The Random Gate Principle

Sheagan John
Department of Mathematics
The University of the West Indies
Mona, Kingston 7, Jamaica
Email: sheagan.john@mymona.uwi.edu

Curtis Busby-Earle
Department of Computing
The University of the West Indies
Mona, Kingston 7, Jamaica
curtis.busbyearle@uwimona.edu.jm

***Abstract*---We present the main theoretical ideas behind a proposed symmetric key algorithm. We show that it can be fairly easily constructed from mathematical pseudo-random parameters and known secure cryptographic functions. We describe how time intervals can be used to establish our algorithm for encryption purposes. We will briefly discuss the decryption of messages passed through the algorithm.**

Keywords-algorithm; encryption; gate; symmetric

I. INTRODUCTION

There exists a plethora of encryption cryptosystems and algorithms of varying security. The 3DES [1] and AES [2] algorithms are well known examples of such symmetric key methods. In this paper, we give the theoretical construction for a symmetric key algorithm which we call the Random Gate Principle (RGP). A schematic overlay of the basic nature of the system is shown in Fig 1.

The Random Gate Principle is a symmetric key algorithm which relies fundamentally on the properties of wide ranging time intervals in order to ensure cryptographic security. The RGP, though unrelated to the public-key system underlying the concept of Merkle's puzzles [3], is influenced by them. While Merkle's puzzles use a large number of messages to hide a particular one, the RGP conceals this message through inserting "garbage" of arbitrary length within the original message itself. The algorithm separates a plaintext message into blocks of predetermined bit lengths and feeds each of these individual blocks through a set of logical gates, which allow passage of a given block after a randomly determined time period. In this way, during the period between the passage of successive blocks, a string of bits can be inserted, where the insertion length varies according to the time interval. As a result, the original message is hidden within a longer garbled one which is then further encrypted using two internally generated keys. The final output from the procedure has the property that no attacker can determine the length of the original message from that of the encrypted one, or from the length of time taken for encryption.

The main idea with regards to proposing the RGP is the ability to combine low complexity cryptosystems to create a secure encryption algorithm. The remainder of this paper is organized as follows. In Section II we provide an outline of the encryption procedure, in Section III some bounds related to the message length are calculated, in Section IV we give an example of some simple attacks against the algorithm, and in Section V the method of decryption is outlined.

Throughout the paper it is to be understood that the output of all deterministic functions changes with the master key used for the encryption process.

II. OUTLINE OF ENCRYPTION

A particular gate of the RGP is denoted by a_k or b_k as shown in Fig 1. The gates are mathematical constructs, essentially consisting of functions which either have a defined image for a given input or do not. These functions have a predetermined and static set of parameters which describe their restricted ranges and as such the set of valid inputs. The most important criterion is that no two gates may ever both consider the same input as valid. Each gate also contains a check function and this will be discussed in relation to A_{in} and A_{out}. The total number of gates, N, may be some predetermined value which is known by both the sender and receiver of the encrypted message. Alternatively, this value may vary accordingly with the output from some deterministic function known to both parties.

The space A_{out} contains a check function and two pseudo-random number generators (PRGs) which we shall denote as PRG_1 and PRG_2. We denote by A_{in}, the representation of a space for implementation of three main functions.

The first function separates the initial plaintext message of length L into individual blocks each of length $\frac{L}{N}$ with the ability to pad a string of zeroes to the last piece until it is of exact length $\frac{L}{N}$. Each block then undergoes a left circular shift, where the number of bit positions shifted is equal to the block's position within A_{in}. This shift is to prevent the blocks of a random

