Decoding Running Key Ciphers

Sravana Reddy*
Department of Computer Science
The University of Chicago
1100 E. 58th Street
Chicago, IL 60637, USA
sravana@cs.uchicago.edu

Kevin Knight
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
knight@isi.edu

Abstract

There has been recent interest in the problem of decoding letter substitution ciphers using techniques inspired by natural language processing. We consider a different type of classical encoding scheme known as the *running key cipher*, and propose a search solution using Gibbs sampling with a word language model. We evaluate our method on synthetic ciphertexts of different lengths, and find that it outperforms previous work that employs Viterbi decoding with character-based models.

1 Introduction

The running key cipher is an encoding scheme that uses a secret key R that is typically a string of words, usually taken from a book or other text that is agreed upon by the sender and receiver. When sending a plaintext message P, the sender truncates R to the length of the plaintext. The scheme also relies on a substitution function f, which is usually publicly known, that maps a plaintext letter p and key letter r to a unique ciphertext letter c. The most common choice for f is the *tabula recta*, where $c = (p + r) \mod 26$ for letters in the English alphabet, with A $= 0$, B $= 1$, and so on.

To encode a plaintext with a running key, the spaces in the plaintext and the key are removed, and for every $0 \leq i < |P|$, the ciphertext letter at position i is computed to be $C_i \leftarrow f(P_i, R_i)$. Figure 1 shows an example encoding using the *tabula recta*.

For a given ciphertext and known f, the plaintext uniquely determines the running key and vice versa. Since we know that the plaintext and running key are both drawn from natural language, our objective function for the solution plaintext under some language model is:

$$\hat{P} = \arg\max_{P} \log \Pr(P) \Pr(R_{P,C}) \qquad (1)$$

where the running key $R_{P,C}$ is the key that corresponds to plaintext P and ciphertext C.

Note that if $R_{P,C}$ is a perfectly random sequence of letters, this scheme is effectively a 'one-time pad', which is provably unbreakable (Shannon, 1949). The knowledge that both the plaintext and the key are natural language strings is important in breaking a running key cipher.

The letter-frequency distribution of running key ciphertexts is notably flatter than than the plaintext distribution, unlike substitution ciphers where the frequency profile remains unchanged, modulo letter substitutions. However, the ciphertext letter distribution is not uniform; there are peaks corresponding to letters (like I) that are formed by high-frequency plaintext/key pairs (like E and E).

2 Related Work

2.1 Running Key Ciphers

Bauer and Tate (2002) use letter n-grams (without smoothing) up to order 6 to find the most probable plaintext/key character pair at each position in the ciphertext. They test their method on 1000-character ciphertexts produced from plaintexts and keys extracted from Project Gutenberg. Their accuracies range from 28.9% to 33.5%, where accuracy is measured as the percentage of correctly decoded char-

*Research conducted while the author was visiting ISI.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 80–84,
Jeju, Republic of Korea, 8-14 July 2012. ©2012 Association for Computational Linguistics

Figure 1: Example of a running key cipher. Note that key is truncated to the length of the plaintext.
Plaintext – linguistics is fun, *Running Key* – colorless green ideas, *tabula recta* substitution where $C_i \leftarrow (P_i + R_i) \mod 26$

Plaintext:	L	I	N	G	U	I	S	T	I	C	S	I	S	F	U	N
Running Key:	C	O	L	O	R	L	E	S	S	G	R	E	E	N	I	D
Ciphertext:	N	W	Y	U	L	T	W	L	A	I	J	M	W	S	C	Q

acters. Such figures are too low to produce readable plaintexts, especially if the decoded regions are not contiguous. Griffing (2006) uses Viterbi decoding and letter 6-grams to improve on the above result, achieving a median 87% accuracy over several 1000-character ciphertexts. A key shortcoming of this work is that it requires searching through about 26^5 states at each position in the ciphertext.

2.2 Letter Substitution Ciphers

Previous work in decipherment of classical ciphers has mainly focused on letter substitution. These ciphers use a substitution table as the secret key. The ciphertext is generated by substituting each letter of the plaintext according to the substitution table. The table may be homophonic; that is, a single plaintext letter could map to more than one possible ciphertext letter. Just as in running key ciphers, spaces in the plaintext are usually removed before encoding.

Proposed decipherment solutions for letter substitution ciphers include techniques that use expectation maximization (Ravi and Knight, 2008), genetic algorithms (Oranchak, 2008), integer programming (Ravi and Knight, 2009), A* decoding (Corlett and Penn, 2010), and Bayesian learning with Dirichlet processes (Ravi and Knight, 2011).

2.3 Vigenère Ciphers

A scheme similar to the running key cipher is the Vigenère cipher, also known as the periodic key cipher. Instead of a single long string spanning the length of the plaintext, the key is a short string – usually but not always a single word or phrase – repeated to the length of the plaintext. Figure 2 shows an example Vigenère cipher encoding. This cipher is less secure than the running key, since the short length of the key vastly reduces the size of the search space, and the periodic repetition of the key leaks information.

