe DHC encryption scheme and examine possible attacks. In Section 4, we propose an encryption scheme that protects against the attacks and in Section 5 give the results of our experiments. Section 6, concludes the paper. as JPEG and MPEG use an entropy coding stage. By incorporating security in this stage it it possible to add security to compression systems at very low cost. In this paper we describe a combined compression and encryption scheme using Dynamic Huffman Coding. We examine se- 2 curity of the system and show how to make it resistant against known attacks. We also give the result of incorporating this system into JPEG.

Keywords

coding, huffman, compression, adaptive, secure

Disciplines

Physical Sciences and Mathematics

Publication Details

Kailasananathan, C., Safavi-Naini, R. & Ogunbona, P. (2000). Secure compression using adaptive Huffman coding. In L. Guan, R. Liu & P. Beadle (Eds.), IEEE.PCM2000 Conference Proceedings. 2000 1st Pacific-Rim Conference on Multimedia (pp. 336-339). Piscataway, NJ, USA: IEEE - Institute of Electrical and Electronics Engineers Inc..

Secure Compression using Adaptive Huffman Coding

C.Kailasanathan , R.Safavi Naini and P.Ogunbona
Center for Computer Security Research, School of IT and Computer Science,
Wollongong University, NSW2522, Australia. e-mail: ck12@uow.edu.au

July 21, 2000

Abstract

Recent developments in the Internet and Web based technologies require faster communication of multimedia data in a secure form. Standard compression algorithms such as JPEG and MPEG use an entropy coding stage. By incorporating security in this stage it it possible to add security to compression systems at very low cost. In this paper we describe a combined compression and encryption scheme using Dynamic Huffman Coding. We examine security of the system and show how to make it resistant against known attacks. We also give the result of incorporating this system into JPEG.

1 Introduction

Adding encryption to compression algorithms is an attractive proposition which could result in combined security and compression and so more efficiency. The main challenge is to design secure systems without degrading compression performance.

Entropy coding algorithms such as Huffman coding [1] and arithmetic coding [2] are not only used for lossless compression but also used as part of lossy compression algorithms such as JPEG (Joint Photographic Experts Group) [13] and so adding security to them will result in securing the larger systems too. Witten and Cleary [11] proposed a method of adding encryption to arithmetic coding. However a number of attacks were published against their proposed systems [9, 12] and their extensions, which clearly showed designing secure compression systems is not an easy task.

In this paper we propose a secure compression system based on Dynamic Huffman Coding. *Huffman coding* is an optimal compression algorithm that in its simplest form uses probabilities of characters in the input sequence as a *model* for the input source, and produces codewords whose lengths are inversely proportional to their probabilities. In *Dynamic Huffman Coding* (DHC) the model is updated and the codeword representing a symbol changes with its position in the input stream. Following Witten and Cleary approach, we will use the model as the key information and obtain a secure compression system.

We show that the resulting system is vulnerable to attacks which have been already proposed against secure arithmetic coding schemes, and propose methods of protecting against these attacks.

In the next section we review DHC, and in Section 3 describe DHC encryption scheme and examine possible attacks. In Section 4, we propose an encryption scheme that protects against the attacks and in Section 5 give the results of our experiments. Section 6, concludes the paper.

2 Dynamic Huffman Coding

Dynamic Huffman coding was independently proposed by Faller [3] and Gallager [4] and considerably improved by Knuth [5] and Vitter [6]. The algorithm uses a *code tree* which is a weighted binary tree capturing the statistics of the input stream and forming the coder's model. Each leaf of the tree corresponds to one alphabet symbol and the weight of the leaf represents the current frequency of that symbol in the input stream representing the weights of each alphabets. Two nodes with a common parent in a code tree are called *siblings*. The weight of an internal node is the sum of the weights of its siblings. Nodes are numbered in increasing order starting from the bottom left node to bottom right node, followed by the nodes in the layers above again from left to right, until the root node is reached. *A binary tree is said to have sibling property* if listing the nodes using the above ordering results in a non-decreasing sequence. Gallager [4] proved that a binary prefix code is a Huffman code if, and only if, the code tree has the sibling property. In a dynamic Huffman tree the structure will be maintained in such a way that sibling property is preserved.

To encode an input symbol, first the the current Huffman tree is used to generate the codeword and then the weight of the leaf corresponding to the incoming symbol is updated. This update will flow through the whole tree. If because of updating of the weights, the sibling property is violated a new round of tree updates that involves the exchange of the nodes and their corresponding subtrees, will be applied to reinstate the property. Details of the update can be found in [6].