 ISBN: 978-1-61208-427-5

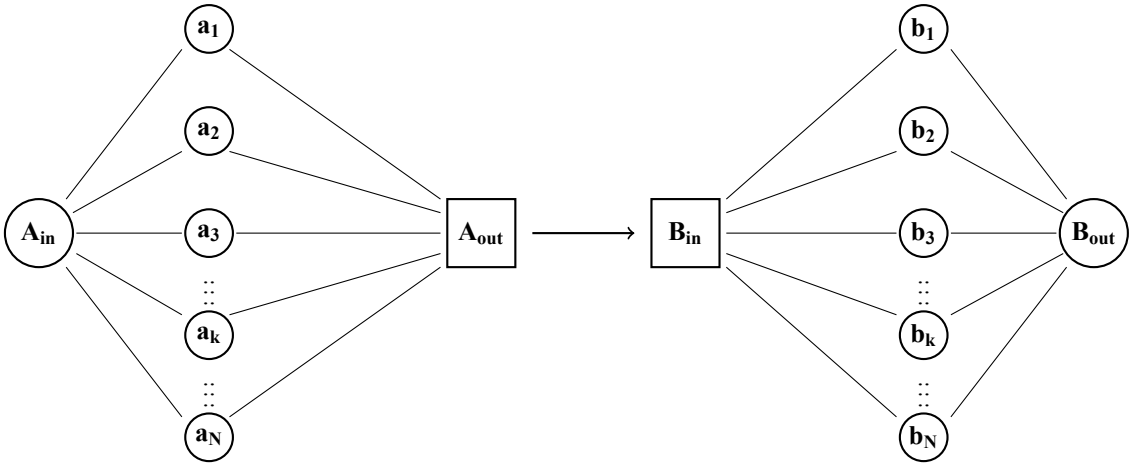

Fig. 1. A schematic presentation of the RGP gates.

message from exhibiting exceptional structure within the "garbage" surrounding them.

The first permuted $\frac{L}{N}$ block is prepended with the 16-bit representation of 1 and likewise each subsequent block till the last is prepended with that of the 16-bit representation of N. This concatenation will be denoted by $m||\frac{L}{N}$ where m is m'th integer.

The second function $H(X)$ takes each block of the form $m||\frac{L}{N}$, in order, and maps it to a gate a_k. This is not based on the parameters which govern the distinct valid inputs for a_k but rather on a conditional statement which assigns the prepended 16-bit value to a local variable within a_k if and only if a certain condition is met. In order for the choice of gate to be indistinguishable from random it is thus needed that the output of $H(X)$ also be indistinguishable from random. As an example, suppose that the Yarrow-160 [4] protocol is implemented within $H(X)$ in order to generate a random number, the last 32 bits of which are evaluated $mod(N+1)$. If the result is zero, a new number is generated and evaluated, otherwise the block $m||\frac{L}{N}$ is sent to the gate which has the number which is equal to the $H(X)$ output. In this manner the mapping $H(X) : A_{in} \longrightarrow a_k$ is independent of the message itself.

The third function is a check function which takes as input a number and outputs either 0, 1 or $\perp$(null). A gate, in order that only the blocks of the original message pass through it, does not assign a value to the local variable until the block is mapped directly to it from $H(X)$. Once the block is accepted, a single, static 256 bit number unique to each gate is relayed to the A_{in} check function. If this number matches the stored value associated with that gate, the check function outputs 0 and no new strings are fed into the $H(X)$ function. Simultaneously, a unique but static 256 bit number is sent from the gate to the check function of A_{out}, which outputs 1, thus initiating the two pseudo-random generators within A_{out}.

Each output of PRG_1 is a 32-bit number sent to and evaluated by all gates. The PRGs output cycle is attached to a counter which increases with every output and beginning with a value of 1 the counter value c increases to a value of N at which point the counter resets to 1. PRG_2 outputs a 16-bit number between 1 and N inclusive and prepends this number r to the output of the first PRG. The gate which receives this concatenated string as input and considers it valid, uses its check function to compare the r value to the value of the number m present in the $m||\frac{L}{N}$ block from $H(X)$. If the gate contains no information or the m value present is not equal to r, the check function will output 0 and upon receiving a zero value the check function within A_{out} outputs $\perp$ forcing PRG_2 to output 0 until the counter value c has reseted to 1. At this point PRG_2 will again output a valid non-zero number and prepends this to a new output from PRG_1 which the accepting gate again checks for equality to m.

