[A. Carlson, I. K. Dutta, B. Ghosh and M. Totaro. (Dec. 06-09, 2021). Modeling Polymorphic Ciphers, *2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC)*, Gandia, Spain, 2021, pp. 1-5, doi: https://doi.org/10.1109/FMEC54266.2021.9732571]

Modeling Polymorphic Ciphers

Albert H. Carlson
Camqed Corp.
Kyle, USA
ltzap1@gmail.com

Indira Kalyan Dutta*, Bhaskar Ghosh†
University of Louisiana at Lafayette
Louisiana, USA
*indira.dutta1@louisiana.edu,
†bhaskar.ghosh1@louisiana.edu

Michael Totaro
University of Louisiana at Lafayette
Louisiana, USA
michael.totaro@louisiana.edu

***Abstract*—Polymorphic ciphers are a relatively new construct in the field of encryption. Although the first real example of the encryption, the One Time Pad (OTP), has been known since the early part of the 20th century, the idea of a "mutating cipher" has not been implemented and well studied until now. Since these ciphers are actually "engines" comprised of single algorithm ciphers with good encryption properties, such as a large key space, evaluating the characteristics of the engine becomes important. In this paper, we present a simple method to classify the polymorphic properties of ciphers and the polymorphic engine. In addition, the security of such engines is presented, along with an analysis of the additional latency and overhead costs.**

***Index Terms*—polymorphic encryption, shannon theory, entropy, keyspace, isomorphic key reduction**

I. Introduction

A. Shannon Theory

Modern cryptography is based largely on the Information Theoretical approach introduced by Claude Shannon in the late 1940's through the early 1950's [1], [2]. Key to this study is the concept of entropy ($H(X)$) and the related measures of redundancy ($R_\lambda(X)$) and unicity distance (n) [1].

Entropy is a measure of the "surprise," or change in knowledge, that encountering the information imparts to the person analyzing the data. High entropy events indicate that not much is known about the event and can add a great deal of information, which can be gleaned from the event. The lowest entropy events have zero entropy ($H(X) = 0$), such as when it is known that an event absolutely must happen or absolutely cannot happen, can give no new information. Based on physical entropy and originally defined by Hartley [3], entropy is defined as

$$H(X) - \sum_{i=1}^{n} pr(x_i) log_2 \left(pr(x_i) \right) \tag{1}$$

and has values as shown in Figure 1.

Redundancy in a language has to do with the repetition of symbols used in the alphabet of a language. The measure of redundancy is tightly related to the entropy, as it gives a measure of how often symbols are repeated in the written record of the language. Redundancy is related to the number of symbols in the alphabet and the semantic and syntactical rules of the language. Every language has its own characteristic redundancy [4]. English, for example, has a calculated redundancy of approximately $R_{English} \approx .75$ [2]. Langendoen and

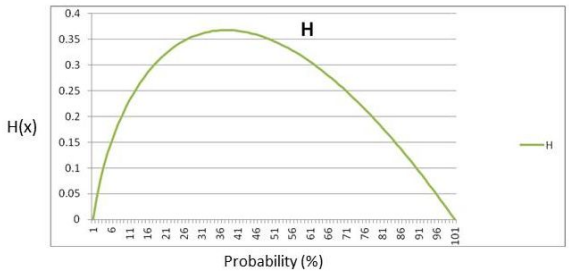

Fig. 1. Entropy Values

Postal took this concept even further [5] when they stated that each person has their own language and the conglomeration of the individuals speaking the language have a characteristic language redundancy. Redundancy is calculated using the equation

$$R_\lambda = 1 - \frac{H(x)}{H_{max}(x)} \tag{2}$$

Redundancy is important because Information Theory [1], [6] states that information accumulates at a measurable rate with respect to ciphertext. Shannon calculated this rate of accumulation and defined unicity distance as the point at which there was enough information to eliminate all spurious (false) keys from consideration during the decyrption process. This measure is defined in terms of the redundancy of the language, and is determined using Shannon's [1] equation below.

