Working Paper Series
ISSN 1170-487X

On the Insecurity of Arithmetic Coding

by John Cleary, Sean Irvine, Ingrid Rinsma-Melchert

Working Paper 94/7

June, 1994

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

On the Insecurity of Arithmetic Coding

John Cleary, Sean Irvine, Ingrid Rinsma-Melchert

School of Computing and Mathematical Sciences
University of Waikato
Private Bag 3105
Hamilton, New Zealand

June 13, 1994

Abstract

Arithmetic coding is a technique which converts a given probability distribution into an optimal code and is commonly used in compression schemes. The use of arithmetic coding as an encryption scheme is considered. The simple case of a single binary probability distribution with a fixed (but unknown) probability is considered. We show that for a chosen plaintext attack $w + 2$ characters is sufficient to uniquely determine a w-bit probability. For many known plaintexts $w + m + O(\log m)$ symbols where m is the length of an initial sequence containing just one of (the two possible) symbols is sufficient. It is noted that many extensions to this basic scheme are vulnerable to the same attack provided the arithmetic coder can be repeatedly reset to its initial state. If it cannot be reset then their vulnerability remains an open question.

1 Introduction

The presence of redundancy in plaintext has always been a fertile source for attacks on encryption schemes. For example, substitution ciphers for English are easily attacked by noting that the letter e is the most frequent character and that a three letter word ending in e is very likely to be *the* and so on. A more recent example is an attack on any public key encryption system using the redundancy of the plaintext to store a table of the most likely sequences and their decodings [1]. One way to protect against such statistical attacks is to remove the redundancy from the plaintext before encrypting it. That is, the plaintext should first be compressed.

Given that compression is necessary, a natural question is whether compression techniques on their own are sufficient to provide a reliable encryption mechanism. This would certainly be attractive because a single mechanism might be able to minimize transmission and storage costs as well as provide encryption thus reducing the overheads of doing each separately.

The intuition here is that the better the compression, the less likely it is that the compressed text can be (statistically) attacked. For compressing natural language text the best current techniques use adaptive models and arithmetic coding—best in the sense of best compression rather than shortest time or smallest space [2, 3, 4, 5]. Such schemes consist of two parts. The first, the *model*, accumulates statistics about the plaintext seen so far. For example, a simple model might remember the frequency of letters in the text seen so far. The result of such a model is a set of probabilities, one for each character that can occur next. The second, *arithmetic coding* [8], takes such a sequence of probability distributions (together with the actual plaintext) and generates an optimal compressed output code. Arithmetic coding is optimal in that it can be arbitrarily close to the true entropy of the probability distributions. Thus it is better than encodings such as Huffman codes [7] which are restricted to integral bits per character. Using more sophisticated models which condition the probabilities on preceding characters it is possible to achieve very high compression. For example, long sequences of English text can be compressed to just over 2 bits per character [2, 6].

One of the attractions of arithmetic coding is that it is easily and efficiently implemented using fixed length computer arithmetic. Encoding requires a fixed point multiply operation and decoding requires a divide operation (hence the name *arithmetic* coding). The use of finite precision arithmetic makes the output sequence suboptimal with respect to the given probabilities. However, w-bit arithmetic coding causes a fractional increase in the output length of order 2^{-w} [9, 10]. Thus for practical purposes 32 bit arithmetic is sufficiently close to optimal.

The security of arithmetic coding plus adaptive models has been considered previously in [12, 14]. The first suggestion of which we are aware for their use for in encryption is [13]. In this paper we consider the security of arithmetic coding on its own. To simplify the analysis we initially consider a single binary probability distribution where the probability of a symbol occurring is fixed but unknown. We then consider the problem of how to extract the probability given some sample output from the encoder. Once the probability is known all future text can be decrypted. It is shown that if the probability is stored to w-bits of precision then using chosen plaintext the probability can be decoded using $w+2$ symbols. For many known plaintexts the attack will take $w+m+O(\log m)$ symbols where m is the length of the initial sequence containing just one of the two possible symbols.

This is a very strong negative result for the use of arithmetic coding on its own as an encryption technique. Stronger for example, than the results in [11, 14] which required assumptions about the model. Bergen's attack [12] requires that the adaptive model be forced to have certain known probabilities by a (very long) sequence of chosen plaintext. This negative result seems to carry over to a wide range of related variations in the encoding mechanism (the probability is adapted, the initial state of the encoder is unknown, the alphabet has more than two symbols, and so on) so long as it is possible to repeatedly reset the encoder and start a new encoding sequence. If it is not possible to reset then it is an open question as to how hard it is to attack an encoder using these variations.