If the two values are equal the check function of the gate outputs a single, unique, static 256 bit number to A_{out} which causes the check function of A_{out} to output 0 and terminate the loop for both PRGs. Simultaneously a unique 256 bit number is sent to the check function of A_{in} which outputs 1, enabling a new block to be fed into $H(X)$. As soon as the m value is checked to be equal to r the gate passes the reference value of its local variable to an array in A_{out} and is dereferenced in order for the variable to take a new value.

A. Bit Insertion

The data in A_{out} is sent through a one way route to B_{in} (see Fig 1) where the configuration of B_{in} is the same as that of A_{in}. Because of this, the complexity of implementation may be simplified by using the same

 ISBN: 978-1-61208-427-5

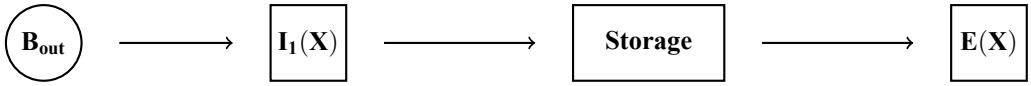

Fig. 2. The encryption process in stages.

set of functions and conditions within the gates for each pair, a_k and b_k. In fact the symmetry of the two halves can be made exact by using Yarrow-160 for both sets of two PRGs (PRG_1, PRG_2) and (PRG_3, PRG_4) present in A_{out} and B_{out}.

We denote by $I_1(X)$ the insertion function containing a one bit PRG linked to two counters. The blocks leaving B_{out} are once again in the form of $\frac{L}{N}$, having had the prepended 16-bit representation of m removed. Upon leaving, the first counter analyzes the constant data stream one bit at a time looking for an null space. Thus, upon finding the end of the first block, the counter increments from 0 to 1 while the PRG inserts a bit into each null space and does so until the counter detects the beginning of the second block. Since the PRG is only activated by the presence of an null space it will not replace any part of the original message. When each new block is detected, the counter increases and the PRG begins insertion once more. This cycle continues until the counter reaches N and resets, thus deactivating the PRG. The second counter records the number of bits $(B_1, B_2, \ldots, B_{N-1})$ inserted by $I_1(X)$ during each cycle.

The entire string is stored in a dynamic array called $Storage$ which holds each incoming bit in sequence. The insertion length values are similarly stored in separate arrays. Two functions are contained within $Storage$.

$D(X)$ takes as input the values $(B_1, B_2, \ldots, B_{N-1})$ and outputs $(C_0, \ldots, C_{N-1})$.

An insertion function, $I_2(X)$ uses a PRG- such as the Yarrow-160- to generate multiple outputs of random bits. These outputs are concatenated until they exceed the length of some value based on the $N-1$ insertion lengths. For example, a naive value may be given by

$$V_1 = \left\lceil \frac{B_1 + B_2 + \cdots + B_{N-1}}{|B_i - B_j|} \right\rceil \tag{1}$$

where the choice of i and j depends on the master key. At this point the concatenated string has all rightmost bits removed until it is of the exact length as V_1. The shortened concatenation is prepended to the garbage filled message . An similar procedure is followed for a second set of outputs of final length V_2, which is appended instead.

