computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

BREAKING BLOCK AND PRODUCT CIPHERS

Albert H. Carlson [1], Robert E. Hiromoto [2], Richard B. Wells [3]

[1] Fontbonne University, St. Louis, MO 63105, USA, acarlson@fontbonne.edu,
[2] University of Idaho, Idaho Falls, ID 83402, USA, hiromoto@uidaho.edu,
[3] University of Idaho, Moscow, ID 83844-1010 USA, rwells@uidaho.edu

Abstract: The security of block and product ciphers is considered using a set theoretic estimation (STE) approach to decryption. Known-ciphertext attacks are studied using permutation (P) and substitution (S) keys. The blocks are formed from two (2) alphabetic characters (meta-characters) where the applications of P and S upon the ASCII byte representation of each of the two characters are allowed to cross byte boundaries. The application of STE forgoes the need to employ chosen-plaintext or known-plaintext attacks. *Copyright © Research Institute for Intelligent Computer Systems, 2013. All rights reserved.*

Keywords: Set theoretic estimation, block and product cipher, byte boundaries, known ciphertext attack, meta-characters.

1. INTRODUCTION

Cryptanalysis is the study of decryption techniques of encrypted information, which involves determining the secret key to the encryption. The key is a "secret" parameter that adds "noise" [1] to the original message and makes the message unreadable.

The process of decryption is a set of iterative applications of *a priori* knowledge applied to a noisy input in an effort to recover the message from the noise. The noise is intentionally injected into the message so that

$$C = E(M, k),$$

where C is the cipher text and $E(M, k)$ is the encryption function using a key, k, as a noise generator for the message M.

Noise generators take on numerous forms. The block and product ciphers apply multiple keys to the same data, one after the other. That is, the first cipher is applied to the plain text. The second cipher is applied to the ciphertext that results from applying the first cipher to the plaintext.

In this paper, we apply a rule-based set theoretic estimation (STE) [2] approach to capture the properties of *m*-grams that are derived from a *stylistic* use of the English language. We treat the resulting collection of *m*-grams as an *a priori* property set that is used to analyze the frequency distribution of *m*-grams symbols and as predictors for either *allowed* or *forbidden* letter combinations.

In the remainder of the paper, we provide background material on permutation, *P*, ciphers and their relation to the substitution, *S*, cipher. Results of STE applied to a block P and block S decryption algorithm is presented.

2. BACKGROUND

All modern block ciphers are product ciphers. The product cipher combines several encryption operations to provide both confusion and diffusion. Confusion substitutes one character for another while diffusion distributes information across the encrypted message [3]. Block ciphers of *PS* and *PSP* type are composed of a *P* cipher with a key representing a mapping. The key space for the *P* cipher is b!, where b is the number of bits in the block [4, 5, 6]. Similarly, the *S* cipher maps an alphabet A to another group of symbols, A′ in some manner, $A \mapsto A'$. For the *S* cipher, it is possible that A = A′. The key space is |A|!, where |A| is the number of symbols in the alphabet.

Another way to increase the key space is to increase the number of times a message is encrypted. Shannon concluded that compounding ciphers could increase the key space for a message, and as a result, increase its security. When the product ciphers use a good mixing transformation [5, 9], the number of keys increases by multiplying the key space for each cipher in the product cipher. The key space for a product cipher with p ciphers is given by

$$\left|K_{productcipher}\right| = \prod_{i=1}^{p} K_i$$

Since Shannon introduced the concept of increasing security by compounding ciphers, it has generally been accepted that product ciphers of the form *PSP* [1, 3, 4] are more secure than a cipher employing only a permutation (See Fig. 1) or a substitution cipher. This is not true for block ciphers (See Fig. 2) whose encryption algorithms end at byte (character) boundaries and are encoded using ASCII.

