REAMC Report-007(2023)
Issue 1, July 2023

# ACORN-QRE: Specification and Analysis of a Method of Generating Secure One-time Pads for Use in Encryption

Roy S Wikramaratna

REAMC Limited (Reservoir Engineering and Applied Mathematics Consultancy)

4 Nuthatch Close, Poole, Dorset BH17 7XR, United Kingdom

Website: https://www.reamc-limited.com

Email: rwikramaratna@gmail.com

Telephone: +44(0)7968 707062



This report has been prepared with the intention of posting both on the REAMC Limited website and on the Cryptology ePrint Archive https://eprint.iacr.org/.

REAMC® Limited

# ACORN-QRE: Specification and Analysis of a Method of Generating Secure One-time Pads for Use in Encryption

Roy S Wikramaratna

## Abstract

The Additive Congruential Random Number (ACORN) generator is straightforward to implement; it has been demonstrated in previous papers to give rise to sequences with long period which can be proven from theoretical considerations to approximate to being uniform in up to $k$ dimensions (for any given $k$).

The ACORN-QRE algorithm is a straightforward modification of ACORN which effectively avoids the linearity of the original algorithm, while preserving the uniformity of the modified sequence. It provides a new method for generating one-time pads that are resistant to attack either by current computers or by future computing developments, including quantum computers. The pads can use any alphabet (including both binary and alphanumeric) and can be used with a Vernam-type cypher to securely encrypt both files and communications.

This report explains how the ACORN-QRE algorithm works and provides evidence for the claim that the resulting one-time pads are inherently not susceptible to cryptanalysis and that they will remain secure against foreseeable developments in computing, including the potential development of quantum computers.

The ACORN-QRE algorithm is patented in the UK under Patent No. GB2591467; patent applied for in the US under Application No. 17/795632. The patents are owned by REAMC Limited, 4 Nuthatch Close, Poole, Dorset BH17 7XR, United Kingdom.

.

# 1 INTRODUCTION

The development of quantum computing threatens to render current cryptographic methods obsolete. Quantum computers will potentially be able to use quantum physics to quickly solve problems that are impractical to solve using current computing technology, including problems such as prime factorisation that are relied on to keep current encryption standards secure. This endangers both communications channels and any data encrypted and stored on online servers, also leading to concerns over harvesting and storing of encrypted data and communications against the possibility of being able to decrypt it in the future.

The Additive Congruential Random Number (ACORN) generator [1,2] is a method for generating uniformly distributed pseudo-random numbers which gives rise to sequences with extremely long period which can be proven from theoretical considerations [3,4] to approximate to uniformity in any specified number of dimensions. Extensive empirical testing, most recently and most significantly as reported in [5,6,7], has demonstrated the excellent statistical performance of the ACORN generators over a very wide range of initialisations. Despite their excellent statistical performance, ACORN generators are not cryptographically secure due to their linearity, which means that exact knowledge of only a small number of consecutive variates is sufficient to allow the entire sequence to be generated over its full period.

The ACORN-QRE algorithm (which is patented in the UK and also has a US patent applied for) is a straightforward modification of ACORN which effectively avoids the linearity of the original algorithm, while preserving the uniformity of the modified sequence. It provides a new method for generating one-time pads that are resistant to attack either by current computers or by future computing developments, including quantum computers. The pads can use any alphabet (including both binary and alphanumeric) and can be used with a Vernam-type cypher to securely encrypt both files and communications, rendering them resistant to attack both present day and for the foreseeable future. The standard implementation of ACORN-QRE is secure enough to use for high-level diplomatic communications while also providing a wide enough choice of pads (far in excess of $10^{30}$) to allow its use for encryption of mass private communications; it can equally be applied to all levels of communication in between these extremes, including military, business, commercial or security services. If desired it is straightforward to scale the algorithm to provide even greater levels of security and larger choice of pads with only a small additional overhead.

This report has been prepared with the primary purpose of explaining how the ACORN-QRE algorithm works in order to allow the cryptographic community to make a proper assessment of the veracity of our claims outlined above for the ACORN-QRE algorithm, most

importantly the claim that "*ACORN-QRE provides a new method for generating one-time pads that are resistant to attack either by current computers or by future computing developments, including quantum computers*". The report is also intended to be accessible to a more general mathematical reader who may not be a cryptographic specialist.

Section 2 of the present report discusses the use of Vernam cyphers and one-time pads for encryption of data. Section 3 provides the definition of an ACORN sequence and an overview of the key theoretical analysis and empirical test results that have been published to date. Section 4 goes on to consider the modifications that are made in the ACORN-QRE algorithm to achieve effective cryptographic security. Section 5 illustrates a number of different approaches to demonstrating cryptographic security for the ACORN-QRE algorithm. Section 6 draws together the key conclusions that result from this work.