In the construction below, we denote the following: P, Q are large prime numbers which are known beforehand by both parties; $\alpha_i \neq j_i$ are numbers ranging from 1 to $N-1$.

$$C_0 = P(B_{j_1})^Q \tag{2}$$

$$C_1 \equiv B_{\alpha_1} (\mathrm{mod} B_{j_1}) \tag{3}$$

$$C_2 \equiv B_{\alpha_2} (\mathrm{mod} B_{j_2}) + C_1 \tag{4}$$

$$\vdots$$

$$C_{N-1} \equiv B_{\alpha_{N-1}} (\mathrm{mod} B_{j_{N-1}}) + C_{N-2} \tag{5}$$

Since each α_i, j_i is unique then each of the insertion bit values is used twice; once as a modular base and once as the number being reduced. The function $D(X)$ is deterministic, in that the sequence of pairs (α_i, j_i) is fixed regardless of the message being encrypted. This means that the value $B_{j_1} = B_k$ $(1 \leq k \leq N-1)$ for a fixed value of k. The importance of the equations given above is shown in Section V, with regards to decryption.

B. Keys and Authentication Codes

At this point the garbled message string and the set of values $C_0, C_1, \ldots, C_{N-1}$ are transfered to arrays in the space denoted by $E(X)$. This space contains an exclusive bitwise addition function and two deterministic functions, $E_1(X)$ and $E_2(X)$. The deterministic functions may be of varying mathematical complexity or may even be physically determined such as described in [6]. We do require that they be difficult to invert.

The purpose of $E_2(X)$ is to output two strings of 128 bits in length where the first output is prepended to the data and a second output is appended to the data. The first output of $E_2(X)$ may be determined by the length of the garbage filled message. The second output is determined using the value of the first output and thus by association would also be dependent on message length. The reliance on message length for generating input values for $E_2(X)$ underlines the importance of widely varying lengths of inserted bits.

Given a large range of possibilities, and since it is likely that any change in the first output will affect the second we can use these pairs as message authentication codes (MAC) [9]. See Section IV for more detail on this procedure.

We wish that an active attacker who either deletes bits from the encrypted message or inserts bits will, through tampering with message length, almost certainly invalidate one or both $E_2(X)$ outputs upon decryption. This is a useful implementation due to the simplicity of verification and given that MAC

 ISBN: 978-1-61208-427-5

authentication has been proven to be very secure [7]. To rigourously prove the claim of invalidation of $E_2(X)$ outputs would require a detailed explanation of how $E_2(X)$ works. Due to constraints on the amount of material which can be presented, we have decided to discuss this in future work.

Unfortunately we do not obtain non-repudiation [8] nor defend against an active attacker who does not change message length. This last concern can be somewhat countered by structuring $E_2(X)$ to depend not entirely on message length but on some other unique property such as a combination of message length and number of 0's contained in the message.

Two fixed length keys are outputted by $E_1(X)$ given the B_{j_1} value mentioned above as its input.

As an example, take a set of 512 bit keys, where the bitwise XOR function applies the first 512 bit key to the first 512 bits of the data. The next 512 bits of data (if more than 512 bits remains) is similarly XORed by the same key until all data is encrypted. This newly encrypted message is again XORed, but with the second key. It is after this procedure that the MACs are concatenated to the message.

III. MESSAGE LENGTH BOUNDS

In this section we must stress that the very nature of the randomness of the gate mechanism only allows for the calculation of *expected* values for most of the bounds. In particular there is no true upper bound as a gate may theoretically never be opened. In fact it is of great importance that the number of gates is small enough so as to allow a high *expectation* that every gate will open within some reasonable time span.

All the values obtained assume perfectly random behaviour. In practice there will likely be some small bias for particular gates.

Let us begin by calculating the lower bounds of time interval between exactly two consecutive blocks of form $m||\frac{L}{N}$. The denotation L is the length of the original plaintext, and $\bar{L}$ is the total length after insertions.

