Correspondence

On Breaking a Huffman Code

David W. Gillman, Mojdeh Mohtashemi, and Ronald L. Rivest

Abstract—**We examine the problem of deciphering a file that has been Huffman coded, but not otherwise encrypted. We find that a Huffman code can be surprisingly difficult to cryptanalyze. We present a detailed analysis of the situation for a three-symbol source alphabet and present some results for general finite alphabets.**

Index Terms—**Huffman codes, cryptography, encoding rules, ambiguity, independent sources, Markov sources.**

I. Introduction

One of the earliest data-compression algorithms is due to D. Huffman [4]. Given the probabilities of each symbol of a source alphabet, this algorithm produces a variable-length binary code which achieves an optimal expected codeword length among codes whose codewords all have an integer number of bits.

In this correspondence we investigate the problem of cryptanalyzing a message that has been compressed using the Huffman algorithm, but not otherwise encrypted. The Huffman algorithm assigns to each source symbol a binary codeword. This assignment, or *encoding rule*, determines the *codeword set*, which is prefix-free and so corresponds to a full binary tree called the *Huffman tree*. Edges in the Huffman tree connecting an internal node with its left child are labeled 0, and edges connecting an internal node with its right child are labeled 1. The codeword associated with a source symbol is the binary string obtained by reading the bits on the unique path from the root of the tree to the leaf labeled with that source symbol. For our purposes, the encoding rule, the codeword set, and the Huffman tree are different names for the same thing. We assume familiarity with the way the Huffman algorithm constructs the tree from a set of source symbol probabilities (see [1] or [4] for an exposition).

The source produces a message called the *source stream*, and the encoded version of the source stream is called the *(binary) file*. We show that depending on what the cryptanalyst knows *a priori* about the source and the Huffman encoding, the file can be impossible to decode unambiguously. This means that for all the cryptanalyst knows the file could have been produced by a different source than the actual one. In the absence of more knowledge about the source, the best that can be hoped for is a choice between two or more decodings of the file. We characterize the probability distributions on three-symbol source alphabets for which this situation occurs. For n-symbol source alphabets we give a criterion for a family of geometric distributions on the source alphabet under which the file can be decoded unambiguously. We also present an efficient procedure that will choose which of two given sources actually produced a given file, if the file is unambiguous, or will give a proof that the file is essentially ambiguous, in the setting of an independent source. This and other results concerning independent sources depend upon conversions from Huffman trees to Markov chains and Markov sources.

To see a case of ambiguity, consider the three-symbol source alphabet $\{A, B, C\}$. In the source stream $BABABAACACAA$ the most common symbol is A, so the Huffman algorithm will assign A the shortest codeword; for example, it may use the encoding rule

$$A \rightarrow 1, \quad B \rightarrow 01, \quad C \rightarrow 00.$$

The encoded file would then be 01101101110010011. But this file is also an encoding of the source stream $ABABABCACAB$, using the equally valid encoding rule

$$A \rightarrow 0, \quad B \rightarrow 11, \quad C \rightarrow 10.$$

So the file is ambiguous.

The idea of using data compression schemes for encryption is very old, dating back at least to Roger Bacon in the 13th century [6, p. 90]. The field of data compression has grown vigorously since Huffman's paper. Rubin [9] and Jones [5] discuss the ways in which data compression algorithms may be used as encryption techniques. Klein *et al.* [7] have considered the cryptographic properties of Huffman codes in the context of a large, compressed natural language database on CD-ROM. Motivated by the same problem, Fraenkel and Klein [2] have shown that the problem of finding the encoding rule given *both* a sample of the source stream and the corresponding sample of the encoded file is NP-complete. By contrast with [2], where the issue is computational difficulty, we are concerned in this correspondence with the issue of information-theoretic impossibility.