## 2 VERNAM CYPHERS AND USE OF ONE-TIME PADS FOR ENCRYPTION

A Vernam cypher or one-time pad (OTP) is an encryption system that turns out to be invulnerable to cryptanalysis and that can therefore not be broken. A 'plaintext' (which is to be encrypted for communication to another party) is paired with a random, secret key (known as a 'one-time pad' or OTP); each character of the plaintext is combined with the corresponding character of the OTP to give the "cyphertext", which looks like another random string of characters. If (and only if) the recipient has access to a copy of the OTP then the original plaintext can be retrieved directly from the cyphertext.

Suppose we define an alphabet, consisting of $N$ characters together with a one-to-one mapping from the characters in the alphabet onto $\{0, \ldots, N\text{-}1\}$. The alphabet and the mapping can be public information and are assumed known to all parties (including a potential attacker). Examples of suitable alphabets might for example include:

- Binary $N$=2 {0, 1}
- Octal $N$=8 {0, 1, 2. …, 7}
- Hexadecimal $N$=16 {0. 1, 2, …, 15}
- Alphanumeric $N$=36 {0, 1, 2, …, 9, A, B, C, …Z}
- Alphanum++ $N$=70 {0, 1, 2, …, 9, a, b, c, …z, A, B, C, …Z, @, &, ?, !, £, $, %, #}

The last of these examples, which we have arbitrarily called Alphanum++, is intended to illustrate the possibility that additional terms (such as lower-case letters, punctuation, special characters, etc) can be included in the alphabet as required; clearly $N$=70 in this case but $N$ might take a different value if additional punctuation were included and/or if a different alphabet were selected.

The message to be transmitted (plaintext) consists of a string of terms from the selected alphabet. It is clear that every character in the plaintext must be a character from the alphabet; on the other hand, not every character from the alphabet need appear in the plaintext.

The OTP is a random string of terms from the same alphabet that is shared between the two parties but is kept secret from anybody else; the OTP needs to be at least as long as the plaintext.

The cyphertext is obtained by combining the plaintext and the relevant number of terms from the OTP using term-by-term addition (modulo $N$). The cyphertext appears as another random string of terms from the alphabet; we note that the cyphertext (in common with the OTP) will include all the characters from the alphabet and they will occur with approximately equal frequency.

The plaintext can be retrieved from the cyphertext by subtraction (modulo $N$) of the terms in the OTP – but this requires knowledge of the OTP. An attacker who has no access to the OTP will be unable to reconstruct the plaintext.

As an aside we note that if the alphabet is binary {0,1} then the encryption and decryption steps are each equivalent to the XOR ("exclusive or") operation, which means that the encryption and decryption can be carried out extremely efficiently.

The idea of encryption using a random one-time pad originated about a hundred years ago and is generally attributed to Gilbert Vernam (from the American Telegraph and Telecommunications Company, AT&T) and Joseph Mauborgne (from the US Army Signal Corps). Some thirty years later, in 1949, Claude Shannon [8] (also a researcher at AT&T) published a proof of conditions for this to be an unbreakable secure encryption method, namely that the key (or one-time pad) should be truly random and at least as large as the plaintext, that sections of the pad should never be reused in whole or in part, and its contents should be kept as a shared secret between the sender and the recipient of the message. In the same article, Shannon also proved that any unbreakable system must share essentially these same characteristics.

## 3 OVERVIEW - ACORN SEQUENCES AND ACORN GENERATORS

Let $k$ be a finite, strictly positive integer. A $k$-th order ACORN sequence is defined from an integer modulus $M$, an integer seed $Y^0{}_0$ satisfying $0 < Y^0{}_0 < M$ and an arbitrary set of $k$ integer initial values $Y^m{}_0$, $m = 1, ..., k$, each satisfying $0 \leq Y^m{}_0 < M$ by the equations

$$Y^0{}_0 = Y^0{}_{n-1} \quad n \geq 1 \tag{1}$$

$$Y^m{}_n = [Y^{m-1}{}_n + Y^m{}_{n-1}]_{\mathrm{mod}M} \quad n \geq 1, m = 1, \ldots, k \tag{2}$$

where by $[Y]_{\mathrm{mod}\,M}$ we mean the (integer) remainder on dividing $Y$ by $M$.

The $k$-th order Additive Congruential Random Number (ACORN) generator is defined by Wikramaratna [1, 2] from equations (1) and (2) together with the observation that the sequence of numbers $Y^k{}_n$ can be normalised to the unit interval by dividing by $M$

$$X^k{}_n = Y^k{}_n / M \quad n \geq 1 \tag{3}$$

The numbers $X^k{}_n$ defined by equations (1) - (3) approximate to being uniformly distributed on the unit interval in up to $k$ dimensions, provided a few simple constraints on the initial parameter values are satisfied. In short, the modulus $M$ needs to be a prime power, with powers of 2 offering the most straightforward implementation, while the seed $Y^0{}_0$ and the modulus should be chosen to be relatively prime (two numbers are said to be relatively prime if they have no prime factors in common, which means that their greatest common divisor is one). This is the approach that we have adopted in most of our previous experiments with the ACORN generator, and it appears to work very successfully.