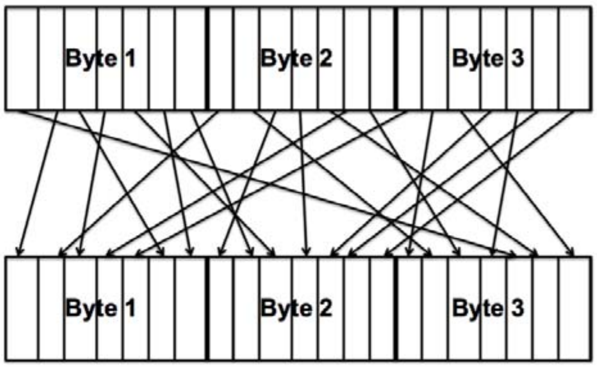


Fig. 1 – Permutation Cipher Mapping Key.

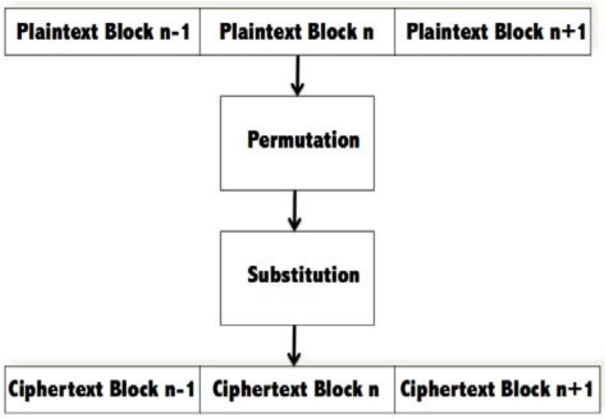


Fig. 2 – PS Type Cipher.

Blocks of integral character size suffer from a significant weakness; that is, information is confined within the block. As such, we suspect that as the block size of the *PSP* cipher increases above two symbols the additional security gained is insignificant as compared to a simple substitution cipher of the same block size. Combining both *P* and *S* ciphers into a block cipher using identical block boundaries results in a key space of size b!|A|!, where b is the number of bits in the block and |A| is the number of symbols in the alphabet A. Extending the block cipher to a *PSP* cipher, the size of the key space is b!|A|!b!. Let X be a compound symbol with an ordered n-tuple of characters $< x_0, x_1, \cdots, x_i >$ regarded as comprising a single (meta) symbol, where $< x_0, x_1, \cdots, x_i > x_i \in A_\lambda$ and A_λ is the alphabet of language λ. The language λ is a meta-language composed of meta-s-characters embedded in the same natural language, where *s* is the number of alphabetic symbols that make up a meta-character. A different meta-language exists for each meta-s-character size. For example, a meta-7-character 'flyinsk' is drawn from a meta-language with an alphabet using 'f', 'l', 'y', 'i', 'n', 's', and 'k'. In this representation, information is not restricted to the same byte of data in which it originated. For example, permutations on multi-byte blocks allow for any bit in the block to be permuted to any other location inside the block, even across byte boundaries. Ciphers that diffuse data are specifically chosen so that they allow diffusion across byte boundaries as depicted in Fig. 1. This is a much more difficult problem than byte-wise decryption. The problem is so difficult that cryptanalysts have chosen to create different attacks rather than deal with the diffusion [4, 12]. The algorithm described in this chapter deals directly with the diffusion across byte boundaries.

A *PSP* cipher is idempotent to an *S* cipher with identical block boundaries. This result follows by taking symbols in a block cipher as a compound symbol of the same size and having the same boundaries as the cipher block. Further, let the S block cipher be applied to the same compound symbol. Then *P* and *S* ciphers are equivalent within a symbol boundary. Therefore, *PSP* reduces to *SSS*. *S* ciphers are idempotent and associative with each other. Therefore *SSS* cipher reduces to a single *S* cipher. It can also be demonstrated that *P* reduces to *S*; however, *S* does not necessarily reduce to *P* [13, 14].