There is some flexibility in how the Huffman algorithm is implemented. We distinguish two cases of interest. In the first, whenever two subtrees are combined, an arbitrary decision may be made as to which subtree should become the left subtree and which subtree should be the right subtree. We call the resulting Huffman code an *arbitrary* Huffman code. In another variant, the subtree of greater total weight (probability) is always made the right subtree. We call the resulting Huffman code a *right-heavy* Huffman code. We assume that the cryptanalyst knows whether the coder is using an arbitrary Huffman code or a right-heavy Huffman code.

An implementor of the Huffman algorithm must also decide how to handle "ties" when there are more than two symbols of minimum or next-to-minimum weight. In this correspondence we assume for simplicity that such ties do not occur.

We assume the cryptanalyst knows the size n of the source alphabet but not the source symbol probabilities $p_1, p_2, \cdots, p_n$. It may be the case that each source symbol is produced *independently* of the previously produced source symbols, in accordance with the source symbol probabilities. We say the source is *independent* in this case. If the source is not independent, we call it *adversarial*. When the source is independent, we distinguish the case in which the cryptanalyst knows *a priori* that the source is independent from the case in which she does not.

Manuscript received November 23, 1995. The work of D. W. Gillman was supported by the NSF under Grant CCR-8912586, the AFOSR under Grant 89-0271, and the Connaught Fund under Grant 3-370-120-20. The work of M. Mohtashemi was supported by the NSF under Grant CCR-8912568, and the AFOSR under Grant 89–0271. The work of R. L. Rivest was supported by the NSF under Grants CCR-8914428 and CCR-9310888, and the Siemens Corporation.

D. W. Gillman is with the Computer Science Department, University of Toronto, Toronto, ON, Canada M5S 1A4.

M. Mohtashemi is at 4371 Winters Chapel Rd., #1426, Atlanta, GA 30360 USA.

R. L. Rivest is with the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Publisher Item Identifier S 0018-9448(96)02997-5.

In this correspondence we focus on the problem of determining the Huffman tree used to create a Huffman-coded file, without determining how the source symbols are attached to the leaves of the tree. We do not address this question of matching up the codewords with the source symbols. This could perhaps be done efficiently once the codewords are known by making use of known source symbol probabilities, for example.

We say that an encoding rule (equivalently, a Huffman tree, or a codeword set) is *Huffman consistent* if, according to what the cryptanalyst knows *a priori*, the file could have been encoded by that rule. An encoding rule is *ambiguous* if, loosely, the file it produces could have been encoded from another source using a different encoding rule. To make this precise, we assume that the frequency in the source stream of the ith source symbol is exactly p_i. In addition, when the cryptanalyst knows that the source is independent, we assume that the asymptotic frequency in the source stream of any finite string w of source symbols is $\Pr[w]$. Here $\Pr[w] := p_{i_1} \cdots p_{i_m}$, when $w = i_1 \cdots i_m$. Equivalently, the cryptanalyst has access to an infinitely long file produced by the source. We justify this definition by observing that for any fixed w, the frequency of w in the file approaches $\Pr[w]$ with probability 1 as the length of the file goes to infinity. When we refer to an ambiguous *file*, we mean the file produced by an ambiguous *encoding rule*.

It may sometimes happen that the cryptanalyst obtains "truncation information:" a given codeword set may be impossible because the file does not parse into an integral number of codewords. (For example, the file 00000 can be produced from the three-symbol codeword set {0, 10, 11} but not from {00, 01, 1}.) In this correspondence we assume that such truncation information does not arise.

According to our definition the encoding is done with respect to the *actual* frequencies of the symbols in the source file, rather than according to some *estimated* frequencies that the encoder may know (say of the letters in English). Huffman coding is normally done with the actual frequencies. We note that using estimated probabilities is certainly possible, and would cause additional problems for the cryptanalyst.

Initially, we had expected the problem of "breaking a Huffman code" to be quite tractable. We were a little surprised to find that there is considerable ambiguity in this process. The techniques we present are capable of resolving the ambiguity in some cases, but in many others we show that it is "intrinsic." This correspondence not only presents techniques for reverse-engineering Huffman codes in many cases, but also presents numerous results indicating when our techniques fail—that is, when the problem presents "intrinsic" or "true" ambiguity. Such ambiguity is a blessing to the cryptographer, but a problem for the cryptanalyst, who must find other means of "breaking" such codes. (We leave this as an open problem.)