One way of viewing the operation of arithmetic coding is that it generates as output a single real number which is uniformly distributed (the individual bits are random) with respect to the probability distribution of the model. Sufficient leading bits from the real number output are transmitted to ensure that the original text can be decoded. The analysis in Section 2 approximates the actual (finite precision) encoder with an infinite precision one. The output for a particular plaintext sequence can then be expressed as a polynomial in the unknown probability. Section 3 shows how to infer the unknown probability by solving the polynomial. Section 4 examines the practical (finite precision) case and shows by example that the infinite precision approximation gives results sufficiently close to the actual answer that there is no difficulty in computing the finite precision probability. Section 5 extends the attack to known plaintext. Section 6 considers related encryption schemes and future work. An appendix provides the details for some of the proofs.

2 A Static Model

Consider encoding the output of a binary source with alphabet $\{a, b\}$ using a static model. Let p be the probability used to encode an a and $q = 1-p$ be the probability used to encode a b. Neither probability can be 0 or 1 as we must always allow some probability to each symbol occurring. The model is set up as illustrated in Figure 1 where l is a lower bound and h an upper bound on the output real number. δ is the difference between the upper and lower bound. Initially $l = 0$, $h = 1$, and $\delta = 1$. At each succeeding step the interval δ is narrowed and either l is incremented or h is decremented.

Assume we have an infinite precision arithmetic coder. The following recursive relations define characteristic polynomials for l, h, and δ, where $\delta = h - l$.

If an a is encoded then

$$\begin{aligned} l' &= l, \\ \delta' &= p\delta, \\ h' &= h - q\delta. \end{aligned} \tag{1a}$$

If a b is encoded then

$$\begin{aligned} l' &= l + p\delta, \\ \delta' &= q\delta, \\ h' &= h. \end{aligned} \tag{1b}$$

Let the input sequence be $S = s_1 s_2 \cdots s_i \cdots s_n$, where s_i is either a or b. We use $\overline{S}$ to denote the complement of sequence S. For example, if $S = abbabbb$ then $\overline{S} = baabaaa$. Define

$$a_i = \begin{cases} 1 & \text{if } s_i = a \\ 0 & \text{otherwise} \end{cases},$$

$$b_i = \begin{cases} 1 & \text{if } s_i = b \\ 0 & \text{otherwise} \end{cases},$$

$$\overline{a_j} = \sum_{i=1}^{j} a_i, \qquad (0 \le j)$$

$$\overline{b_j} = \sum_{i=1}^{j} b_i. \qquad (0 \le j)$$

That is, $\overline{a_j}$ and $\overline{b_j}$ count the number of as and bs respectively in the input up to and including the jth character.

Let the characteristic polynomials for l, h, and δ after encoding i input characters be l_i, h_i, and δ_i respectively. From (1)

$$\delta_i = p^{\overline{a_i}} q^{\overline{b_i}} = p^{\overline{a_i}} (1-p)^{\overline{b_i}}. \tag{2a}$$

The lower bound only changes when a b is sent, thus

$$l_i = \sum_{j=1}^{i} b_j p \delta_{j-1}. \tag{2b}$$

Likewise the upper bound only changes when an a is sent, thus

$$h_i = 1 - \sum_{j=1}^{i} a_j q \delta_{j-1} = 1 - \sum_{j=1}^{i} a_j (1-p) \delta_{j-1}. \tag{2c}$$

On occasion we will have need of the derivatives of the characteristic polynomials. The derivatives are easily calculated to be:

$$\frac{\mathrm{d}}{\mathrm{d}p}\delta_i = p^{\overline{a_i}-1}(1-p)^{\overline{b_i}-1}[(1-p)\overline{a_i} - p\overline{b_i}], \tag{3a}$$

$$\frac{\mathrm{d}}{\mathrm{d}p} l_i = \sum_{j=1}^{i} b_j p^{\overline{a_{j-1}}} (1-p)^{\overline{b_{j-1}}-1}[(\overline{a_{j-1}}+1)(1-p) - \overline{b_{j-1}}p], \tag{3b}$$

$$\frac{\mathrm{d}}{\mathrm{d}p} h_i = \sum_{j=1}^{i} a_j p^{\overline{a_{j-1}}-1} (1-p)^{\overline{b_{j-1}}}[p(\overline{b_{j-1}}+1) - \overline{a_{j-1}}(1-p)]. \tag{3c}$$

We use lg to denote logarithms base 2.