The original implementation proposed in [1] used real arithmetic modulo one, calculating the $X^k{}_n$ directly. This implementation suffered from a number of conceptual and practical limitations (in particular, the sequences generated with any specific initialisation could not be guaranteed reproducible on different hardware or with different compilers, although the statistical properties of the sequences were unaffected). These limitations could be overcome [2] through the use of the integer implementation based on equations (1) – (3). Theoretical analysis given by Wikramaratna [2] has shown that the numbers $Y^m{}_n$ are of the form

$$Y^m{}_n = \left[\sum_{i=0}^{m} Y^i{}_0 Z^{m-i}{}_n\right]_{\mathrm{mod}M} \tag{4}$$

where for any integer values of $a$ (non-negative) and $b$ (positive) we define $Z^a{}_b$ by

$$Z^a{}_b = \frac{(a+b-1)!}{a!(b-1)!} \tag{5}$$

More extensive theoretical analysis and empirical testing of the algorithm have been described in subsequent papers, including [3] and [4].

From a theoretical viewpoint [3] the ACORN generator was shown to be a very particular special case of a multiple recursive generator; when this formulation was written in a specified matrix form, it led in turn to the discovery of certain special matrices (called centro-invertible matrices) which have some interesting and unusual properties [9]. The theoretical

analysis in [4] led to a proof that a $k$-th order ACORN generator with modulus $2^{30p}$ approximates to being $k$-distributed in a particular sense that was defined in the paper.

A recent paper [10] addressed the periodicity of ACORN sequences for any specified order, modulus and any choice of seed that is relatively prime with the modulus. As an example, every ACORN generator with order at least 8, modulus $2^{120}$ and any choice of odd seed has a period length in excess of $2^{123}$. This period is already far in excess of the maximum period that might be required in the largest conceivable computationally practicable (using current hardware available in 2023) application requiring a source of uniformly distributed pseudo-random numbers; however, we note that the ACORN algorithm extends naturally and very easily to even longer period lengths simply by increasing the modulus to a larger power of 2.

Empirical tests carried out previously by the author, making use of the Diehard statistical test suite, Marsaglia [11], have been reported in [3]. Further empirical testing was carried out in 2008 and reported by the author [4], using the Version 0.6.1 of the TestU01 package described by L'Ecuyer and Simard [12]. More recently, empirical testing has been carried out using the most current Version 1.2.3 of the TestU01 package as reported in [13], [14]; that work has now been systematically extended to ACORN generators with much wider choices of initialisations.

A key recent paper [5] proposed two conjectures on the very wide range of conditions (specifically relating to the choice of seed and initial values) under which ACORN sequences having modulus $2^{120}$ and order 8 or larger can be expected to reliably pass all the TestU01 tests. In summary the conjectures state that if we define an ACORN generator with modulus $2^{120}$ and order $k$ of 8 or larger, select an odd seed "at random", together with any specified set of initial values (which may either be selected "at random", or assigned particular chosen values) then the resulting ACORN sequence will almost certainly pass all the tests in version 1.2.3 of the TestU01 BigCrush test suite.

Results were presented in [5] to support the conjectures for a selection of ACORN generators with 1000 different randomly chosen seed values and initial values chosen either as all zero, or initial values chosen at random, for modulus $2^{120}$ and for orders 8, 9 and 10. These results have been extended for the same modulus, seed values and corresponding initial values, to cover ACORN generators with orders 11 to 15 inclusive [6]. The results are further extended in [7] to cover cases with 11 selected orders between 16 and 101 inclusive.

The report [5] was the second part of a two-part report. In the first part of that report [15], the standard TestU01 package of statistical tests for uniform distribution of sequences were applied to a range of ACORN sequences having modulus equal to $2^{120}$, with a range of different seed values and of different orders from 8 through 25 as well as a selection of larger

orders up to 101. The results presented in [15] identified some very specific seed values that should be avoided; we note that making a random choice for the seed effectively avoids these undesirable seeds.

The ACORN algorithm provides an enormous number of very long sequences that each approximate to uniform distribution in $k$ dimensions, for any specified $k$. However, it should be noted that ACORN sequences are not cryptographically secure, in the sense that a complete knowledge of any $k$+1 consecutive terms in a $k$-th order ACORN sequence is sufficient to completely determine all the terms in the sequence. This is due to the linearity of the algorithm, and is clear from the mathematical analysis that is included in [3] and [4], showing that an ACORN generator is a special case of a multiplicative recursive generator.