II. Three-Symbol Source Alphabets

We assume that the source alphabet is $\{A, B, C\}$, and that these symbols have respective probabilities a, b, and c in the source file. When $n = 3$, there are only two legal codeword sets: $C_0 = \{0, 10, 11\}$ and $C_1 = \{1, 01, 00\}$. We assume here that C_1 was chosen by the Huffman algorithm to code the file, so that a, b, and c are the respective probabilities of 1, 01, and 00 when the file is parsed according to C_1. When the file is parsed according to C_0, the frequencies of 0, 10, and 11 are, respectively, denoted x, y, and z.

A. Adversarial Source

In this subsection we assume that the cryptanalyst knows only that the file was encoded using the right-heavy Huffman algorithm and that the source has three symbols. The source is adversarial; that is, it generates any stream of symbols in which A, B, and C appear with frequencies a, b, and c, respectively, but the order is arbitrary.

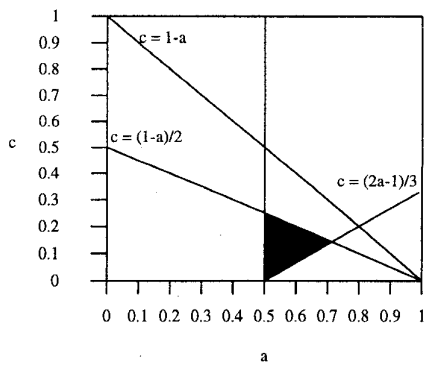

Fig. 1. Region of ambiguity for 3-symbol right-heavy Huffman trees, adversarial source.

We give a necessary and sufficient condition on rational numbers a, b, and c, under which the source may produce an ambiguous file. Since we are assuming that the right-heavy Huffman algorithm uses codeword set C_1, we immediately have $a > \frac{1}{2}$ and $c \leq (1-a)/2$.

Theorem 1: Let (a, b, c) be a probability distribution of rational numbers such that $a > \frac{1}{2}$ and $c \leq (1-a)/2$. Then there exists an adversarial three-letter source with probability distribution (a, b, c) on the source alphabet which produces an ambiguous file, if and only if $2a - 3c \leq 1$.

The region of ambiguity is shown in Fig. 1.

Proof: Assume the file is ambiguous. Recall the definition of x, y, and z at the beginning of the section. The frequency of 1's in the file is

$$\frac{a+b}{a+2b+2c} = \frac{y+2z}{x+2y+2z}$$

which can be written

$$\frac{1-c}{2-a} = \frac{1+z-x}{2-x}.$$

The right-hand side achieves a maximum value of $\frac{2}{3}$ when $z = x = \frac{1}{2}$, and it follows that $2a - 3c \leq 1$.

For the other direction we outline the construction of a source stream with the correct symbol frequencies which produces an ambiguous file. First, pick an integer k large enough so that ka, kb, and kc are all even integers. The source stream will consist of blocks of k symbols each, of two types. The first type, when encoded according to C_1 and parsed according to C_0, will give an equal number of 0's and 11's and some positive number of 10's. The second type will give no 10's and more 0's than 11's. It is fairly straightforward to construct such blocks given the condition of the theorem. Out of such blocks it is easy to build a source stream which, when encoded according to C_1 and parsed according to C_0, gives frequencies x, y, and z that satisfy $y \leq z \leq x \leq \frac{1}{2}$. ■

To illustrate the construction, suppose that $a = 0.6$, $b = 0.3$, and $c = 0.1$, and let us choose $k = 20$. The two blocks are

$$B_1 = BABABACAAAABABABACAA$$

and

$$B_2 = AABCBABAAAAABCBABAAA.$$

Parsed according to C_0, B_1 gives ratios $x : y : z = 10 : 0 : 9$ and B_2 gives $8 : 2 : 8$. Therefore, the concatenation $B_1 B_2$ gives ratios $18 : 2 : 17$ and it is ambiguous.