3 Inferring p

The interval left at the end of encoding the sequence S using probabilities p and q will be $\delta_n = h_n - l_n$. Let $\hat{p}$ be an estimate of p. If we are going to succeed in decoding all n characters using $\hat{p}$ we require an overlap between the coding interval which resulted when S was encoded with p and the coding interval which would result if S were encoded using $\hat{p}$. Let the width of values of p that decode correctly be Δp. Then the range of either l_n or h_n can be twice the width of the interval; that is,

$$\Delta p \frac{\mathrm{d}}{\mathrm{d}p} l_n = 2\delta_n, \tag{4a}$$

or equivalently,

$$\Delta p \frac{\mathrm{d}}{\mathrm{d}p} h_n = 2\delta_n. \tag{4b}$$

In obtaining these equations we have made the assumption that the width of the coding interval resulting from encoding S does not change significantly over the range Δp, as shown in Figure 2. The assumption is reasonable because the derivative $\left|\frac{\mathrm{d}}{\mathrm{d}p}\delta_n\right|$ will be very small for most sequences. Later we verify this assumption more rigorously.

If we are going to succeed in determining p then we require Δp to be as small as possible. This can be achieved by making δ_n small and $\frac{\mathrm{d}}{\mathrm{d}p}h_n$ large (or equivalently $\frac{\mathrm{d}}{\mathrm{d}p}l_n$ large). In particular, for a w-bit arithmetic encoder we require $\Delta p \le 2^{-w}$ to uniquely determine p.

Theorem 1: For any sequence beginning with bab

$$\frac{\mathrm{d}}{\mathrm{d}p}h_n \geq \begin{cases} 1 & \text{for } 0 < p \leq \frac{1}{2} \\ 2(1-p) & \text{for } \frac{1}{2} < p < 1 \end{cases} \tag{5a}$$

Proof: Using (3c) we have

$$\frac{\mathrm{d}}{\mathrm{d}p}h_n = 2 - 2p + \sum_{j=3}^{n} a_j p^{\overline{a_{j-1}}-1}(1-p)^{\overline{b_{j-1}}}[p(\overline{b_{j-1}}+1) - \overline{a_{j-1}}(1-p)].$$

Terms in the sum will be negative when

$$\overline{a_{j-1}} > \frac{p}{1-p}(\overline{b_{j-1}}+1).$$

Next note $p^{\overline{a_{j-1}}-1}(1-p)^{\overline{b_{j-1}}}$ is decreasing for increasing j, and consequently the interval is narrowed between subsequent as. Sending bs cannot cause expansion. Therefore the derivative will be a minimum when the sequence is bab or $baba^{\infty}$. If the sequence is bab then the derivative is $2-2p$ and the result follows. If the sequence is $baba^{\infty}$ then the derivative is at least 1 for $0 < p \leq \frac{1}{2}$ and at least $2(1-p)$ for $\frac{1}{2} < p < 1$ (as shown in the appendix). This completes the proof.

Δ

Corollary 1: For any sequence beginning with aba

$$\frac{\mathrm{d}}{\mathrm{d}p}h_n \geq \begin{cases} 2p & \text{for } 0 < p \leq \frac{1}{2} \\ 1 & \text{for } \frac{1}{2} < p < 1 \end{cases} \tag{5b}$$

Proof: By symmetry. Take $p_1 = 1-p$ and $S_1 = \overline{S}$ and apply Theorem 1 to p_1 and S_1.

Δ

Theorem 2: For the plaintext $S = (ba)^{\frac{n}{2}}$, $w+2$ symbols are sufficient to determine a w-bit probability.

Proof: First let us check the magnitude of $\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})$.
Using (3a),

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| = \left|\frac{n}{2}p^{\frac{n}{2}-1}(1-p)^{\frac{n}{2}-1}(1-2p)\right|.$$

It follows that (details are in the appendix),

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| \ll 1 \text{ and } \left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| \ll 2(1-p)$$

except when $p \to 1$, in which case replace the $\ll$ by $\leq$.
If $p \leq \frac{1}{2}$ then $\frac{\mathrm{d}}{\mathrm{d}p}h_n \geq 1$ and $\delta_n \leq \left(\frac{1}{2}\right)^n$ (with equality occurring when $p = \frac{1}{2}$). So from (4b):

$$\begin{aligned} \Delta p &\leq 2\left(\frac{1}{2}\right)^n \\ &= \left(\frac{1}{2}\right)^{n-1} \end{aligned}$$