$$n = \frac{log|K_c|}{R_\lambda log|A|} \tag{3}$$

All of these measures are typically applied to the entire corpus (i.e., body of work) of a language. This is the "global" value, while smaller portions of a work or message are termed to be the "local" value of the individual metrics. Because they are a small portion of the whole, these metrics can vary greatly from the global values. Individual corpus content also varies by the habits and practice of the individual speaker. The unicity distance is a measure of security for the message, when encrypted as a cipher with a higher unicity distance is

inherently safer than one with a smaller unicity distance. That is, if s_i represents the security of a cipher, then

$$n_{c_0} > n_{c_1} \rightarrow s_{c_0} > s_{c_1} \tag{4}$$

B. Security and Keyspace

While unicity distance is one measure of security, the keyspace of the cipher can also be used as a security measure. If a cipher is strong, there are no methods to allow for removing numbers of prospective solutions with the application of a single key. Each key must be checked in order to find a solution. Applying each key, in turn, is known as the Brute Force attack (BFA) [7]. While the BFA is guaranteed to always find the solution, the time it takes to do a BFA is very large. On the average the BFA takes

$$t_{solve} = \frac{|K_c| t_a}{2} \tag{5}$$

where $|K_c|$ is the size of the keyspace for the cipher and t_a is the time it takes to apply the key, determine that the key worked or did not work, and select the next key in the test sequence [8]. Therefore, relative security can be measured in terms of keyspace, since time to solve a cipher depends directly on keyspace.

$$|K_{c_0}| > |K_{c_1}| \rightarrow s_{c_0} > s_{c_1} \tag{6}$$

C. All Ciphers are Ultimately Substitution Ciphers

Keyspace is a valid measure when the time to investigate the key is the same for the ciphers involved. Feistel showed that this is the case because all ciphers are ultimately substitution (S) ciphers [9]. Carlson demonstrated this using the concept of "isomorphic cipher reduction" [10] in which he showed that all ciphers could be replaced with an equivalent S cipher with the appropriate key. In this proof, Carlson made use of the "metacharacter assumption," in which he treated block ciphers with a block size of $|B|$ as being applied to a "metalanguage" based on the original language with an alphabet made up of characters with a size of $|B|$ letters in the original alphabet. As a result of this assumption, Carlson was able to break block ciphers using the same approaches as used for single byte S ciphers. This approach was applied to single ciphers as well as P, PS, PSP, and similar cipher compositions [10]. He then used this concept to show how "polymorphic," or sub-message constrained cipher engines could be used to defeat that attack.

II. Polymorphic Ciphers

Polymorphic ciphers [11], more commonly called "mutating" ciphers, are ciphers that are divided into smaller sections, called "shards." They are constrained in length.

Definition 2.1 (Polymorphic Cipher): A polymorphic cipher is a cipher that changes the cipher/key pair at intervals during the encryption/decryption process. More correctly a "cryptographic engine," this software uses established, peer reviewed strong ciphers for some period of time before changing. Ideally, the changes occur at frequent, short, and irregular intervals (as measured in alphabetic characters).

Polymorphic ciphers have been available in a limited scope since the advent of the Viginere cipher in 1553 [7]. Unfortunately, the Viginere cipher suffered from a limited keyspace with a single cipher. A true polymorphic cipher did not make it's appearance until the Vernum cipher (1917) [12]. This version of the polymorphic engine required changing the password for every character in the message. While it was termed the only "mathematically provably secure" cipher [1], the cost of employing the OTP is so onerous that it nearly bankrupted the Soviet Union [13] and showed that a true random number generator was required to ensure the safety of the message content. A summary of the advantages and disadvantages of polymorphic ciphers is shown in Table I.

Describing polymorphic ciphers can be done using many metrics. One of the more important is that which characterizes any cipher by how many times a cipher changes cipher/key pairs.

Definition 2.2 (Polymorphic Number): The Polymorphic number (N_P) is defined by the number of times a message or file will change cipher/key pairs during the encryption process. Polymorphic numbers may be specified in terms of an actual number or a formula that indicates the number of changes. For example, a cipher that keeps the same cipher and key pair throughout encryption has a polymorphic number of $N_P = 1$. A polymorphic cipher that changes every x blocks of characters will have a polymorphic number given by

$$N_P = \left\lfloor \frac{|M|}{|B|} \right\rfloor \tag{7}$$

where $|M|$ is the size of the message and $|B|$ is the size of the block using the same cipher/key pair. Ideally, the polymorphic number would be $N_P = |M|$, the polymorphic number of the OTP.