In implementing the above algorithm, a *halving* process is introduced. Halving is invoked if the weight of the root node reaches W_{max} which is the maximum possible weight of a node. In this case the weight of each node will be replaced with half its original value. This ensures that the

tree structure is not changed and model statistics (probabilities) are also intact. For example if a 16-bit unsigned integer is used for storing W_{max}, weights up to $2^{16} = 65535$ can be accommodated.

3 Huffman Coding Encryption Schemes

Static Huffman uses a fixed codebook. To add security one can use the codebook as the key. However the resulting system will be vulnerable to simple chosen plaintext attack and hence will not be secure. Using Dynamic Huffman coding is more promising as the codebook is not fixed.

Adaptive Huffman coding encryption scheme is further motivated by the following observations [11].

1. Model for DHC with reasonably large alphabet set (for example, 8 bit per symbol) would be very large and so the key space would be enormous.

2. An adaptive model depends on the entire transmitted text and finding the key would require tracking the changes to the model through the whole text.

The system may work in one of the following two ways [11].

Scheme 1: The key is the initial tree which will be securely sent to the decoder. The secret information includes the shape of the tree and the set of weights attached to the nodes and leaves.

Scheme 2: The initial model is public. Encoder and decoder share a secret key sequence which is run through their encoders and updates the code tree. The output is suppressed. At the end of the string the tree has a structure and weights that is unknown for outsiders.

3.1 Security of the schemes

The first step in evaluating security of a system is to find the size of the key space. In DHC encryption schemes 1 and 2, the key is the initial tree and the secret pseudorandom string, respectively. In the former case the size of the key space is the number of all possible trees. Assuming there are n symbols in the alphabet, the total number of possible trees is obtained as follows. The number of binary trees with n leaves is $\frac{\binom{2n-1}{n-1}}{n}$, known as Catalan number [8]. The number of possible Huffman trees with n distinct symbols , that is the number of possible keys for Scheme 1, will be

$$n!\frac{\binom{2n-1}{n-1}}{n} = \frac{(2n-1)!}{(n-1)!}$$

The size of the key space for Scheme 2 is 2^ℓ where ℓ is the length of the secret string.

It is important to note that the size of the key space only gives an upperbound on the level of security. To have a realistic assessment of the security of the system possible attacks on the system must be considered.

3.2 Attacks on Huffman Coding

Ciphertext only attack on static Huffman codes with three symbols has been analysed by Mohtashemi [7]. Extending this analysis to higher number of symbols, for example 256 symbols, and dynamic models is a daunting task. Moreover such study will be mainly of theoretical interest because the attack model is very restricted and in practice the adversary is much more powerful. In the following we extend two attacks originally proposed on adaptive arithmetic coding encryption schemes to DHC encryption schemes.

Bergen and Hogans (BH) attack for adaptive Huffman

Bergen and Hogan [9] proposed an attack on adaptive arithmetic coding encryption scheme. The attack is chosen plaintext and allows the attacker to feed plaintexts of his choice to the encoder and modify the model into a known form. That is, it uses the adaptiveness of the system to direct into a known state. Once the model is known the attacker can decrypt the communication.

The attack scenario is as follows.

1. The system is a DHC encryption scheme 1 or 2.

2. Attacker can send any sequence of symbols to the encoder and obtain the output.

3. Attacker has a decoder.

The attack proceeds as follows.

1. The attacker sends a long string of a single symbol ϕ to cause halving. If the string is chosen long enough, with enough number of halving occurring ϕ eventually moves to the top right hand side of the tree and frequencies of all other symbols become 1. If more ϕ are sent so that another halving occurs the frequency of ϕ will become a constant after each halving. We refer to this state of the tree as *synchronised state.* In the synchronised state inputing each extra ϕ will produce a constant output of 1. At this stage the attacker does not know the positions of other symbols in the tree.

2. If the attacker sends a sequence of a second symbol and observes the output, because of the sibling property of the coder, the symbol will migrate towards the right top of the tree. If halving occurs during this process, already ordered symbols will not be affected. When the encoder starts emitting the first 01 the attacker will know that the second symbol has reached its final position in the tree. The attacker repeats the procedure for the third and fourth symbols until all the symbols in the tree are ordered.

The code tree will look as Figure 1. Now the model can be verified by decoding a message.