If $p > \frac{1}{2}$ then $\frac{\mathrm{d}}{\mathrm{d}p}h_n \geq 2(1-p)$. So from (4b):

$$\begin{aligned} \Delta p &\leq \frac{2p^{\frac{n}{2}}(1-p)^{\frac{n}{2}}}{2(1-p)} \\ &\leq p^{\frac{n}{2}}(1-p)^{\frac{n}{2}-1} \\ &\leq [p(1-p)]^{\frac{n}{2}-1} \\ &\leq \left(\frac{1}{4}\right)^{\frac{n}{2}-1} \\ &= \left(\frac{1}{2}\right)^{n-2}. \end{aligned}$$

Therefore a sequence of $w+2$ symbols is sufficient to uniquely determine the state of a w-bit register. Δ

Other sequences work as well. Since $\delta_n = p^{\overline{a_n}}(1-p)^{\overline{b_n}}$ it is desirable to send more as when p is near 0 and more bs when p is near 1 to ensure δ_n is small.

In practice a few more symbols may be needed since actual implementations of arithmetic coders have small errors due to the use of finite-precision arithmetic. Figure 3 gives experimental evidence to support the theory developed above (when an arithmetic coder adapted from [8] is used). In all cases at most $w-1$ symbols were needed—slightly better than the bound achieved above. We did not allow probabilities $p=0$ and $p=1$ since these would cause infinite length outputs, but every other possible w-bit probability was tried. The number of bits used in the arithmetic was 17 (the implementation required this so that overflow and underflow did not occur).

4 Chosen Plaintext Attack

We have seen that a sequence of $w+2$ characters is sufficient to uniquely determine p in a w-bit arithmetic coder. This section shows how an attacker can mount a chosen-plaintext attack to efficiently determine the value of p.

Since the attacker knows the plaintext, S, the attacker can determine the characteristic polynomials $h(S)$ and $l(S)$, the bounds for the coding interval. By examining the output of the arithmetic coder the attacker can determine the result of encoding S, call this value γ. Since γ will lie in the interval bounded by l_n and h_n, the following inequalities hold:

$$l(S) \leq \gamma \leq h(S).$$

By solving these polynomial inequalities the attacker can bound p.

For example suppose we encode the plaintext $S = babababa$ using an 8-bit arithmetic coder with $p = \frac{94}{256} \approx 0.3672$. The output of an ideal arithmetic coder could be 011110101 which corresponds to $\gamma \approx 0.4785$.

The characteristic polynomials for *babababa* are

$$h(babababa) = p + p^2 - 6p^5 + 9p^6 - 5p^7 + p^8,$$

and

$$l(babababa) = p + p^2 - p^4 - 2p^5 + 3p^6 - p^7.$$

The following inequalities hold:

$$p + p^2 - p^4 - 2p^5 + 3p^6 - p^7 \leq 0.4785 \leq p + p^2 - 6p^5 + 9p^6 - 5p^7 + p^8.$$

By solving these polynomials, we get the following estimates for p: From $l(S)$, $p \approx 0.3683$; from $h(S)$, $p \approx 0.3663$. As expected $p \approx 0.3672$ falls between these two constraints and further $\frac{93}{256}$ and $\frac{95}{256}$ lie outside these constraints. So we have successfully uniquely identified p to be $\frac{94}{256}$.

The value of γ will vary slightly depending on the implementation. The arithmetic coder we used for the example above actually gave 0111101011 which corresponds to $\gamma \approx 0.4794$.

5 Known Plaintext Attack

It is possible to generalize the results of Section 3 to a known plaintext attack. In practice it is easier for an attacker to use (or guess) known plaintext than to use a chosen plaintext.

Obviously if the known plaintext starts with *aba* or *bab* then we can immediately apply our previous results.

By generalizing Theorem 2 to cover other possible sequences we can bound the number of symbols needed for a known plaintext attack. Theorem 3 shows that many short sequences can be successfully used to determine p.

Theorem 3: To determine a w-bit probability:

1. The sequences b^n and a^n require $n = p2^{w+1}$ symbols.
2. Many sequences of the form $b^m a^{n-m}$ or $a^m b^{n-m}$ with $m \geq 1$ and $n \geq m+1$ require $n = w + m + 1 + O(\lg m)$ symbols.