The lower bound can be found exactly, where $\mathbb{P}(X \to a_k)$ describes the probability of a block being sent to a given gate. Here, X represents a plaintext block. Assuming the Yarrow-160 protocol is indistinguishable from true random, then $\mathbb{P}(X \to a_k)$ will be $\frac{1}{N}$ whilst the time (t_h) to execute is constant regardless of the chosen gate. The probability $\mathbb{P}(X \to A_{out})$ is $\frac{1}{N^2}$ since the probabilities that the non-empty gate accepts the 32-bit number generated PRG_2 and the correct output from PRG_1 is prepended both have probability $\frac{1}{N}$. Assuming that both PRGs output the correct values then the time taken is just the constant $t_{g(o)}$ which is the time taken to open the gate and is a measure of the time taken for one complete cycle of the mechanism process described above. At this point the second block trails the first by a value of $t_{g(o)} + t_h$ but if the first block fails to pass through the second set of gates in one try then the second block will join it in B_{in}. Furthermore, if the second passes through immediately then the time interval remains $t_{g(o)} + t_h$. In fact this true for any two blocks for which the first block takes longer to pass through the second set of gates than the second block takes to pass through the first set of gates. It should also be noted that the time taken to send data from A_{out} to B_{in} is assumed to be negligible compared to $t_{g(o)}+t_h$ thus this value is what the lower bound is expressed as between consecutive message blocks.

The upper bound occurs when the time interval between two consecutive message blocks is greatest. This will occur when the first block passes through both sets of gates immediately whilst the second takes all tries. Using the probabilities above, we can show that the time taken for first block to pass from, A_{in} to B_{out} is equivalent to

$$T_1 = 2(t_{g(o)} + t_h) \tag{6}$$

The time taken for the second block to pass from A_{in} to A_{out} is

$$T_{2a} = (t_h + (N^3 - N)t_{g(r)} + t_{g(o)}) \tag{7}$$

where $t_{g(r)}$ represents the time taken for the gate to receive and reject r.

Since the probability $\mathbb{P}(X \to A_{out})$ is $\frac{1}{N^2}$ this means that there will be $(N^2 - 1)$ attempts where PRG_2 is forced to output 0 until the counter resets. Each counter cycle takes time $Nt_{g(r)}$ and thus for all incorrect attempts the total time is

$$(N^2 - 1) * Nt_{g(r)} \tag{8}$$

with the single correct attempt being of time $t_{g(o)}$. By the same argument the time

$$T_{2b} = (t_h + (N^3 - N)t_{g(r)} + t_{g(o)}) \tag{9}$$

which is the time taken for the second block to pass from B_{in} to B_{out} is the probabilistic upper bound. Therefore the total time interval is established as

$$2(t_h + (N^3 - N)t_{g(r)} + t_{g(o)}) - 2(t_h + t_{g(o)}) \tag{10}$$

assuming a perfectly random mechanism.

The effect of this time interval is to allow the one bit PRG within the insertion function to insert zeros and ones between two consecutive blocks. If the PRG performs an insertion every t_{ins} then the number of bits inserted between the two blocks has bounds given by equation (11).

 ISBN: 978-1-61208-427-5

$$\frac{t_{g(o)}+t_h}{t_{ins}} \leq n(Bits) \leq \frac{2[t_h+t_{g(r)}(N^3-N)+t_{g(o)}]-2(t_{g(o)}+t_h)}{t_{ins}} = \frac{2(N^3-N)t_{g(r)}}{t_{ins}} \quad (11)$$

This upper bound, however, is true only for the first two blocks after which the bound is represented by that of equation (12).

$$\frac{t_{g(o)}+t_h}{t_{ins}} \leq n(Bits) \leq \frac{(t_h+(N^3-N)t_{g(r)}+t_{g(o)})-2(t_{g(o)}+t_h)}{t_{ins}} = \frac{t_{g(r)}(N^3-N)-(t_{g(o)}+t_h)}{t_{ins}} \quad (12)$$

This wide ranging insertion length is meant to ensure that accurate prediction of the final length is extremely difficult. The probability of an inserted string being of a given possible length is $\frac{1}{N^2}$ and that of each $N-1$ insertion all having a particular length such that the sum of the lengths of all inserted strings is a given value can be approximated.