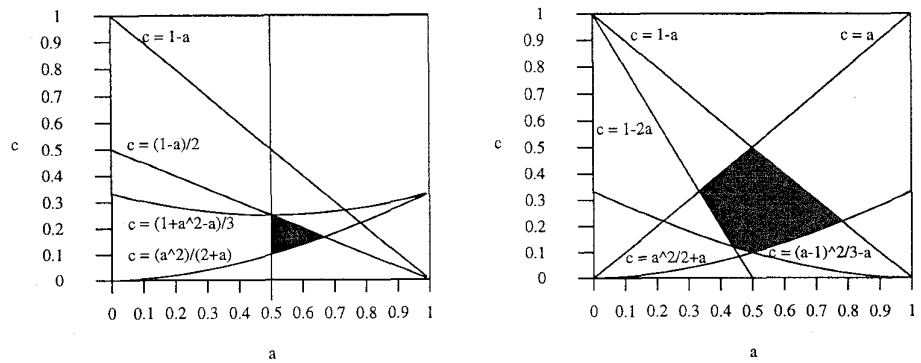

Fig. 2. Regions of ambiguity for right-heavy (left) and arbitrary (right) Huffman codes.

It seems fortunate that the necessary condition we arrived at by considering only the frequency of 1's in the file turned out to be sufficient, too. For a fixed $n > 3$ and a given n-symbol encoding rule, the frequency of 1's will give rise to a necessary condition for ambiguity. It would be interesting to determine whether this condition is sufficient in general.

B. Independent Source, Independence Unknown to Cryptanalyst

In this subsection we assume that the three-symbol source is independent, but that the cryptanalyst does not know this *a priori*.

For any fixed encoding rule an independent source generates a Markov chain on the internal nodes of the Huffman tree in a natural way. Here we are assuming the file is encoded according to C_1. In this case the Markov chain has two states, 1 (= root) and 2, and transition probabilities $p_{11} = a$, $p_{12} = 1 - a$, and $p_{21} = 1$. We can think of the transition from 2 to 1 as a 0-transition of probability $c/(1-a)$ plus a 1-transition of probability $b/(1-a)$.

Parsing according to C_0 is deterministic, so the independent source induces a process on the internal nodes, 3 (= root) and 4, of the Huffman tree used to parse the file. This process is not Markov, but there is also a process induced on the set of pairs of internal nodes, one from the encoding Huffman tree and one from the parsing Huffman tree. This process is a Markov chain, which we call the *cross-product machine* (CPM). The states of the CPM are $(1, 3)$, $(1, 4)$, $(2, 3)$, $(2, 4)$, and the relevant transition probabilities are $p_{(1,3)(1,4)} = a$, $p_{(1,3)(2,3)} = 1-a$, $p_{(1,4)(1,3)} = a$, $p_{(1,4)(2,3)} = 1 - a$, $p_{(2,3)(1,3)} = c/(1-a)$, and $p_{(2,3)(1,4)} = b/(1-a)$. The state $(2, 4)$ is transient. Using elementary linear algebra we can find the stationary probabilities of the recurrent states $(1, 3)$, $(1, 4)$, and $(2, 3)$. Each transition in the CPM corresponds to a parsing of a 0 or a 1 in the file. Using this fact and the stationary probabilities we compute

$$\begin{aligned} x &= \frac{(a-1)(2c+ac+ab)}{D} \\ y &= \frac{(a-1)(ac+b)}{D} \\ z &= \frac{-a(ac+b)}{D} \end{aligned} \tag{1}$$

where

$$D = a^2b + a^2c - ab - b - 2c.$$

Using (1), we can derive necessary and sufficient conditions for ambiguity in terms of a, b, and c. In the following two theorems we give two separate sets of conditions, one for right-heavy Huffman codes and one for arbitrary Huffman codes.