3. BCBB ALGORITHM

Defeating permutation across block boundaries requires treating the language as if the block of characters was actually a single character of a different language. The block comprises a character made up of characters, or a "meta-character," which is part of a "meta-language."

The Block Cross Byte Boundary (BCBB) algorithm is designed to decrypt block ciphers of *PS* and *PSP* type. The BCBB algorithm is based on the framework of STE as described earlier. The algorithm starts by reading in both the encrypted text and the data sets needed for decryption. The algorithm then sets up a solution matrix of possible mappings from ciphertext to plaintext meta-s-characters. The mapping is stored by means of a hash-table that associates invalid key mappings for a particular ciphertext meta-s-character to a plaintext meta-s-character. Mappings that do not appear in the hash-table are still considered possible. Lists of meta-s-characters seen in the encrypted text and mappings found to be part of the key are also maintained.

When choosing property sets for use in decrypting multi-byte product ciphers, data sets are based on meta-s-characters of the block size being decrypted. In this case, only the meta-s-character frequency and allowed meta-s-gram (meta(s,m)) sets are used for decryption tests. For example, the text composed of 'theonl' is a meta(3,2) made up of two different meta-3-characters 'the' and 'onl'. Note that a meta(s,m) is equivalent to an m-gram of the size $s * m$.

The first property set applied to the BCBB algorithm for a product cipher is the global frequency property set. Global redundancy is applied only once. Global frequency is the frequency of characters in the meta-alphabet found in the message. Following the Law of Large Numbers [6], the larger the message, the more likely the meta-s-character frequency from the message will accurately reflect the meta-language alphabet frequency. The meta-2-character frequencies are studied by [13]. Both high and low frequency characters are of interest in this set. A large division in the data collected occurs between the first 10 and subsequent members of the meta-2-character set. This division is referred to as the "high frequency threshold." The top ten meta-2-characters on the frequency list are dubbed "high frequency" meta-2-characters. When interpreting data returned from the BCBB algorithm, higher frequency meta-2-character combinations are assumed to belong to the top ten-frequency set.

Similarly, the corpus identifies a set of low frequency characters. "Low frequency" meta-2-characters fall within the bottom 5% of the meta-2-character frequency list. Again, finding a frequency where it is possible to distinguish between sets of meta-2-characters sets the threshold. Any meta-2-character occurring more frequently than the threshold cannot be mapped to any member inside the low frequency set. Thus the mapping(s) can be eliminated. Together the initial application of the global frequency set results in a reduction of mappings in the solution matrix.

Other property sets selected for use are the frequency of meta-s-characters and the forbidden meta(s,m) set. The use of meta(s,m) sets subsumes the redundancy and multiple letter sets used for the algorithm, indicating that no additional property sets need to be applied.

Once the solution structure is set up, the algorithm began to eliminate mappings. The mapping elimination procedure follows the following steps: 1) the first property is applied to the global meta-s-character frequency data. The entire message is processed and then compared against a normalized global frequency list, 2) high frequency meta-s-characters passing the frequency threshold are mapped to a select set of plaintext characters that contained the only characters seen above the threshold, 3) the message is checked for redundant meta-s-characters in each meta(s, m) gram for all m selected for evaluation. Redundancy in a meta(s,m)-gram yields low entropy information. Therefore, processing the redundancies further eliminates mappings, 4) following frequency and redundancy checks, the main body of message analysis begins. A meta-s-character is read from the message and the meta(s,m)-gram set is applied to the portion of the message that is being analyzed. This process is iterated upon as new meta-s-characters are introduced from the message and reanalyzed until the message is decrypted or there are no more meta-s-characters left for evaluation. This entire process is called the *relaxation* stage. Additional details of the BCBB algorithm are found at [13].