TABLE I
Comparison of Polymorphic Encryption Advantages and Disadvantages

Advantages	Disadvantages
Vastly larger keyspace	Higher latency and overhead
No shard reaches n_{eff}	Takes more resources
Can be done in parallel	More complex
Higher security	
Faster	
Scalable	

In addition to being composed of a single cipher per shard, polymorphic encryptions may also be comprised of product ciphers [14]. If a product cipher is used, the order of the application of the ciphers is important. In order to compare the security of traditional ciphers and polymorphic encryption engines using traditional cipher algorithms as base encryptions for the polymorphic engine via keyspace, it is necessary to present a single keyspace equation for all of these algorithms.

A. Keyspace

A single equation can be used for calculating the keyspace of any cipher, or polymorphic cipher that does not include a randomization algorithm.

Theorem 2.1: The keyspace of any cipher, without a randomization function, can be given by the following equation:

$$|K_p| = \prod_{t=1}^{s} \left(n! \prod_{i=1}^{n} \left(\sum_{j=1}^{r} \left(pr(c_j) |K_{c_j}| \right) \right) \right) \tag{8}$$

Proof: This proof will be in multiple parts. A universal key space equation must correctly return the same results as an equation targeting a specific type of cipher. In the first part of the proof, it will be shown that the equation, with the right values inserted, reduces to the correct form for ciphers with an $N_P = 1$.

For the case of a single cipher, there are exactly one shard and one way to select the order of the ciphers. Therefore, for $N_P = 1$ the value of $s = 1$, $n = 1$, and the probability of selecting that cipher is $pr(c_j) = 1$. With these values inserted into Equation 8 becomes

$$|K_p| = \prod_{t=1}^{1} \left(1! \prod_{i=1}^{1} \left(\sum_{j=1}^{1} \left(1 \times |K_{c_1}| \right) \right) \right) \tag{9}$$

which reduces to

$$|K_p| = |K_{c_1}| \tag{10}$$

as required.

Next, the case of selecting between a number of ciphers that may be selected and used for the entire message ($N_P = 1$), the values of $s = 1$ and $n = 1$ for a single shard with no reordering possible for the ciphers results in the equation

$$|K_p| = \prod_{t=1}^{1} \left(1! \prod_{i=1}^{1} \left(\sum_{j=1}^{r} \left(pr(c_j) |K_{c_j}| \right) \right) \right) \tag{11}$$

which reduces to

$$|K_p| = \sum_{j=1}^{r} \left(pr(c_j) |K_{c_j}| \right) \tag{12}$$

which is the result arrived at by Shannon in his 1949 paper on encryption [1].

In the next part of the proof we consider cascade and product ciphers. Cascade ciphers are a trivial case, since the various ciphers all use the same key [14]. The keyspace for a cascade cipher is the same size as for the single cipher and that reduction has already been addressed. Product ciphers, however, use a different key for every cipher applied. In this case, each key must be correctly identified and applied. Since each key is unique the key space is each keyspace multiplied together [10], [14]. Therefore, the keyspace is the product of each of the individual keyspaces. Again, assuming that the product cipher has an $N_P = 1$, then the value for $s = 1$. However, the order is also important. Since $N_P = 1$ there is no reordering of the ciphers and the $n!$ term, which represents the number of orders that the product cipher can take, must also be single. Therefore, $n! = 1$ while the n term in the product remains at the $n = |p|$ for the number of product ciphers used. For this calculation, it will also be assumed that there is no selection of ciphers at each step. Therefore, the equation for this case is

$$|K_p| = \prod_{t=1}^{1} \left(1 \times \prod_{i=1}^{n} \left(\sum_{j=1}^{r} \left(1 \times |K_n| \right) \right) \right) \tag{13}$$

which reduces to

$$|K_p| = \prod_{i=1}^{n} |K_n| \tag{14}$$

as required. However, if the order of the ciphers is variable, then the number of possible combinations is increased to $n!$ and that term is not 1. The formula for the keyspace becomes

$$|K_p| = n! \prod_{i=1}^{n} |K_n| \tag{15}$$

when the initial n term is not replaced by unity. Additionally, if there is a choice for each step in the product cipher, then the keyspace term returns to the probabalistic form and reduces to

$$|K_p| = n! \prod_{i=1}^{n} \left(\sum_{j=1}^{r} \left(pr(c_j) |K_{c_j}| \right) \right) \tag{16}$$