Theorem 2 (The Right-Heavy Case): Suppose a source produces the symbols A, B, and C, independently with probabilities a, b, and c, respectively. Suppose that $a \geq \frac{1}{2}$ and $c \leq (1-a)/2$, and the file is encoded using codeword set C_1. Then the encoding rule is ambiguous if and only if

$$a^2/(a+2) \leq c \leq (a^2+1-a)/3.$$

Proof: Assume that the encoding rule is ambiguous. Then the encoding rule using codeword set $C_0 = \{0, 10, 11\}$ with respective probabilities x, y, and z is also right-heavy Huffman-consistent; that is, $x \leq y + z$, $x \geq z$, and $z \geq y$. Plugging in the expressions for x, y, and z, from (1) and eliminating b, we get, respectively, $c \leq (a^2+1-a)/3$, $c \geq a^2/(a+2)$, and $a \geq \frac{1}{2}$, as desired.

The above derivation process is reversible. Thus if

$$a^2/(a+2) \leq c \leq (a^2+1-a)/3$$

and $a \geq \frac{1}{2}$, then $z \leq x \leq y + z$ and $z \geq y$; and hence the encoding rule is ambiguous. ■

Theorem 3 (The Arbitrary Case): Suppose a source produces the symbols A, B, and C, independently with probabilities a, b, and c, respectively. Suppose that $1 - 2a \leq c \leq a$, and the file is encoded using codeword set C_1. Then the encoding rule is ambiguous if and only if $c \geq a^2/(a+2)$ and $c \geq (a^2+1-2a)/(3-a)$.

Proof: Assume that the encoding rule is ambiguous. Then the encoding rule using codeword set $C_0 = \{0, 10, 11\}$ with respective probabilities x, y, and z is also right-heavy Huffman-consistent; that is, $x \geq y$ and $x \geq z$. Plugging in the expressions for x, y, and z, from (1) and eliminating b, we get, respectively, $c \geq (a^2+1-2a)/(3-a)$ and $c \geq a^2/(a+2)$, as desired.

The above derivation process is reversible. Thus if $c \geq a^2/(a+2)$ and $c \geq (a^2+1-2a)/(3-a)$, then $x \geq y$ and $x \geq z$; and hence the encoding rule is ambiguous. ■

Fig. 2 depicts the regions of ambiguity defined by the constraints on a and c in the theorems above.

C. Independent Source, Independence Known to Cryptanalyst

In this subsection we assume that the three-symbol source is independent, and that the cryptanalyst knows this *a priori*. We find that, as one expects, there is less room for ambiguity than in the setting where the cryptanalyst does not know that the source is independent.

Recall from the Introduction that the encoding rule is ambiguous if there is another source that, for any string w of source symbols, produces w with probability $\Pr[w]$. Here $\Pr[w] := p_{i_1} \cdots p_{i_m}$, when $w = i_1 \cdots i_m$.

Theorem 4: Suppose a source produces the symbols A, B, and C, independently with probabilities a, b, and c, respectively. Suppose that $a \geq \frac{1}{2}$ and $c \leq (1-a)/2$, and the file is encoded with codeword set C_1. Then the encoding rule is ambiguous if and only if $c = (1-a)^2$ and $a \leq (\sqrt{5}-1)/2$.

Proof: Let us assume that the encoding rule is ambiguous. Then the file could have been produced by another source using the encoding rule for codeword set C_0 with associated probabilities $y \leq z \leq x \leq \frac{1}{2}$. The string 011 is a *synchronizer* for both three-symbol encoding rules. That is, when the file is parsed by either rule, there is a word break after each instance of 011. Therefore, by independence, $(1-a) = \Pr[0|011] = x$ and $c = \Pr[00|011] = x^2 = (1-a)^2$. ($\Pr[w'|w]$ refers to the probability of seeing w' after seeing w.) The last inequality follows from $a^2 = z \leq x = 1 - a$.