To protect against this attack a proposal by Bergen and Hogan can be adapted. The basic approach is to change

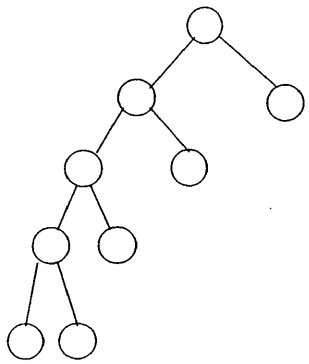

Figure 1: Code Tree

the maximum frequency, W_{max}, using a synchronous pseudorandom sequence in the encoder and decoder. Since halving occurs when $W_{root} = W_{max}$ the halving points become unpredictable. This randomization might reduce the compression rate and cannot prevent the model being modified. Lim, Boyd and Dawson [12] suggest regular resetting of the model during encoding a message which discards the model modified by an attacker.

Lim, Boyd and Dawson Attack

Bergen and Hogan attack does not find the key. Lim, Boyd and Dawson showed [12] a chosen plaintext attack on arithmetic coding encryption that discovers the key. The attack can also be extended to DHC encryption scheme to find the code tree. The basic idea is to send single character to the encoder and observe the output which in the case of DHC encryption reveals the position of the corresponding leaf. The encoder is reset to its initial state and the next symbol is sent to the encoder. In this way the position of all symbols and hence the structure of the code tree will be known however the weight of the nodes cannot be determined.

To protect against this attack, the system must not be resetable: that is the effect of the incoming symbols should not be removable from the model.

4 Securing Dynamic Huffman Encryption System

In the following we describe two methods of providing protection against BH attack, which is the main attack on DHC encryption system. The first method is masking the output such that the adversary cannot distinguish the state of the model by observing the output, and the second one is by making the system semi-adaptive.

1. Masking the output of an adaptive Huffman encoder

As noted earlier the basic approach in BH attack is to *flood* the model with a particular symbol and by observing the output to determine the state of the symbol in the code tree. To prevent the attack we can corrupt the observation data used by the adversary.

Two possible methods are i) *buffering* the output that removes the symbols boundaries and makes the input output relationship more complex, and ii) *permutation* that hides the repetitive sequences that mark synchronised state of the model.

Buffering masks the separation between each codeword and prevents the attacker from knowing the exact beginning/ending of a codeword. We assume a 1 byte buffer is used in the output and describe masking through an example.

When the model is flooded with a sequence of the same symbol, the system will ultimately produces a string of $a_1 = 1111111$. Knowing n, the number of symbols used to reach this stage, the attacker can send another sequence of length at most n until a sequence of $a_2 = 01010101$, is obtained. This process is continued until the attacker can synchronise the whole tree. For the third symbol, the buffer will be filled with multiple copies of 001 which depending on the starting point can produce one of the following blocks: $a_3 = 00100100, a_4 = 10010010, a_5 = 01001001$.

If we only consider the top 5 levels of the code tree, masking must be applied to $A = \{a_1, a_2 \cdots a_5\}$. A simple masking is to replace elements of A with another set of blocks $B = \{b_i, i = 1, \cdots 5\}$. This mapping will be secret and its inverse will be used in the decoder. In this case to avoid erroneous decoding, it is necessary to map back elements of B to elements of A. So, masking will consist of a hidden substitution of elements of A with B, and those of B with A. Let a and b denote the sizes of A and B respectively, and m denote the size of the alphabet set. Then B can be chosen in $\binom{m}{a}$ and the masking table may be defined in $(a!)^2 \times \binom{m}{a}$. However although masking can hide the exact form of an output block but cannot hide the repetitive nature of the output sequence, as it replaces a sequence of a_i with a sequence of b_js. This means that in this basic form masking cannot prevent detection of the synchronised state.

A possible solution is to use more than one substitution table and use a pseudorandom generator to alternate between them. To minimise the amount of storage, one can use a master table T and a number of other tables $t_1, \cdots t_u$ which are obtained by permuting the rows of T. If T has ℓ rows, then the tables can be chosen in $\binom{\ell!}{u}$ ways. Combined with the number of ways of constructing T, the result will be a huge key space which makes the exhaustive key search infeasible.

By increasing the length of the buffer and considering more levels of the trees the size of the key space can be further increased.

2. Semi-adaptive Time Variant Modeling

Barbir proposed [10] a semi-adaptive modeling approach for combined compression and encryption using arithmetic coders. The basic idea is to introduce some randomness into model probabilities. This is achieved by performing updates at random intervals specified by the

output of a pseudorandom sequence.