The probability of a message of original length L being mapped into that of maximum or minimum length; max($\bar{L}$), min($\bar{L}$) is

$$\frac{1}{N^2 * N^{N-1}} \quad (13)$$

since there is only one way of summing to either the maximum or minimum possible final length.

The probability $\mathbb{P}(L \rightarrow \bar{L})$ where the insertion length is non(max, min) increases with N and is largest for a total insertion length corresponding to N gate rechecks. We will determine the least number of repeats of the same message that may be processed before the output lengths are forced to match.

This can be solved exactly by considering the weak partition, $wp(N)_{N-1}$, of N into $N-1$ non-negative integers.

$$wp(N)_{N-1} = \frac{1}{(N-2)!}\prod_{i=1}^{N-2}(N+i) \quad (14)$$

$$P(L \rightarrow \bar{L}) = \frac{wp(N)_{N-1}}{N^2 * N^{N-1}} \quad (15)$$

Thus the number of repeated cycles of the same message must be no more than this absolute smallest value for mandatory correlation for any message length. The more numerous the gates, the larger this value becomes. At $N = 16$ this value is

$$\mathbb{P}(L \rightarrow \bar{L}) = \frac{wp(16)_{15}}{16^2 * 16^{15}} \cong \frac{1}{2^{40.89}} \quad (16)$$

and this represents the total insertion length most likely to be present. An equal total length, however does not indicate that the sequence of insertion lengths is the same, as the probability of a given inserted length is still $\frac{1}{N^2}$.

The last bound of importance is that of message length to number of gates. Since the security is based on random insertion length, the original message length must be at least N-bits. The upper bound of message length is determined by the security of encrypting a string of zeros, as will be shown in the next section.

IV. SIMPLE ATTACKS

We illustrate a chosen-plaintext attack. Consider an attacker who knows that the MAC lengths are 128 bits. We show the method by which an attacker can break the security in the shortest possible time with only this knowledge.

Assume the attacker sends a string of zeroes through the RGP to be encrypted. The garbled message has been XORed by two 512 bit keys, as in Fig. 3. Note that the circular shift on each block of zeroes does not affect it at all.

We let the length of $1st\,Division$, denoted by C, be arbitrarily long and thus the last $512 - V_1$ bits of the XORed keys will always be XORed with zeros. Knowing the combined length of the first MAC and B is $128 + V_1$ bits, the attacker assumes C is XORed with some $512 - V_1$ bit portion of the keys. The attacker knows that the encrypted $512 - V_1$ bits they are searching are in fact the keys themselves but does not know what V_1 is, and will not know even if the length of C is exactly 512 bits. This is because the beginning V_1 bits of the first 512 bit encrypted portion are different than the first V_1 bits of the subsequent portion and thus no repeating pattern yet emerges. As C exceeds 512 bits the sequence of the further bits exactly repeats that of the aforementioned $512 - V_1$ bits. At $C = 1024 - V_1$ bits the entire repetition is shown and the attacker can determine what the XOR value of the last $512 - V_1$ bits of the two keys is. This gained information is enough to determine which portions of the ciphertext are simply a long string of zeroes. The same method can be used if C is just a string of ones.

For complete security of any message encrypted with a key of length K it is thus recommended that the message be no longer than KN bits in length where N is the number of gates. This is easily remedied by forcing an initial message to be split into pieces and each piece fed into A_{in} in sequence, where new keys are generated each time. Another, less simple countermeasure is to construct $E_1(X)$ such that key length is dynamic and the keys may have differing lengths. Under these two improvements, chosen-

 ISBN: 978-1-61208-427-5

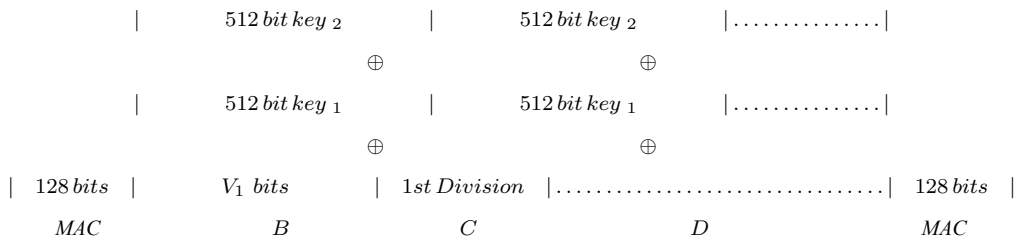

Fig. 3. Ciphertext output for an arbitrary message.