Conversely, if $c = (1-a)^2$, then the file could have been produced by a two-symbol source with probabilities a and $1-a$. Equivalently, it could have been produced by a source using the encoding rule for codeword set C_0 with associated probabilities $x = 1-a$, $y = a(1-a)$, and $z = a^2$. A source with these probabilities really would use codeword set C_0 because $a \leq (\sqrt{5}-1)/2$ and therefore $y \leq z \leq x \leq \frac{1}{2}$. ■

Remark: The theorem implies that any ambiguous file is indistinguishable from a file produced by a two-symbol source with probabilities a and $1-a$. Surprisingly, for every case we have examined, the same thing happens for alphabets of more than three symbols: whenever two Huffman trees are ambiguous with an independent source they both "collapse" to a two-symbol tree.

III. Arbitrary Finite Source Alphabets

A. Independent Source, Independence Unknown to Cryptanalyst

In this subsection we consider an n-symbol source alphabet and an independent source. We assume that the cryptanalyst knows n *a priori*, but does not know that the source is independent. Our analysis of three-symbol source alphabets showed that the region of ambiguity depends on the probability distribution on the source alphabet. We now consider the family of geometric probability distributions on an n-symbol source alphabet. For the geometric distribution $p_i = c\alpha^{i-1}$, $i = 1, \cdots, n$, where c is a normalizing constant, we establish a bound on α below which the encoding rule is never ambiguous.

Let us assume that the encoding rule uses a right-heavy Huffman tree. When $p_i = c\alpha^{i-1}$ for $\alpha < \frac{1}{2}$, each internal node of the Huffman tree has a leaf as its right child. We call a tree satisfying this condition a *left-leaning chain*. By the independence of the source, each internal node x has a fixed probability r_x of branching right. This is the probability that the next bit in the file is a 1, given that the bits seen so far, parsed by the tree, end at x.

Lemma 1: Let T be the generating (right-heavy) Huffman tree for an independent source. Suppose T is a left-leaning chain with n leaves, and according to the geometric distribution the leaf probabilities are c, $c\alpha$, $c\alpha^2$, $\cdots$, $c\alpha^{n-1}$, where $c = (1-\alpha)/(1-\alpha^n)$. Then for each internal node x in T

$$r_x := \Pr[\text{branching right}] \geq 1 - \alpha.$$

Proof: Let d_x be the depth of x. By the left-leaning property, the right child of x represents a source symbol with probability $c\alpha^{d_x}$, and the probability of branching right is

$$\begin{aligned} r_x &= \frac{\alpha^{d_x}}{\alpha^{d_x} + \alpha^{d_x+1} + \cdots + \alpha^{n-1}} \\ &= \frac{1-\alpha}{1-\alpha^{n-d_x}} \\ &\geq 1 - \alpha. \end{aligned}$$

■

Therefore, on any Huffman tree for parsing the source stream, each internal node also has probability at least $1-\alpha$ of branching right, and at most α of branching left.

The following theorem and its corollary establish a condition under which there is no ambiguity. The cryptanalyst knows the number n of symbols in the source alphabet but not that the source is independent, and "Huffman-consistent" is with respect to this *a priori* knowledge.

Theorem 5: Assume that $(1-\alpha)^2 \geq \frac{1}{2}$. Let T be an n-symbol right-heavy Huffman tree for an independent source such that the probability of branching right at any internal node of T is $\geq 1-\alpha$. Then the rightmost leaf (RML) of T is a child of the root.

Proof: Suppose the right child of the root is not a leaf. Then it has a right child x. At some point in the Huffman algorithm, x is considered to be a leaf with probability $\geq (1-\alpha)^2 \geq \frac{1}{2}$ (see [1]). According to the algorithm, its depth should be 1, a contradiction. ■

Corollary 1: Consider an n-symbol independent source in which the probability distribution on the source alphabet is geometric; i.e., $p_i = c\alpha^{i-1}$, where $c = (1-\alpha)/(1-\alpha^n)$. If $\alpha \leq 1 - 1/\sqrt{2} = 0.292\cdots$, then the encoding rule for this source is not ambiguous.