Using a similar approach in DHC encryption can prevent BH attack. The system works as follows. The encoder works as normal: it reads the input, produces an output and updates the code tree. After n_i input symbols where n_i is the output of a pseudorandom generator, let $f_1^i, \cdots f_m^i$ denote the frequencies of the m symbols in the input alphabet. At this stage a sequence of length n_i', where n_i' is determined by the pseudorandom generator, is used to update the model without producing any output. To reduce the effect of this randomisation on compression ratio, the number of the j^{th} symbol in the update string will be chosen proportional to f_j^i/n_i.

This ensures that the model will match the input data and the input probability distribution will not change and the compression performance is not affected.

5 Experiment

We encrypted the Lenna image using a JPEG implementation which uses adaptive Huffman coding for its entropy coding stage. Next the image was encrypted with the proposed DHC encryption scheme. We decoded the encrypted image without decrypting it (Figure 2(b)) and then decoded the encrypted image with a wrong key (Figure 2(c)). These experiments showed that compression ratios do not drop after adding encryption. Moreover using an incorrect key does not reveal any partial information.

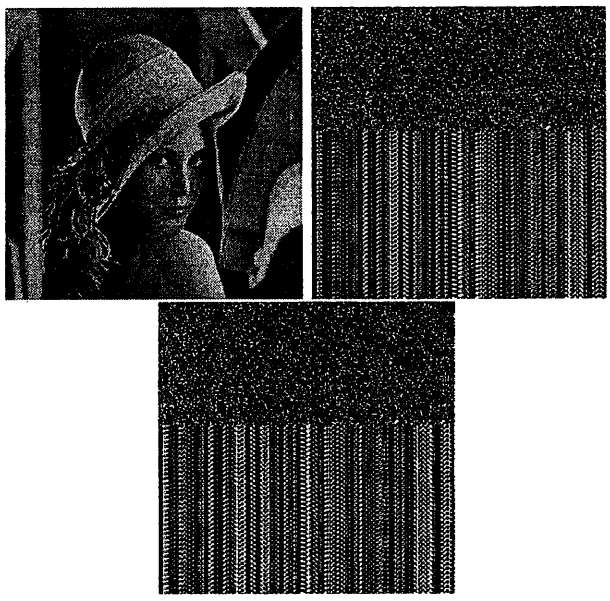

Figure 2: (a) Original Lenna (b) Lenna after encryption and (c) Lenna after decrypting with a wrong key

6 Conclusion

We showed that a model-based approach to combined compression and encryption using adaptive Huffman coding results in a system that is vulnerable to attacks known on arithmetic coding encryption schemes. We proposed methods of preventing against these attacks and gave the results of incorporating the system into JPEG compression.

References

[1] D.A.Huffman, A method for the construction of minimum redundancy codes in Proc. IRE, vol 40, no.9, pp. 1098-1101, 1952.

[2] R.Pasco. Source Coding Algorithms for Fast Data Compression. Ph.D. thesis, Stanford University, 1976. J.J.Rissanen and G.G.Langdon. Arithmetic Coding. IBM Journal of Research and Development, 23(2):149-162, March 1979.

[3] Faller.N. An adaptive system for data compression. In Record of the 7th Asilomar Conference on Circuit, Systems, and Computers. 1973,pp. 593-597.

[4] Gallager.R.G. Variation on a theme by Huffman. IEEE Tans. Info. Theory IT-24,6 (Nov. 1978),668-674

[5] Knuth.D.E. Dynamic Huffman coding. J. Algorithm 6(1985),163-180

[6] J.S.Vitter. Design and Analysis of Dynamic Huffman Codes. Journal of ACM,34(4):825-845, October 1987.

[7] M.Mohtashemi. On the Crypytanalysis of Huffman Codes, Masters thesis, Dept.of Computer Science, MIT. Institute of Technology

[8] Donald E.Knuth. The Art of Computer Programming: Sorting and Searching, Volume 3. Addison-Wesley, 1973.

[9] Helen A. Bergen and James M. Hogan. A chosen plaintext attack on an adaptive arithmetic coding compression algorithm. Computers and Security, 12:157-167, 1993.

[10] Abbie Barbir. A Methodology for Performing Secure Data Compression. System Theory 1997 Proceedings of the Twentininth Southeastern Symposium on pages 266-270, March 9-11, 1997.

[11] I.H.Witten and J.G.Cleary. On the privacy afforded by adaptive text compression. Computers and Security, 7:397-408, 1988.

[12] Jen Lim, Colin Boyd, and Ed Dawson. Cryptanalysis of adaptive arithmetic coding encryption scheme. ACISP, pages 216-227, 1997.

[13] Wallace,G.K., "The JPEG still picture compression standard.", Communication of the ACM, vol. 34, no. 4, April 1991, pp 30-40.