3. Sequences beginning with $b^m ab^l$ (respectively $a^m ba^l$) with $m \geq 1$ and $l \geq 1$ require $n = m + 1 + O\left(\frac{w + 1 - \lg(m+1)}{-\lg(1-p)}\right)$ symbols.

4. Sequences beginning with $b^m a^l b$, $m \geq 1$, $l \geq 2$ require $n = l + 1 + O(2^w p^l)$ provided $p \leq \frac{1}{2}$. The same bound holds for sequences beginning with $a^m b^l a$ when $p \geq \frac{1}{2}$.

Proof: This theorem is proved in the appendix for the sequences starting with b. The ideas used in the proof are much the same as those used in the proof of the earlier theorems. The proofs are symmetric for the sequences starting with a.

Δ

6 Further Work

The problem we have attacked above is a very simplified version of what happens in a real adaptive compressor. In such a situation there may be many more parts of the systems that will be unknown besides the single probability. For example: the initial state of the arithmetic coder—the upper and lower bound—may be unknown. In terms of the analysis that we have done above this introduces two new variables into the polynomial characterizing the output. Thus with a single attacking text there is no way to uniquely solve the polynomial which now has three unknowns. However, if it is assumed that the encoder can be restarted three times and each time attacked with a different input sequence then the three different polynomials can be solved to obtain all the unknowns in the system. We have not done a detailed analysis of this situation but if the m unknowns are stored to a precision of w bits them m sequences of approximately w symbols each, for a total of $O(mw)$ symbols seems to be enough to crack the system.

Many extensions can be modelled in this way by extending the polynomial to more unknowns. For example an alphabet of a symbols has $a - 1$ unknown probabilities. The model may have many states each with its own set of $a - 1$ probabilities, and so on. We also examined the case where the model probabilities are updated as text is seen ([2, 12]), again this gives a characteristic polynomial similar to those seen above with only the initial (unadapted) probability as an unknown. Thus many potential extensions of arithmetic coding are vulnerable to this style of attack.

It is unclear how reasonable it is to assume that the encoder can be repeatedly reset and retried to get the different independent polynomials. In situations where a communications link is unreliable and error correction and retry are not possible then it is necessary to periodically reset the state of the system and re-establish contact. This would be the case if the encrypted message were being broadcast to a number of recipients some of which were not receiving at one time but were later. Thus the need to resynchronize makes the system vulnerable to attack.

However, in a point-to-point communication or in applications where text is being stored it is reasonable to assume that resetting is never needed. The proposed attack then fails for any system with more than one unknown and it is an open question as to whether the extensions mentioned above are vulnerable to attack.

This discussion has also ignored the question of the complexity of finding solutions for n polynomials in n unknowns. In particular, the complexity of finding solutions as discrete valued solutions "near" the exact solutions need to be explored. It is not clear what the complexity of this calculation is; keeping in mind that the polynomials have a special form and so results in general about the complexity of solving polynomials may not apply here.

We are continuing investigations into the vulnerability of both types of systems, those that can and cannot be periodically reset.

References

[1] William J. Wilson, "Chinks in the armor of public key cryptosystems," University of Waikato Technical Report 94/3, March 1994.

[2] Timothy C. Bell, John G. Cleary, Ian H. Witten, *Text Compression*, Prentice Hall, 1990.

[3] G. V. Cormack & R. N. Horspool, "Data compression using dynamic Markov modelling," *Comput. J.*, **30,** (6), 1987.

[4] R. N. Horspool & G. V. Cormack, "Dynamic Markov modelling—a prediction technique," *Proc. Int. Conf. on the System Sciences*, Honolulu, HI, January 1986.

[5] J. A. Storer, *Data Compression: Methods and Theory*, Computer Science Press, Rockville, MD, 1988.

[6] J. G. Cleary & I. H. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Trans. Commun.*, COM-32 (4), pp. 396–402, 1984.

[7] D. A. Huffman, A method for the construction of minimum-redundancy codes, *Proc. Inst. Electr. Radio. Eng.*, **40,** 9, 1952, pp. 1098–1101.

[8] Ian H. Witten, Radford M. Neal, John G. Cleary, "Arithmetic coding for data compression," *C ACM*, **30,** 6, pp. 520–540, June 1987.

[9] Colin Boyd, "Enhancing Secrecy by Data Compression: Theoretical and Practical Aspects," 1990.