plaintext attack is not enough to break security. This is since the MAC, the value of V_1, the two keys, and the garbage inserted, constantly change with each encryption, even with the same plaintext used multiple times.

We will now allow the same attacker to record the time intervals between several different length message inputs and their encrypted outputs. We must show that such a timing attack is insufficient to determine with non-negligible probability which encrypted message the attacker is viewing. The attacker is given the power to submit an arbitrary length string of zeros or string of ones, both of which have length less than or equal to KN. The attacker then outputs the probability that the encrypted message is that of the zeros. The probability will be $\frac{1}{2} + \varepsilon$ where the value of ε must be non-negligible for the algorithm to be considered secure.

One way to accomplish this is by forcing all encrypted messages to remain in the system until a minimal time

$$2(N^3 - N)t_{g(r)} * (N - 1) * \alpha \qquad (17)$$

has elapsed, where α may be arbitrarily large. Though, theoretically, a message portion may take an infinite amount of time to pass through the gates, by choosing a large value for α the majority of message encryption times can be standardized. Such an α would depend on the number of gates and would be determined experimentally. The obvious downside to this approach is the artificially long encryption time for some messages.

A. MAC and Key Integrity

With regards to an attack on message integrity we consider an attacker attempting an existential forgery under a chosen-plaintext attack. Also the attacker wants to be able to create a message which has the same pair of MACs as a valid message.

Suppose that the attacker intercepts an encrypted message and attempts to use the same MACs on an arbitrary plaintext of his choosing. As was shown in Section III, even with only sixteen gates, the number of messages that must be encrypted to guaranteed message length collision is large. Even then, the probability that the number of 0's occurring is the same in both cases is unlikely. Given that the $E_2(X)$ outputs depend on these two factors, a breach of integrity reduces to finding a collision for $E_2(X)$. It is thus extremely important that $E_2(X)$ be sensitive to even very minor changes in its inputs as well as possessing a large range of outputs. Taking into consideration a birthday attack, the number of random messages an attacker must send before being guaranteed a collision pair for both MACs is

$$2^{64} \cdot 2^{64} = 2^{128}$$

This assumes that there is no exploitable bias in the construction of $E_2(X)$.

Similarly, trying to force decryption of an arbitrary message with an externally generated message authentication code is an undertaking by brute force.

Assuming the bounds of message length for security against pattern recognition are adhered to and the function $E_1(X)$ which generates the keys is not known, then any simple attack on the message through key integrity which does not rely on exploiting some unforeseen weakness in the $E_1(X)$ function will be a brute force attack on the key length.

V. MESSAGE RECOVERY

Consider a fully encrypted message which has been sent to a recipient along with the values, $C_0, C_1, \ldots, C_{N-1}$. The intended recipient must possess knowledge of various portions of the encryption process. Namely, the number of gates, the values of the prime pair P, Q as well as the sequence of pairs (α_i, j_i) (see Section II.A) must all be shared knowledge. It is given that the recipient's RGP decryption algorithm contains exact replicas of $E_1(X)$, $E_2(X)$, the circular shift function, and the deterministic sub-function of $I_2(X)$ which calculates the values V_1 and V_2.