Recent work on decoding periodic key ciphers perform Viterbi search on the key using letter n-gram models (Olsen et al., 2011), with the assumption that the length of the key is known. If unknown, the key length can be inferred using the Kasiski Test (Kasiski, 1863) which takes advantage of repeated plaintext/key character pairs.

3 Solution with Gibbs Sampling

In this paper, we describe a search algorithm that uses Gibbs Sampling to break a running key cipher.

3.1 Choice of Language Model

The main advantage of a sampling-based approach over Viterbi decoding is that it allows us to seamlessly use word-based language models. Lower order letter n-grams may fail to decipher most ciphertexts even with perfect search, since an incorrect plaintext and key could have higher likelihood under a weak language model than the actual message.

3.2 Blocked Sampling

One possible approach is to sample a plaintext letter at each position in the ciphertext. The limitation of such a sampler for the running key problem is that is extremely slow to mix, especially for longer ciphertexts: we found that in practice, it does not usually converge to the optimal solution in a reasonable number of iterations even with simulated annealing. We therefore propose a blocked sampling algorithm that samples *words* rather than letters in the plaintext, as follows:

1. Initialize randomly $P := p_1 p_2 \ldots p_{|C|}$, fix R as the key that corresponds to P, C
2. Repeat for some number of iterations
   (a) Sample spaces (word boundaries) in P according to $\Pr(P)$
   (b) Sample spaces in R according to $\Pr(R)$
   (c) Sample each word in P according to $\Pr(P)\Pr(R)$, updating R along with P
   (d) Sample each word in R according to $\Pr(P)\Pr(R)$, updating P along with R

Figure 2: Example of a Vigenère cipher cipher, with a 5-letter periodic key, repeated to the length of the plaintext. *Plaintext* – linguistics is fun, *Periodic Key* – green, *tabula recta* substitution.

Plaintext:	L	I	N	G	U	I	S	T	I	C	S	I	S	F	U	N
Running Key:	G	R	E	E	N	G	R	E	E	N	G	R	E	E	N	G
Ciphertext:	R	Z	R	K	H	O	J	X	M	P	Y	Z	W	J	H	T

3. Remove spaces and return P, R

Note that every time a word in P is sampled, it induces a change in R that may not be a word or a sequence of words, and vice versa. Sampling word boundaries will also produce hypotheses containing non-words. For this reason, we use a word trigram model linearly interpolated with letter trigrams (including the space character).[1] The interpolation mainly serves to smooth the search space, with the added benefit of accounting for out-of-vocabulary, misspelled, or truncated words in the actual plaintext or key. Table 1 shows an example of one sampling iteration on the ciphertext shown in Figure 1.

Table 1: First sampling iteration on the ciphertext NWYULTWLAIJMWSCQ

Generate P, R with letter trigrams	P: WERGATERYBVIEDOW R: RSHOLASUCHOESPOU
Sample spaces in P	P: WERGAT ER YB VIEDOW
Sample spaces in R	R: RS HOLASUCHOES POU
Sample words in P	P: ADJUST AN MY WILLOW R: NT PATAWYOKNEL HOU
Sample words in R	P: NEWNXI ST HE SYLACT R: AS CHOLESTEROL SAX

4 Experiments

4.1 Data

We randomly select passages from the Project Gutenberg and Wall Street Journal Corpus extracts that are included in the NLTK toolkit (Bird et al., 2009). The passages are used as plaintext and key pairs, and combined to generate synthetic ciphertext data. Unlike previous works which used constant-length ciphertexts, we study the effect of message length on decipherment by varying the ciphertext length (10, 100, and 1000 characters).

Our language model is an interpolation of word trigrams and letter trigrams trained on the Brown Corpus (Nelson and Kucera, 1979), with Kneser-Ney smoothing. We fixed the word language model interpolation weight to $\lambda = 0.7$.

[1] $\Pr(P) = \lambda \Pr(P|\text{word LM}) + (1 - \lambda) \Pr(P|\text{letter LM})$, and similarly for $\Pr(R)$.

4.2 Baseline and Evaluation

For comparison with the previous work, we re-implement Viterbi decoding over letter 6-grams (Griffing, 2006) trained on the Brown Corpus. In addition to decipherment accuracy, we compare the running time in seconds of the two algorithms. Both decipherment programs were implemented in Python and run on the same machines. The Gibbs Sampler was run for 10000 iterations.

As in the Griffing (2006) paper, since the plaintext and running key are interchangeable, we measure the accuracy of a hypothesized solution against the reference as the max of the accuracy between the hypothesized plaintext and the reference plaintext, and the hypothesized plaintext and the reference key.

4.3 Results

Table 2 shows the average decipherment accuracy of our algorithm and the baseline on each dataset. Also shown is the number of times that the Gibbs Sampling search failed – that is, when the algorithm did not hypothesize a solution that had a probability at least as high as the reference within 10000 iterations.