4. EXAMPLE ATTACKS

Consider a message encrypted by a permutation cipher, followed by a substitution cipher (referred to as *PS*). Without loss of generality, let the plain text consist only of lower case alphabetic English characters with all spaces that delineate words within the message, and punctuations removed. The message uses standard ASCII encoding and the permutation employs a two-character (byte) block. Assume that the byte and block boundaries are known for the encrypted message. Blocks are restricted to coincide with byte boundaries. We treat the entire block as a single meta-character in a meta-language.

Each meta-symbol is analyzed for common language statistics based on their appearance in English. Language statistics constitute property sets that can be exploited using Set Theoretic Estimation (STE) and Set Methodology [2, 10]. The STE property sets used in the attack are listed in Table 1.

The ASCII representation used in this example requires the use of three special (constant) bits that indicate the case and alphabetic letters used. These bits are termed *static* bits, and are also permuted under the mapping of the key. The location of these bits can be determined by performing an XOR function between two blocks. Any bits that do not change are identified as *static* bits. A block diagram illustrating the *static* bit algorithm is shown in Fig. 3.

The resulting information is used as one of the XOR operators and additional blocks are XOR'ed with the composite information. Since ASCII employs the same three most significant bits, nine bits are located (i.e., three bits per a two letter block), leaving 10 remaining bits to be mapped.

Table 1. STE Property Sets for Block Cipher.

Label	Property Sets
Φ_0	English Language
Φ_1	Static Bits
Φ_2	Byte Grouping
Φ_3	Bit Exclusivity
Φ_4	3-grams Forbidden

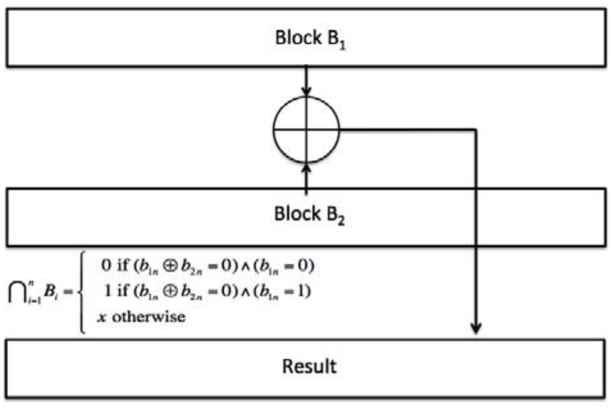

Block B_1
Block B_2
$\bigcap_{i=1}^{n} B_i = \begin{cases} 0 \text{ if } (b_{1_n} \oplus b_{2_n} = 0) \wedge (b_{1_n} = 0) \\ 1 \text{ if } (b_{1_n} \oplus b_{2_n} = 0) \wedge (b_{1_n} = 1) \\ x \text{ otherwise} \end{cases}$
Result

Fig. 3 – Static Bit Algorithm.

The remaining 10 bits are taken five at a time and permuted internally to identify all legitimate ASCII characters. The permutation ordering is defined relative to the corresponding block of the cipher text. Using the internal bit representation for ASCII letters, these letters are ordered in ascending order. If the group of five bits cannot be arranged to form a valid pattern for a letter, it is discarded. At the completion of this process, there should be at most two five-bit groupings, each containing one or several valid letters that arise through a unique permutation mapping. In addition, certain bit combinations are not allowed in the ASCII representation. Patterns of five "1"s or five "0"s, for example, are not valid and, therefore, disregarded. This is a property set that is referred to as *bit exclusivity*. This entire process is applied to *n* two-byte block (the order is not important) in the message.

At the end of processing *n* two-byte blocks, the resulting allowed permutation mappings in each block most likely contain the correct key and numerous spurious keys. At this stage, each bit within each byte of a block is assigned a unique index about which valid letter permutations are identified. In the next step, the *derived* information (allowed permutations) is analyzed by intersecting all possible permutations in each block across all block bytes. This step takes into account the fact that the same mapping key encrypts each block. With this in mind, the resulting intersections should determine the set of equivalent permutation mappings for the encryption key. At this point, the correct key that determines the correct ordering of letters is still unknown.