[10] Paul Glor Howard, *The Design and Analysis of Efficient Lossless Data Compression Systems*, Doctoral Thesis, Dept. of Computer Science, Brown University, Providence, R. I. 02912–1910, 1993.

[11] Helen A. Bergen & James M. Hogan, "Data security in a fixed-model arithmetic coding compression algorithm," *Computers & Security*, **11,** pp. 445–461, 1992.

[12] Helen A. Bergen & James M. Hogan, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Computers & Security*, **12,** pp. 157–167, 1993.

[13] Ian H. Witten & John G. Cleary, "On the privacy afforded by adaptive text compression," *Computers & Security*, **7,** (4), pp. 397–408, 1988.

[14] H. A. Bergen, J. H. Holford, & D. Longley, "A chosen plaintext attack on a non-adaptive arithmetic coding encryption algorithm," to be published.

Appendix – Derivation of Mathematical Results

Theorem 1: For any sequence beginning with *bab*

$$\frac{\mathrm{d}}{\mathrm{d}p}h_n \geq \begin{cases} 1 & \text{for } 0 < p \leq \frac{1}{2} \\ 2(1-p) & \text{for } \frac{1}{2} < p < 1 \end{cases}$$

Proof: This is not a complete proof, it only contains the details missing from the proof in the main text. From (3c)

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}p}h_n(baba^{n-3}) &= 2-2p+(1-p)^2\sum_{j=4}^{n}p^{j-4}[jp-j+3] \\
&= 2-2p+(1-p)^2\left[\sum_{j=4}^{n}jp^{j-3}-\sum_{j=4}^{n}jp^{j-4}+3\sum_{j=4}^{n}p^{j-4}\right] \\
&= 2-2p+(1-p)^2\left[np^{n-3}-4-\sum_{j=5}^{n}p^{j-4}+3\sum_{j=4}^{n}p^{j-4}\right] \\
&= 2-2p+(1-p)^2\left[np^{n-3}-4+3+2\sum_{j=5}^{n}p^{j-4}\right] \qquad \text{(geometric)} \\
&= 2-2p+(1-p)^2\left[np^{n-3}-1+2\frac{p(1-p^{n-4})}{1-p}\right] \\
&= (1-p)^2\left[np^{n-3}-1+\frac{2p-2p^{n-3}+2}{1-p}\right] \\
&= (1-p)\left[1+3p+(n-2)p^{n-3}-np^{n-2}\right]
\end{aligned}$$

If $0 < p \leq \frac{1}{2}$,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}p}h_n(baba^{n-3}) &= 1 + 2p + (n-2)p^{n-3} - (2n-2)p^{n-2} + np^{n-1} - 3p^2 \\ &\geq 1 + p[2 + (n-2)p^{n-4} - (n-2)p^{n-4} + np^{n-2} - 3p] \\ &\geq 1 + p[2 - 3p + np^{n-2}] \\ &\geq 1\end{aligned}$$

To show,

$$\frac{\mathrm{d}}{\mathrm{d}p}h_n(baba^{n-3}) \geq 2(1-p)$$

for $\frac{1}{2} < p < 1$, it suffices to show that $3p + (n-2)p^{n-3} - np^{n-2} \geq 1$. For the case $n = 3$ it is easy to see that $3p + (n-2)p^{n-3} - np^{n-2} = 1$.

Now, $3p + (n-2)p^{n-3} - np^{n-2} \geq 1$ if and only if $n \geq \frac{(1-3p)/p^{n-3}+2}{1-p}$. The right-hand side of this last inequality cannot exceed $2/(1-p)$, since $1 - 3p/p^{n-3} \leq 0$ for all cases under consideration. It follows that $n \geq 4$ satisfies the inequality.

Thus, $\frac{\mathrm{d}}{\mathrm{d}p}h_n(baba^{n-3}) \geq 2(1-p)$, for $\frac{1}{2} < p < 1$ and $n \geq 3$, which is what we wanted to prove.

△

Theorem 2: For the chosen plaintext $S = (ba)^{\frac{n}{2}}$, $w + 2$ symbols are sufficient to determine a w-bit probability.