Proof: Since $\alpha < \frac{1}{2}$ the generating Huffman tree for this source is left-leaning. Let T be any Huffman-consistent tree. We will show that T must be left-leaning, and the corollary will follow. By Lemma 1 and Huffman consistency, the probability of branching right at any internal node of T is $\geq 1-\alpha$. But every subtree of T is a Huffman tree for some independent source. The assumption on α implies that $(1-\alpha)^2 \geq \frac{1}{2}$. So by Theorem 5, the RML of every subtree of T is a child of the root. Therefore, the right child of every internal node of T is a leaf, so by definition T is left-leaning. ■

B. Independent Source, Independence Known to Cryptanalyst

In this subsection we again consider an n-symbol source alphabet and an independent source, but here we assume that the cryptanalyst knows *a priori* that the source is independent in addition to knowing n.

We consider the following hypothesis testing problem: given the Huffman tree for the encoding rule and another Huffman tree, decide which tree is the encoding rule or prove that the file is ambiguous. This problem can be reduced to checking the Huffman consistency of a single tree, and the naive algorithm for either problem involves evaluating $\Pr[w]$ for all w of length n. Although nothing better is known for checking Huffman consistency, we show that the cryptanalyst can solve the hypothesis testing problem by evaluating $\Pr[w]$ for $O(n^2)$ different w each of length at most $2n$. This suggests that the problems of checking consistency of a tree or determining whether a file is ambiguous may admit efficient solutions, where the measure of cost is the number of binary strings w for which the frequency in the file is evaluated. We solve the hypothesis testing problem by reduction to the equivalence problem for finite Markov sources.

A *Markov source,* or *hidden Markov chain,* (M, χ), is a Markov chain M in which each state s is labeled $\chi(s) = 0$ or 1. Starting from some distribution π on the set of states, the Markov chain proceeds randomly through a sequence of states whose labels form a binary string. $\Pr_M[w] = \Pr_{M,\chi,\pi}[w]$ is the probability that M will produce the binary string w.

An independent source is equivalent to a Markov source, by a conversion shown in Fig. 3. Under this conversion, there is a state of the Markov chain corresponding to each internal node in the tree, and the label of that state is the label of the tree branch leading down to the node. The root corresponds to two states (r_0 and r_1 in the figure), since branches to leaves implicitly lead back to the root. The starting distribution π is the point mass at either of the two root states.

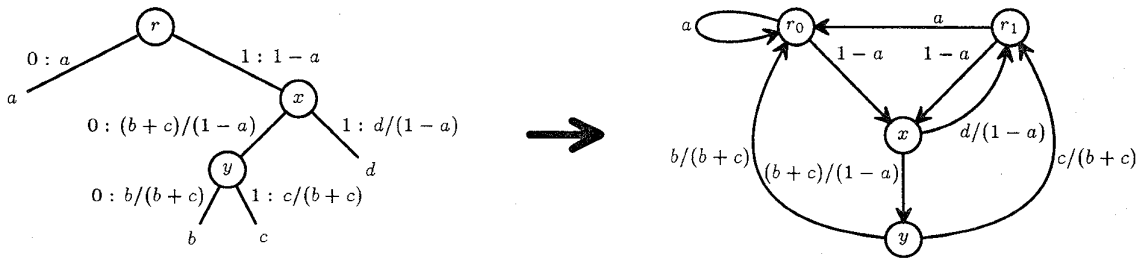

Fig. 3. Converting a Huffman tree to a Markov source with 0-states r_0 and y and 1-states r_1 and x.

Two Markov sources M and N are *equivalent* if for every $w \in \{0, 1\}^*$, $\Pr_M[w] = \Pr_N[w]$. For every m-state hidden Markov source L (with a given starting distribution) there are m^2 "witnesses" $w_1, \cdots, w_{m^2}$, $w_i \in \{0, 1\}^*$, such that the values $\Pr_L[w_1], \cdots, \Pr_L[w_m]$ characterize the source up to equivalence. Recent papers on Markov sources [3], [10] give algorithms for finding these witnesses with cost $O(m^2)$, where the cost is the number of strings w for which $\Pr_L[w]$ must be evaluated. Each such evaluation involves at most the multiplication of $2m$ $m \times m$ matrices, since the witnesses are all of length at most $2m$. Two sources M and N are equivalent if and only $\Pr_M[w] = \Pr_N[w]$ for each w which is a witness for M or N.