It is clear that the Gibbs Sampler is orders of magnitude faster than Viterbi decoding. Performance on the short (length 10) ciphertexts is poor under both algorithms. This is expected, since the degree of message uncertainty, or *message equivocation* as defined by Shannon, is high for short ciphertexts: there are several possible plaintexts and keys besides the original that are likely under an English language model. Consider the ciphertext WAEEXFPROV which was generated by the plaintext segment ON A REFEREN and key INENTAL AKI. The algorithm hypothesizes that the plaintext is THE STRAND S and key DTAME OPELD, which both receive high language model probability.

Table 2: Decipherment accuracy (proportion of correctly deciphered characters). Plaintext and key sources for the ciphertext test data were extracted by starting at random points in the corpora, and selecting the following n characters.

Length of ciphertext	Plaintext and key source	# Cipher-texts	Average Accuracy		Avg. running time (sec)		# Failed Gibbs searches
			Viterbi	**Gibbs**	Viterbi	**Gibbs**	
10	Project Gutenberg	100	14%	**17%**	1005	**47**	5
	Wall Street Journal	100	10%	**26%**	986	**38**	2
100	Project Gutenberg	100	27%	**42%**	10212	**236**	19
	Wall Street Journal	100	22%	**58%**	10433	**217**	12
1000	Project Gutenberg	100	63%	**88%**	112489	**964**	32
	Wall Street Journal	100	60%	**93%**	117303	**1025**	25

Table 3: Substitution function parameterized by the keyword, CIPHER. $f(p, r)$ is the entry in the row corresponding to p and the column corresponding to r.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	C	I	P	H	E	R	A	B	D	F	G	J	K	L	M	N	O	Q	S	T	U	V	W	X	Y	Z
B	I	P	H	E	R	A	B	D	F	G	J	K	L	M	N	O	Q	S	T	U	V	W	X	Y	Z	C
C	P	H	E	R	A	B	D	F	G	J	K	L	M	N	O	Q	S	T	U	V	W	X	Y	Z	C	I
...																										

However, on the long ciphertexts, our algorithm gets close to perfect decipherment, surpassing the Viterbi algorithm by a large margin.[2] Accuracies on the Wall Street Journal ciphertexts are higher than on the Gutenberg ciphertexts for our algorithm, which may be because the latter is more divergent from the Brown Corpus language model.

5 Future Work

5.1 Unknown substitution functions

Some running key ciphers also use a secret substitution function f rather than the *tabula recta* or another known function. In typical cases, these functions are not arbitrary, but are parameterized by a secret keyword that mutates the *tabula recta* table. For example, the function with the keyword CIPHER would be the substitution table shown in Table 3. Decoding a running key ciphertext under a latent substitution function is an open line of research. One possibility is to extend our approach by sampling the keyword or function in addition to the plaintext.

5.2 Exact search

Since some the errors in Gibbs Sampling decipherment are due to search failures, a natural extension of this work would be to adapt Viterbi search or other exact decoding algorithms like A* to use word-level language models. A naive implementation of Viterbi word-based decoding results in computationally inefficient search spaces for large vocabularies, so more sophisticated methods or heuristics will be required.

5.3 Analysis of Running Key Decipherment

While there has been theoretical and empirical analysis of the security of letter substitution ciphers of various lengths under different language models (Shannon, 1949; Ravi and Knight, 2008), there has been no similar exposition of running key ciphers, which we reserve for future work.

6 Conclusion

We propose a decipherment algorithm for running key ciphers that uses Blocked Gibbs Sampling and word-based language models, which shows significant speed and accuracy improvements over previous research into this problem.

Acknowledgments

We would like to thank Sujith Ravi for initial experiments using Gibbs sampling, and the anonymous reviewers. This research was supported in part by NSF grant 0904684.

[2]The accuracies that we found for Viterbi decoding are lower than those reported by Griffing (2006), which might be because they use an in-domain language model.

References

Craig Bauer and Christian Tate. 2002. A statistical attack on the running key cipher. *Cryptologia*, 26(4).

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Eric Corlett and Gerald Penn. 2010. An exact A* method of deciphering letter-substitution ciphers. In *Proceedings of ACL*.

Alexander Griffing. 2006. Solving the running key cipher with the Viterbi algorithm. *Cryptologia*, 30(4).

Friedrich Kasiski. 1863. *Die Geheimschriften und die Dechiffrir-Kunst*. E. S. Mittler und Sohn.

Francis Nelson and Henry Kucera. 1979. *The Brown Corpus: A Standard Corpus of Present-Day Edited American English*. Brown University.

Peder Olsen, John Hershey, Steven Rennie, and Vaibhava Goel. 2011. A speech recognition solution to an ancient cryptography problem. Technical Report RC25109 (W1102-005), IBM Research.

David Oranchak. 2008. Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems. In *Proceedings of the Conference on Genetic and Evolutionary Computation*.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of EMNLP*.

Sujith Ravi and Kevin Knight. 2009. Attacking letter substitution ciphers with integer programming. *Cryptologia*, 33(4).

Sujith Ravi and Kevin Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proceedings of ACL*.

Claude Shannon. 1949. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4).