Finally, each shard is an orthogonal problem and must be solved independently. Like the keyspace for a product cipher, the keyspace for each shard must be multiplied together to arrive at the final keyspace size. For the shard keyspace ($|K_s|$), this equates to

$$|K_p| = \prod_{t=1}^{s} |K_s| \tag{17}$$

and substituting in the value for a shard keyspace found in Equation 16, the equation becomes

$$|K_p| = \prod_{t=1}^{s} \left(n! \prod_{i=1}^{n} \left(\sum_{j=1}^{r} \left(pr(c_j) |K_{c_j}| \right) \right) \right) \tag{18}$$

which is the same as claimed. A table summarizing the correct substitutions for each variable in the equation is found in Table II.

TABLE II
REQUIRED SUBSTITUTIONS FOR EACH TYPE OF CIPHER

Cipher Type	s	n	$n!$	r	$\|K\|$	$pr(c_j)$
Single Cipher	1	1	1	1	$\|K\|$	1
Choice of Ciphers	1	$\|c\|$	1	$\|c\|$	$\|K_{c_j}\|$	$pr(c_j)$
Cascade Cipher	1	1	1	1	$\|K_{min}\|$	1
Product Cipher	1	$\|c\|$	$\|c\|!$	1	$\|K_{c_j}\|$	1
Serial Cipher	1	1	1	1	$\|K_{c_j}\|$	1
Polymorphic Cipher	$\|s\|$	$\|c\|$	$\|c\|!$	$\|c\|$	$\|c\|$	$pr(c_j)$

[a] $|s|=|shards|$, $|c|=|ciphers|$

B. Complexity Analysis of Encryption Types

Since all ciphers are, at their core, S ciphers [9] it is possible to pre-calculate tables of the mappings used by the cipher [15]. There is some cost associated with the calculation and assembly of the table that is the same order of complexity as the base cipher and is repeated $|A|$ times. That cost is added to the complexity for mapping. However, the more times the encryption takes place, the smaller the effect of the original overhead.

The complexity [16] of several related types of ciphers related to polymorphic ciphers will be explained in the following three sections by the type of cipher.

1) Symmetric Ciphers: Most ciphers such as DES, Triple DES, AES and Blowfish normally operate on a fixed block size and follow the 1:1 and onto principal. Therefore it is possible to create a table for mapping each cipher from plaintext to ciphertext. Then each encryption action can be indexed out of memory. The complexity for encryption is then $O(1)$ in time and $O(n)$ in memory, for an alphabet of size n. As such the symmetric ciphers are independent of the input. We note, that we have to precalculate the n entries into the table.

The maximum memory requirement is bounded by the size of the alphabet or the number of blocks. This happens because the blocks become the alphabet in the meta language. The memory requirement can be predetermined as the size of the alphabet or the number of blocks, because the blocks become the alphabet in the meta language [10]. The constant associated with the access time is c, where c is less than or equal to the |size of the alphabet|. The constant c can be reduced by threading or parallel accesses. For example, if there are four cores, then c would be 1/|number of cores| = $1/4 = 0.25$. For each individual thread access time is still $O(1)$. But it appears overall to be faster because the actions are taken in parallel.

2) Asymmetric Ciphers: Any cipher can be mapped into a equivalent substitution cipher [1] [10] and this remains true for asymmetric ciphers. Asymmetric ciphers have more than one key. However, since one of the keys is typically published, this pair of keys is reduced to a single unknown key that must be cracked. When creating S tables for both encryption and decryption it is necessary to create two tables, one for encryption and a separate one for decryption. In terms of the required memory it's $O(2n)$, which reduces to $O(n)$. Once the cipher is mapped, the access time complexity becomes $O(1)$.