Proof: From (3a),

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| = \left|\frac{n}{2}p^{\frac{n}{2}-1}(1-p)^{\frac{n}{2}-1}(1-2p)\right|,$$

which is a maximum when,

$$p = \frac{1}{2} \pm \frac{1}{2\sqrt{n-1}}.$$

It follows that (especially as n gets large)

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| \ll 1.$$

To show that

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| \ll 2(1-p)$$

it suffices to show that

$$\phi = \left|\frac{n}{2}p^{\frac{n}{2}-1}(1-p)^{\frac{n}{2}-2}(1-2p)\right| \ll 2.$$

ϕ is a maximum when

$$p = \frac{1}{2}\frac{2n-3+\sqrt{4n-7}}{2n-4}.$$

It follows that (especially as n gets large)

$$\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}((ba)^{\frac{n}{2}})\right| \ll 2(1-p).$$

△

Theorem 3: Many sequences can be used to uniquely determine a w-bit probability:

1. The sequences b^n and a^n require $n = p2^{w+1}$ symbols.
2. Many sequences of the form $b^m a^{n-m}$ or $a^m b^{n-m}$ with $m \geq 1$ and $n \geq m+1$ require $n = w + m + 1 + O(\lg m)$ symbols.
3. Sequences beginning with $b^m ab^l$ (respectively $a^m ba^l$) with $m \geq 1$ and $l \geq 1$ require $n = l + 1 + O\left(\frac{w+1-\lg(m+1)}{-\lg(1-p)}\right)$ symbols.

4. Sequences beginning with $b^m a^l b$, $m \geq 1$, $l \geq 2$ require $n = l + 1 + O(2^w)$ provided $p \leq \frac{1}{2}$. The same bound holds for sequences beginning with $a^m b^l a$ when $p \geq \frac{1}{2}$.

Proof:

1. The sequence $S = b^n$ requires $n = p2^{w+1}$.
   Since $l_n(b^n) = p^n$ the derivative is easily seen to be np^{n-1}. It follows that

$$\left(\frac{1}{2}\right)^w = \frac{2p^n}{np^{n-1}},$$

   which readily reduces to $n = p2^{w+1}$. Further, $\left|\frac{\mathrm{d}\delta_n}{\mathrm{d}p}\right| = n(1-p)^{n-1}$ which is much less than np^{n-1} except when $p \to 0$.

2. Sequences of the form $S = b^m a^{n-m}$ with $m \geq 1$ and $n \geq m+1$.
   Using (3c),

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^{n-m}) &= (1-p)^m \sum_{j=m+1}^{n} p^{j-m-2}[jp - j + m + 1] \\ &= (1-p)^m \left[m\frac{(1-p^{n-m-1})}{1-p} + np^{n-m-1}\right].\end{aligned}$$

   Using (3a),

$$\left|\frac{\mathrm{d}}{\mathrm{d}p}\delta_n(b^m a^{n-m})\right| = p^{n-m-1}(1-p)^{m-1}|n-m-np|.$$

   Before we can use (4b) we must verify our assumption about the interval; that is, we require

$$\left|\frac{\mathrm{d}}{\mathrm{d}p}\delta_n(b^m a^{n-m})\right| \ll \frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^{n-m}).$$

   Now,

$$\begin{aligned}\frac{\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^{n-m})}{\left|\frac{\mathrm{d}}{\mathrm{d}p}\delta_n(b^m a^{n-m})\right|} &= \frac{(1-p)^m\left[\frac{m(1-p^{n-m-1})}{1-p} + np^{n-m-1}\right]}{p^{n-m-1}(1-p)^{m-1}|n-m-np|} \\ &= \frac{m(1-p^{n-m-1} + (1-p)np^{n-m-1}}{p^{n-m-1}|n-m-np|} \\ &= \frac{m}{p^{n-m-1}|n-m-np|} + \frac{n-m-np}{|n-m-np|} \\ &= \frac{m}{p^{n-m-1}|n-m-np|} \pm 1 \\ &\gg 1.\end{aligned}$$

   So $\left|\frac{\mathrm{d}}{\mathrm{d}p}\delta_n(b^m a^{n-m})\right| \ll \frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^{n-m})$ for all values except $p \to 1$.
   Now using (4b),

$$\begin{aligned}\frac{2\delta_n(b^m a^{n-m})}{\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^{n-m})} &= \frac{2p^{n-m}(1-p)^m}{(1-p)^m\left[\frac{m(1-p^{n-m-1})}{1-p} + np^{n-m-1}\right]} \\ &= \frac{2p^{n-m}(1-p)}{m - mp^{n-m-1} + np^{n-m-1} - np^{n-m}}.\end{aligned}$$

   In the interval $0 < p \leq \frac{1}{2}$, this has a maximum when $p = \frac{1}{2}$, and has the value $m2^{n-m-1} + n$. Therefore,

$$\begin{aligned} 2^w &\geq m2^{n-m-1}+n \\ 2^w &\geq m2^{n-m-1} \\ n &\leq w+m+1-\lg m. \end{aligned}$$

For $p > \frac{1}{2}$ the situation is much more complex, and no general bound is available.