The final step of the process reduces to the application of the *Last-Man-Standing* technique described above. Each equivalent permutation contains a string of valid letters but only one sequence is the correct ordering sought. This problem now becomes identical to the substitution cipher problem that was solved by the application of *m*-grams. The nature of the two-byte block, therefore, suggests the application of 3-grams to determine correct ordering of letters.

Fig. 4 illustrates the intersections of property sets as applied in STE to decryption. The requirements for an STE application are: 1) the problem must mapped one-to-one and onto the solution domain; 2) the problem must have a bounded error for a given input; 3) each property set is composed of distinct elements; and 4) property sets differentiate inputs based on set membership.

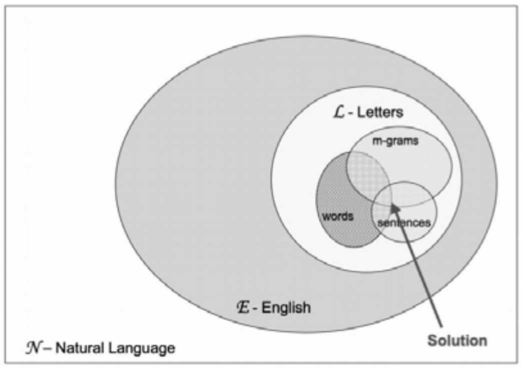


Fig. 4 – STE Application.

Any information (rules, constraints) known about the problem is encoded in sets in the solution space. Each rule or constraint is represented by its own unique set. Estimates may be a member of more than one set, but must be in at least one set to be considered. Information known about both the inputs and about the rules is treated similarly. Rules are expressed as assertions in STE. An assertion A takes the set of possible inputs and gives a set of resulting outputs, or solutions, for the operation, O, as specified in the rule.

The *m*-grams used in our STE approach are drawn from a survey of English prose styles from 1600 - 2000 AD [6] that are found in Project Gutenberg [11]. Each symbol in this meta-language represents an *m*-gram of English. This definition reduces the number of symbols in the meta-language alphabet to the number of unique *m*-grams allowed in English. All encryption occurs within a single meta-symbol. Results for block S and block P are shown in Table 2.

4.1. META-2-GRAM EXAMPLE

Consider the following encrypted input text:

fa_63_aa _fa_85_fa_63_aa_fa

where the plaintext is given by:

The_truth_is_the_truth

Table 2 gives the corresponding key for the encryption.

Table 2. Encryption Key.

Key
63 → et
85 → is
aa → ru
fa → th

Applying the key to the encrypted input text results in the recovered plaintext:

th_et_ru_th_is_th_et_ru_th

The BCBB algorithm reads the encrypted meta-character input text from left to right as illustrated in Table 3.1.

Table 3.1. Encrypted Message Input String.

Read 1st	fa
Read 2nd	fa_63
Read 3rd	fa_63_aa
Read 4th	fa_63_aa _fa
Read 5th	fa_63_aa _fa_85
• • •	• • •

The decryption method takes the following form:

- Form sequences of two allowed 2-grams meta(2,2), e.g.,

etru, isis, isth, ruth, thet, this

- Form a correspondence between *like* and *unlike* patterns using symbols such as A and B (Table 3.2) where A is different from B and vice versa.

Table 3.2. Meta(2,2) + Pattern Matching.

etru	isis	isth	ruth	thet	this
↕	↕	↕	↕	↕	↕
AB	AA	AB	AB	AB	AB

- Form sequences of three allowed 2-grams, meta(2,3) such as

etruth, isthet, ruthis, thetru, thisth

- Form their associated patterns of meta(2,3) in Table 4.

Table 4. Meta(2,3) + Pattern Matching.

etruth	isthet	ruthis	thetru	thisth
↕	↕	↕	↕	↕
ABC	ABC	ABC	ABC	ABA

Finally for a maximum window size of four then

- Form sequences of four allowed 2-grams meta(2,4)

etruthis, isthetru, ruthisth, thetruth, thisthet

- Form their associated matching patterns of (2,4) in Table 5.