Theorem 6: Suppose an n-symbol source is independent. Suppose the Huffman tree T for the encoding rule is given along with another n-symbol Huffman tree S. Then there is an algorithm with running time polynomial in n that determines whether that file could have been produced by T only, could have been produced by S only, or that the file is ambiguous as to whether it could have been produced by T or S.

Proof: (Algorithm). Parse the file according to both T and S. Then the frequencies of the left and right branches of each internal node of each tree will yield the transition probabilities of the hidden Markov sources M_T and M_S corresponding to T and S, respectively, according to the conversion shown in Fig. 3. (For each internal node, the probability of branching left is simply $(\Pr[W0]/(\Pr[w])$for a string w determined by the node.)

Test whether M_T and M_S are equivalent using an existing algorithm (the cost of this is $O(n^3)$ $n \times n$ matrix multiplications). If they are, then the file is ambiguous. If not, then for some witness w_S for M_S it is the case that $\Pr[w_S] \neq \Pr_{M_S}[w_S]$. To find w_S, compare $\Pr[w]$ to $\Pr_{M_T}[w]$ for each witness w for M_T and compare $\Pr[w]$ to $\Pr_{M_S}[w]$ for each witness w for M_S, at a cost of $O(n^2)$ evaluations. ■

Remark: The problem of efficiently determining whether a given Huffman tree S is consistent with the file remains open. The naive algorithm is to to evaluate $\Pr[w]$ for each w of length n in the file, at a cost of 2^n evaluations. The algorithm given by Theorem 6 does not immediately solve this problem, except in the case where M_S is *minimal*, i.e., not equivalent to any Markov source with fewer states [3]. The problem is that the witness strings for the tree T which was used to produce the file remain unknown, and even if T is not equivalent to S, the file may mimic S on all of the witnesses of S (we speak loosely of trees being equivalent instead of the corresponding Markov sources).

It remains an open question to determine efficiently whether a given n-symbol Huffman tree is ambiguous, when the source is independent and this is known to the cryptographer. As we remarked in Section II-C, it could be the case that any ambiguous n-symbol Huffman tree is equivalent to a two-symbol tree. By extending the method of Section II-C, we have shown that this is the case for $n \leq 5$

ACKNOWLEDGMENT

The first author wishes to thank M. Sipser for many discussions about Markov sources.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Boston, MA: MIT Press/McGraw-Hill, 1990.
[2] A. S. Fraenkel and S. T. Klein, "Complexity aspects of guessing prefix codes," *Algorithmica*, vol. 12, pp. 409–419, 1994.
[3] D. Gillman and M. Sipser, "Inference and minimization for hidden Markov chains," in *Proc. 1994 ACM Symp. on Computational Learning Theory*, July 1994.
[4] D. A. Huffman, "A method for the construction of minimum-redundancy codes," in *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.
[5] D. W. Jones, "Applications of splay trees to data compression," *Commun. ACM*, vol. 31, no. 8, pp. 996–1007, 1988.
[6] D. Kahn, *The Codebreakers.* New York: Macmillian, 1967.
[7] S. T. Klein, A. Bookstein, and S. Deerwester, "Storing text-retrieval systems on CD-ROM: Compression and encryption considerations," *ACM Trans. Inform. Syst.*, vol. 7, pp. 230–245, 1989.
[8] M. Mohtashemi, "On the cryptanalysis of Huffman codes," MIT Lab. for Computer Sci. Tech. Rep., p. 617, May 1992.
[9] F. Rubin, "Cryptographic aspects of data compression codes," *Cryptologia*, vol. 3, pp. 202–205, 1979.
[10] W.-G. Tzeng, "A polynomial-time algorithm for the equivalence of probabilistic automata," *SIAM J. Comput.*, vol. 21, no. 2, pp. 216–227, 1992.