3. If $S = b^m a b^l$ then $\delta_n(b^m a b^l) = p(1-p)^{m+l}$ and $\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a b^l) = (m+1)(1-p)^m$. We require

$$\begin{aligned} \left(\frac{1}{2}\right)^w &\leq \frac{2\delta_n(b^m a b^l)}{\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a b^l)} = \frac{p(1-p)^{m+l}}{(m+1)(1-p)^m} \\ &= \frac{2p(1-p)^l}{m+1} \\ &\leq \frac{2(1-p)^l}{m+1}. \end{aligned}$$

The larger l is, the more accurately approximation (4b) is satisfied. This can be seen by comparing $\left|\frac{\mathrm{d}\delta_n(b^m a b^l)}{\mathrm{d}p}\right| = (1-p)^{m+l-1}(1-p-mp-lp)$ to $\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a b^l)$.
Therefore, since $n = m+l+1$,

$$n \leq m+1+\frac{w+1-\lg(m+1)}{-\lg(1-p)}.$$

Clearly, as $p \to 0$ an infinite number of symbols are required. Otherwise $m+1+O\left(\frac{w+1-\lg(m+1)}{-\lg(1-p)}\right)$ symbols are required.

4. If $S = b^m a^l b$ then $\delta_n(b^m a^l b) = p^l(1-p)^{m+1}$ and $\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^l b) = [lp^{l-1}(1-p)+(1-p^l)m](1-p)^{m+1}$. We require

$$\left(\frac{1}{2}\right)^w \leq \frac{2\delta_n(b^m a^l b)}{\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^l b)} = \frac{2p^l(1-p)^2}{lp^{l-1}(1-p)+(1-p^l)m}.$$

Solving for m we get

$$m \leq \frac{2^{w+1}p^l - lp^{l-1}(1-p)}{(1-p^l)}.$$

Provided $p \leq \frac{1}{2}$ and since $n = m+l+1$,

$$n \leq l+1+O(2^w p^l).$$

The cases where

$$\left|\frac{\mathrm{d}\delta_n(b^m a^l b)}{\mathrm{d}p}\right| = p^{l-1}(1-p)^m|l-lp-mp-p|$$

is much less than $\frac{\mathrm{d}}{\mathrm{d}p}h_n(b^m a^l b)$ are precisely those where $p \leq \frac{1}{2}$. These sequences cannot be used when $p \to 1$.

Δ

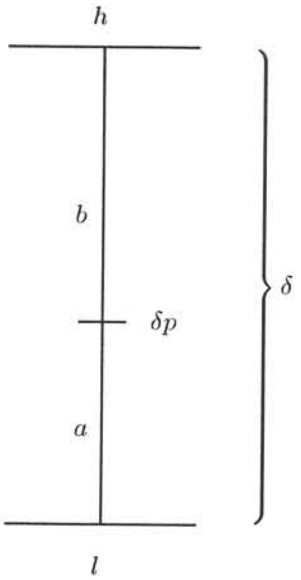

Figure 1: A binary arithmetic coder.

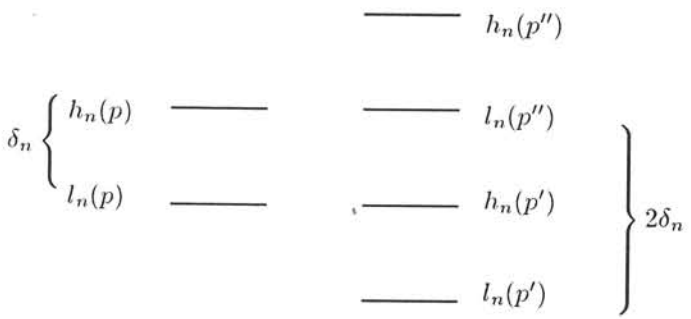

Figure 2: Range of decoding estimates. $\Delta p = p'' - p'$.

w	maximum symbols needed
6	5
7	6
8	7
9	8
10	9
11	10
12	11
13	12
14	13
15	14

Figure 3: Experimental results on the number of symbols needed to uniquely determine the state of an arithmetic coder for the sequence $(ab)^*$.