Table 5. Meta(2,4) + Pattern Matching.

etruthis	isthetru	ruthisth	thetruth	thisthet
↕	↕	↕	↕	↕
ABCD	ABCD	ABCB	ABCA	ABAD

- Form sequence of encrypted input meta characters (window sizes two – four)
- Associate their patterns with patterns of the allowed sequence of m 2-grams (above), where m = {2, 3, 4}
- Use pattern matching to converge encrypted meta-character to plaintext characters

Table 6 illustrates the decryption matrix (d-matrix) before the start of the relaxation steps.

Table 6. D-Matrix.

	fa	63	aa	85
et	.	.	.	.
is	.	.	.	.
ru	.	.	.	.
th	.	.	.	.

where the decryption is built up from the meta-2-character inputs.

In this example, the input is read from left to right:

fa_63_aa _fa_85_fa_63_aa_fa

The relaxation steps are applied after each meta-character read of the inputs. In Read 1st, the meta-character fa is inserted into the d-matrix. Read 1st does not provide sufficient information to make an analysis. Thus relaxation waits on Read 2nd at which point the meta-character 63 is inserted into the d-matrix. The relaxation procedure can now form the combination fa63 and the associated pattern AB. Checking the allowed two 2-gram patterns: AB, AA, AB, AB, AB, AB from Table 3.2, we find that all patterns AB are possible so no change to the d-matrix is made. In the Read 3rd step, the meta-character aa is inserted into the d-matrix. At this point, the relaxation procedure forms the sequence fa63aa and assigns it the pattern ABC. This pattern is checked against the allowed three 2-gram, meta(2,3), in Table 4 with patterns: ABC, ABC, ABC, ABC, and ABA. Again since the pattern ABC occurs multiple times, no change is made to the d-matrix. After Read 4th, the relaxation step forms the sequence fa63aafa and its corresponding pattern ABCA. Checking allowed four 2-gram patterns, meta(2,4), in Table meta-4, we find the patterns: ABCD, ABCD, ABCB, ABCA, ABAD. Since there is only one occurrence of the pattern ABCA, this implies that A = 'fa' = 'th.' The mapping 'fa' to 'th' is termed a "*partial binding,*" that results in updating the d-matrix.

	fa	63	aa	85
et	0	.	.	.
is	0	.	.	.
ru	0	.	.	.
th	1	0	0	0

Next the meta-character 85 is read into the d-matrix. Since we have discovered that 'th' maps to 'fa', we form the following sequences: 63aa, 63aafa, 63aafa85 and their associated patterns AB, ABC, ABCD. Note that we have set the relaxation window size for analysis to be 4, which indicates that max (m) = 4. Using these patterns, we attempt to identify additional partial bindings: 63aa – AB is not bound to 'th', since 'fa' = 'th'; therefore, etru is the only mapping. Thus '63' = 'et' and 'aa' = 'ru' and we update the d-matrix to indicate the new partial bindings.

	fa	63	aa	85
et	0	1	0	0
is	0	0	0	.
ru	0	0	1	0
th	1	0	0	0

The d-matrix now has only one unique unresolved mapping; therefore, set '85' = 'is'

	fa	63	aa	85
et	0	1	0	0
is	0	0	0	1
ru	0	0	1	0
th	1	0	0	0

4.2 BCBB DECRYPTION RESULTS

Table 7, is a list of the books and authors that we used in testing the BCBB algorithm. Every text was correctly decrypted regardless of the cipher type employed. The time required for decryption was nearly identical for each cipher type (see Table 8). Variation in the time required for decryption appears to be dependent on several properties of the files. The properties identified are: 1) author style; 2) file size, non-standard English, such as name, place names, and imaginary words; 4) the ear in which the work was written; and 5) the original language in which the work was written.