3) Randomization: If the encryption has randomization, such as modes [7], then the time complexity becomes $O(n+1)$, which reduces to $O(n)$, where n is the message size. The modes of operations can have different complexity. Since each block needs to be operated on separately at least once, thus $O(n)$ is the minimum complexity. Simple randomization doesn't add much to the complexity and most randomization involves the xor function. More complex randomization schemes will have correspondingly higher complexity. Therefore the complexity would be the complexity of the randomization and the cipher itself [17].

III. Modeling Ciphers by their Polymorphic Number

Polymorphic ciphers do not fit easily into the presently accepted models of ciphers. Because they can be composed of any type of cipher and frequently change which ciphers they use the best way to classify them is by their polymorphic number. All ciphers can be so classified because each cipher can also be modeled as a polymorphic cipher. Using the base simple cipher and then sharding the message so that the key is "randomly" changed accomplishes this change. Therefore, any cipher being discussed should also give the polymorphic number as part of the description of the cipher.

A. Deciding Factors for the Size of the Shards

With an increasing number of shards, the overhead also increases and processing efficiency decreases. Also the number of keys increase and the encryption model faces the same problem as the one-time-pad [12]. OTP is the most secure encryption, but the number of secure keys that must be properly generated, securely distributed and securely stored quickly becomes too resource intensive and expensive for any user to sustain.

Choosing the smallest unit of the information as the size of the shard results in the message having the largest number of possible shards. The security gets stronger as the size of the shards gets smaller. However the processing efficiency decreases with the decreasing shard size because of the requirement for more resources to cope with the computation of additional shards.

If the size of the message is equal to $|M|$, theoretically the number of possible shards (m) is $1 \leq m \leq |M|$. The larger number of shards results in the shards carrying less information. With the increasing number of shards, the security increases up to a certain point. According to Shannon [1], it is impossible to reliably break anything that is smaller than the unicity distance of the shard, n_l. By remaining just below n_l, the shard will remain mathematically secure. If there is not enough information at $n-1$, there is still not enough information at a shard shize of $n-2$, $n-3$, $n-4$,...1.

The shard size can still be varied to any number below the unicity distance area to make it virtually impossible to find and exploit the shard boundaries. In this case it becomes necessary to pre-calculate the unicity distance. Then the boundary becomes data-dependant and effectively pseudo-random. Shard boundaries are then a session variable and can not be $a\ priori$ determined by an attacker. The shard size acts as if it is a key in the encryption process.

With this constant security with the shard size less than n_l, reducing the shard size is unnecessary and below n_l no additional resources are required to achieve the same security. The idea is to make the job of decryption as difficult as possible for the attacker. The size of shard, $|S|$ is calculable $a\ priori$ by using Shannon's equation for unicity distance.

By calculating the global unicity distance prior to encryption it is possible to to set the shard size before encryption. Then the shard size can be varied during encryption to confuse

the attacker. This allows the users to consider security levels and extended efforts to balance these two accordingly. In this solution, the sizes of the shards were predetermined to minimize computation and retain security levels while working in a constrained environment [17].

B. The Polymorphic Continuum

The polymorphic number is actually a continuum of ciphers and combination of ciphers. At the lower end of the continuum are single ciphers that use the same password for an entire message or file. Ciphers with the same N_P are ranked from lowest to highest key space. For example, on the low end of the continuum are the single ciphers. The other end of the scale is the Vernam cipher, or One Time Pad (OTP). At the bottom of the single ciphers is the Caesar cipher, which has a key space of $|A|$. Going towards the higher end of the scale comes a single byte permutation (P) cipher, which has a key space of $2^8 = 256$ for an ASCII encoded version of English. Further towards the $N_P = 2$ mark is the substitution (S) cipher, which has a key space of $|A|!$ (26! for single letter English). This will continue to $N_P = 2$. The same progression of ciphers by polymorphic number and key space continue to the OTP (see Figure 2).