Firstly, the value B_{j_1} is recovered from C_0 by dividing by P and then applying Fermat's Little Theorem using Q. Once B_{j_1} is found it is used as input for $E_1(X)$ and thus the key pair is generated. The ciphertext is XORed, initially with the second secret key, and then with the first. We will assume, as

 ISBN: 978-1-61208-427-5

proposed in Section II.B, that $E_2(X)$ takes as its input a combination of ciphertext length and number of 0's occurring. It is thus easy to determine both of these values, and consequently, to verify the validity of the MACs. If the MACs are valid then the ciphertext may now be "ungarbled" in order to extract the original data; otherwise it is discarded.

Recovering the plaintext is accomplished by using B_{j_1} and the values $C_1, \ldots, C_{N-1}$ given by equations (3) to (5) to obtain $B_1, \ldots, B_{N-1}$. For this procedure it is vital that the sequence of pairs (α_i, j_i) be known. Once each of the values $B_1, \ldots, B_{N-1}$ is obtained, it is simple to remove all inserted bits . Now, the beginning of the ungarbled message is prepended by the first output of $I_2(X)$,so we proceed by removing the V_1-bits. Likewise, the appended V_2-bits is similarly removed. Reversing the circular shift on each block, concatenating the remaining data, and removing the padded zeroes from the final block recovers the original plaintext.

VI. A SIMPLIFIED EXAMPLE

We now give an example of encryption of some sample messages using a method based on a significant simplification of the RGP algorithm. We thank Ritesh Reddy for writing the code which implements this encryption.

```
Number of Blocks: 10     Plaintext Length:
       24     Ciphertext Length: 30
   Original Plaintext: Hello My Name is
                  Ritesh!
  Encrypted:æ└ -°a0\└+¥Êür}næÛ"<<<ñ¬õËËË
    Decrypted: Hello My Name is Ritesh!

 Number of Blocks: 4     Plaintext Length:
       21     Ciphertext Length: 24
Original Plaintext: NSA NSA NSA SECRET!!!
   Encrypted: õúèÇõúcBpucBvhfuhw×××444
     Decrypted: NSA NSA NSA SECRET!!!
```

VII. CONCLUSIONS

In addition to the specific issues noted in previous sections, as an immediate concern, we aim to practically ascertain the relative performance and safety of the RGP versus some of the commonly used cryptographic methods. At the time of writing this paper, this analysis is only preliminary due to lack of a complete working model of the RGP. We also plan to investigate the robustness of the algorithm with regards to accurate decryption when the initial message is comparatively long and to determining how the ciphertext length varies with the number of gates.

References

[1] http://csrc.nist.gov/publications/nist-pubs/800-67-Rev1/SP-800-67-Rev1.pdf (Accessed on July 10, 2015)

[2] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (Accessed on July 10, 2015)

[3] R. C. Merkle, "Secure communications over insecure channels, " Communications of the ACM, Vol. 21, Issue 4, pp. 294-299, 1978.

[4] B. Schneier, J. Kelsey, N. Ferguson, "Yarrow-160: Notes on the design and analysis of the Yarrow cryptographic pseudorandom number generator, " SAC '99 Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography, pp. 13-33, 1999.

[5] G. Hanaoka, J. Shikata, Y. Zheng and H. Imai, “Unconditionally secure digital signature schemes admitting transferability, " ASIACRYPT, LNCS, vol. 1976, pp.130-142, 2000.

[6] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions, " Science, vol. 297, pp. 2026-2030, 2002. [DOI:10.1126/science.1074376]

[7] G. Hanaoka, J. Shikata, Y. Zheng and H. Imai, “Unconditionally secure digital signature schemes admitting transferability, " ASIACRYPT, LNCS, vol. 1976, pp.130-142, 2000.

[8] F. Oggier, H. Fathi, "An authentication code against pollution attacks in network coding, " IEEE/ACM Transactions on Networking, 2009. [DOI:10.1109/TNET.2011.2126592]

[9] A. S. Aiyer, L. Alvisi, R. A. Bazzi and A. Clement, "Matrix signatures: From MACs to digital signatures in distributed systems, " Distributed Computing, 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008.

 ISBN: 978-1-61208-427-5