Table 7. Files, Tiles, and Authors.

File	Title	Author
linc11.txt	The Writings of Abraham Lincoln	Abraham Lincoln
wr10.txt	On War	Carl von Clauswitz
alice30.txt	Alice in Wonderland	Lewis Carroll
anne11.txt	Anne of Green Gables	Lucy Maud Montgomery
hend10.txt	Howard's End	E. M. Forser
janc10.txt	Jefferson and His Colleagues	Allen Johnson
jmlta10.txt	The Jew of Malta	Christopher Marlowe
glass18.txt	Through the Looking Glass	Lewis Carroll
will10.txt	The Wind in the Willows	Kenneth Grahame
wrld10.txt	The Way of the World	William Congreve

Table 8. Two Byte Block Decryption Results (sec).

File	S	P	PSP	Mean
linc11.txt	675	669	676	670.4
wr10.txt	14507	14442	14273	14418
alice30.txt	44617	44690	44470	44527
anne11.txt	778	770	774	774
hend10.txt	861	847	848	841
janc10.txt	1387	1381	1388	1389
jmlta10.txt	12680	12723	12616	12626
glass18.txt	7851	7664	7603	7732.4
will10.txt	765	743	744	750.4
wrld10.txt	546	550	546	548.8

Authors have distinct styles of writing, including the use of same sentence structure and lexicon in all of their works. Reusing the same patterns in structure and words results in a set of m-grams trained with those patterns. As a consequence, authors that share similar patterns of styles should decrypt in similar times and number of ciphertext characters. For example, *Alice in Wonderland* and *Through the Looking Glass*, both by Lewis Carroll, show similar decryption results.

Of all the files tested, *Alice in Wonderland* had the greatest diversity of names. It took the longest time of all text files to decrypt. Correspondingly, *Through the Looking Glass* also took longer to decrypt than other test files, due to the imaginary words and names contained in the text. *The Jew of Malta*, a work that included a large number of foreign names and locations also had problems with the low frequency m-grams that result from those words. Patterns in those words, and consequently the m-grams, are not as likely to be represented in the m-gram sets.

During the time periods covered by the corpus, English usage evolved, changed, and has been re-characterized. Word and usage patterns regularly change with popularity. Changes in the lexicon and language habits can result in literary era dependent m-gram sets, and; therefore, give rise to different decryption performance. Customizing m-gram sets for a particular era, over which the language has remained relatively static, may increase future decryption accuracy and efficiency. Sets of data derived from the same time period as the message are more likely to consist of the same patterns of word usage and frequency as the message.

5. SUMMARY AND CONCLUSION

P is a subset of *S*. *P* has the proper that under *P* the number of 1's and 0's are preserved between the plaintext and the ciphertext. This proper is illustrated by the uniform decryption time seen in the results listed in Table 1. The application of the *S* cipher does not necessarily preserve the number of 1's and 0's. The *S* block cipher, therefore, requires greater time to converge to the correct cipher key. Since P is S, it is expected that *PSP* is also *S*. Therefore, under block *PSP* the decryption time should be proportional to a *S* cipher key. Therefore, under block *PSP* the decryption time should be proportional to *S*.

We have presented several results that indicate the application of the STE method on performing a known-ciphertext attack for block *S* and *P*. These results did not rely on either the use of the knowledge of the chosen-plaintext or the known-plaintext attacks. The statistic of the language is provided within the property sets of the STE in the form of m-grams. Letter frequencies are also used as a property set. In general, any block cipher method that employs products of *P* and *S* should be deciphered in *S* time, independent of the intermediate block product combinations.

As the size of the meta-s-character increases, the number of meta(s,m) grams in a language also increases. Successful decryption using the forbidden meta(s,m) sets necessitates having enough of the language represented in the sets to find valid language patterns for most messages. Variations in language style and lexicon affect the set size and membership. On the average, smaller allowed meta(s,m) gram sets are less likely to contain all of the meta(s,m) grams found in a message. The necessary size of the sets, compared to the meta-language, has not previously been studied and is unknown.