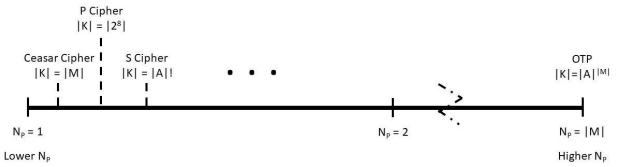

Fig. 2. The Polymorphic Encryption Continuum

This continuum gives the hierarchy of security, as well. Note that cascade ciphers will act like a simple cipher. Product ciphers can be reduced to a single S cipher [10] order by number and associated keyspace. Polymorphic ciphers using the same cipher type will be lower on the security scale than those that change cipher types. Product ciphers also act as a single cipher, since they can be reduced to a single substitution cipher [10].

IV. Conclusions and Future Work

Polymorphic ciphers are essentially an engine that combines two or more cipher/key pairs together to form what appears to be a completely different cipher system. At this time, however, there has been no method suggested that allows for the comparison between simple ciphers and between different instantiations of a polymorphic engine. This paper has proposed that the best way to compare both simple and polymorphic ciphers is to place them on a continuum that considers the number of times the engine changes cipher/key pairs (polymorphic number (N_P)) and the key space associated with such ciphers. For a given N_P the ciphers with the same polymorphic number are then arranged by their key space.

Calculating the key space for a polymorphic cipher can be done using a single equation. This equation assumes that no randomization is used as part of the cipher. An equation dealing with the standard types of ciphers was presented and it was proven to be valid for those cipher types.

An analysis of the security and the cost of using polymorphic ciphers was presented. In general, the higher the value of N_P, the better the security. Latency and overhead are greatly affected by the ability to encrypt polymorphic ciphers in parallel, while most ciphers must be treated linearly. The amount of speed up is shown, as is its dependence on the number of threads, cores, or GPUs use. Simple ciphers typically cannot be handled in this manner, making the additional security and safety from polymorphic ciphers available while speeding message handling. Latency, overhead, and additional resources do not disappear, but in terms of total time to encrypt these additional costs are hidden due to the parallel treatment of the message/file.

The need for good RNGs is also shown in the discussion of polymorphic encryption. If poor RNGs are used the system becomes susceptible to the Venona attack [13], resulting in a possible compromise for the encrypted data. However, if strong RNGs are used then the security, as measured by a large key space without clues to the key, then the security far exceeds simple ciphers.

References

[1] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656 – 715, 1949.

[2] Claude Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50 – 64, 1951.

[3] Paul Garrett. *The Mathematics of Coding Theory*. Pearson/Prentice Hall, Upper Saddle River, 2004.

[4] Edwin B. Newman and Nancy C. Waugh. The redundancy of texts in three languages. *Information and Control*, 3:141–153, 1960.

[5] D. Terence Langendoen and Paul Postal. *The Vastness of Natural Languages*. The Camelot Press, Ltd., Southampton, 1984.

[6] Thomas Cover and Joy Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc, New York, 2nd edition, 2005.

[7] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons Inc., New York, 2nd edition, 1996.

[8] Matthew Bishop. *Computer Security: Art and Science*. Addison-Wesley Professional, Boston, 2003.

[9] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15 – 20, 1973.

[10] Albert Carlson. *Set Theoretic Estimation Applied to the Information Content of Ciphers and Decryption*. PhD thesis, University of Idaho, 2012.

[11] Albert Carlson and Robert Le Blanc. Polymorphic encryption engine, 2015.

[12] Uli Maurer. A universal test for random bit generators. *Journal of Cryptography*, 5(2):89–105, 1992.

[13] John Earl Haynes and Harvey Klehr. *Venona: Decoding Soviet Espionage in the United States (Yale Nota Bene)*. Yale University Press, 1999.

[14] Uli Maurer and James Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6(1):55 – 61, 1993.

[15] Albert Carlson, Bob Le Blanc, and Carlos Gonsalez. One time pad matrix, 2016.

[16] Oded Goldreich. *Foundations of Cryptography I*. Cambridge University Press, Cambridge, 2001.

[17] Indira Kalyan Dutta. *A Novel Lightweight Polymorphic Encryption System for Data Associated with Constrained Internet of Things (IoT) Devices*. PhD thesis, University of Louisiana at Lafayette, 2021.