6. ACKNOWLEDGMENT

We are grateful to F. Mitchell and S. Mitchell for their extraordinary effort in assisting us with portions of the software development for which the results of this paper are based.

7. REFERENCES

[1] E. Fogal, and Y. F. Huang, On the value of information in system identification - bounded noise case, *Automatica*, (18) 2 (1982), pp. 229-238.

[2] A. Carlson, R. B. Wells, and R. E. Hiromoto, Using set theoretic estimation to implement Shannon secrecy theory, *The Proceedings of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications IDAACS'2005*, Sofia, Bulgaria, September 5-7, 2005, pp. 435-438.

[3] C. E. Shannon, Communication theory of secrecy systems, *Bell Systems Technical Journal*, (28) (1949), pp. 656-715.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed., John Wiley and Sons Inc., New York, NY, 1996.

[5] R. B. Wells, *Applied Coding and Information Theory*, 1st ed., Prentice Hall, Upper Saddle River, NJ, 1999.

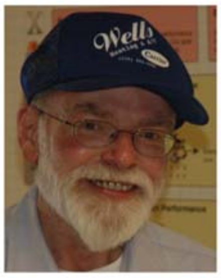
[6] S. Ross, *A First Course in Probability*, 1st ed., MacMillan Publishing, Inc., New York, NY, 1976.

[7] P. Garrett, *The Mathematics of Coding Theory*, 1st ed.. Pearson/Prentice Hall, Upper Saddle River, NJ, 2004.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed., Wiley-Interscience, Hoboken, NJ, 2001.
[9] D. R. Stinson, *Cryptography: Theory and Practice*, 3rd ed., Chapman Hall/CRC Press, London, England, 2006.
[10] P. L. Combettes, The foundations of set theoretic estimation, *Proceedings of the IEEE*, (81) 2 (1993), pp. 182-208.
[11] The Gutenberg Project, Internet, http://www.gutenberg.net
[12] Fred Schweppe, Recursive state estimation: Unknown but bounded errors and system inputs, *IEEE Transactions on Automatic Control*, (13) 1 (1968), pp. 22-28.
[13] Albert Carlson, *Set Theoretic Estimation Applied to the Information Content of Ciphers and Decryption*, PhD Thesis, University of Idaho, 2012.
[14] Horst Feistel, Cryptography and computer privacy, *Scientific American*, (228) 5 (1973), pp. 15-20.

***Albert H. Carlson** received his MS and Ph.D. degrees in Computer Science from the University of Idaho. Albert earned a BS in Computer Engineering from the University of Illinois, Urbana. With over 20 years experience as an Engineer and Engineering Manager, he has worked in safety critical circuits, consumer and industrial electronics, communications, control circuits, and embedded systems. Presently, he is an Assistant Professor at Fontbonne University, St. Louis, MO in the Department of Math and Computer Science.*

***Robert E. Hiromoto**, received his Ph.D. degree in Physics from the University of Texas at Dallas. He is professor of computer science at the University of Idaho. His areas of research include ad hoc wireless mobile networks for unmanned autonomous vehicles, and information-based design of sequential and parallel algorithms, decryption techniques using set theoretic estimation, and parallel graphics rendering systems.*



***Richard B. Wells** is Emeritus Professor and former Director of the UI Neuroscience Graduate Program, Professor of Electrical and Computer Engineering, and holds an Affiliate appointment with the Department of Physiology and Biophysics at the University of Washington School of Medicine. He also served as Associate Director of UI's MRC Institute and the Program Director for the UI's Neuroscience REU Site. He had 8 years of technical management experience at the Hewlett Packard Company prior to joining the UI in 1993. Dr. Wells is a Senior Member of IEEE.*