# 4 ACORN-QRE ALGORITHM

The ACORN Quantum Resistant Encryption algorithm (ACORN-QRE) represents a straightforward modification of the ACORN algorithm which results in sequences that retain the desirable distribution properties of ACORN at the same time as rendering the resulting sequences cryptographically secure, and hence suitable for use as one-time pads in secure encryption. As will be discussed and demonstrated in section 5, we believe that the algorithm remains secure against future developments in conventional computer hardware, as well as possible against possible future developments in quantum computing.

The ACORN-QRE algorithm is patented in the UK under Patent No. GB2591467; patent applied for in the US under Application No. 17/795632. The patents are owned by REAMC Limited; contact details are shown on the title page of this report or on the REAMC Limited website, https://www.reamc-limited.com.

## 4.1 ACORN-QRE Specification

Given a suitably initialised ACORN sequence of order $k$ and modulus $2^M$, together with an alphabet of size $a$ and an integer value $b<<M$ chosen such that $a<2^b$, the ACORN-QRE algorithm can be defined in three steps, as follows:

Step 1. Calculate next ACORN variate, equal to $A$; we note that the variate $A$ has $M$ bits, and can take values from the set $\{0, 1, 2, \ldots, 2^M-1\}$

Step 2. Calculate reduced ACORN variate, $B$, by taking the leading $b$ bits of $A$. The reduced ACORN variate can take values from the set $\{0, 1, 2, \ldots, 2^b-1\}$

Step 3. Define a one-to-one mapping from $\{0, 1, 2, \ldots, 2^b-1\}$ to $\{0, 1, \ldots, a-1, a, \ldots 2^b-1\}$.

(a) If the reduced ACORN variate maps onto a term from the set {a, …, $2^b$-1} then discard the term and go back to repeat Step 1
(b) Otherwise, the ACORN variate maps onto a term from the set {0, 1, …, $a$-1}; take this for the next term in the OTP, and go back to Step 1

In this report we will assume that the algorithm is set up to generate a binary one-time pad for use in encrypting binary files; hence $a$=2 and $b$=2, the reduced ACORN variates take values from {0, 1, 2, 3} and the alphabet is {0,1}. However, with suitable modification the analysis can be extended to any desired alphabet with an arbitrary number $a$ of characters (including alphanumeric, binary, octal, hexadecimal, etc) together with an appropriate value of $b$.

The parameters used for ACORN-QRE are assumed in this discussion to be as follows (we note that this information can be shared freely without compromising the cryptographic security of the method; on the other hand, if all or part of this information were also kept secret it would make the encryption even more difficult to break). The order $k$ of the underlying ACORN generator is at least 8 and $M$ (which is the logarithm base 2 of the modulus) is assumed here to be 120 (although in Section 5 we will also consider the impact of choosing a smaller value, $M$=60). The first part of the reduction (Step 2) is assumed to use base 4, leading to a sequence of period $2^{123}$ comprising elements from the set {0, 1, 2, 3}. For simplicity, the second part of the reduction retains just those terms drawn from the set {0,1}, with all terms from the set {2, 3} being dropped. We can write this as a mapping from the set {0,1,2,3} to {0,1, x, x} where the x indicates a term that is dropped. We could equally have chosen to use a different mapping to reduce the sequence from base 4 to a binary sequence, for example dropping terms from {0,1} and mapping 2 to 0 and 3 to 1, which we could write as a mapping from the set {0, 1, 2, 3} to {x, x, 0, 1}. The period length for the fully reduced ACORN-QRE sequence will be approximately $2^{122}$ (since after the first reduction roughly one quarter of the terms will take each value and two of the four values are dropped in the second part of the reduction). It is currently unclear whether it might be possible to prove an exact value for this period; however, the important point is the order of magnitude of the period length rather than its exact value. For conciseness and clarity in what follows, we will make use of the following terminology:

(i) the original sequence (base 120) is the “ACORN sequence”;
(ii) the sequence (base 4) obtained after the first part of the reduction is the “reduced sequence”;
(iii) the final sequence (base 2) obtained after the second reduction step is the “fully reduced sequence” or the “One-Time Pad” (or “OTP”).

# 5 CRYPTOGRAPHIC SECURITY OF ACORN-QRE

Shannon's 1949 result [8] that any unbreakable cryptographic system must share some specific characteristics, as noted at the end of section 2, might appear to suggest that methods using a numerical algorithm to generate a one-time pad can never be considered completely secure. However, as will be seen here, the ACORN-QRE algorithm can be considered "effectively secure", in the sense that even though the key (comprising the seed, and initial values) is shorter than the length of the message it can be expanded using ACORN-QRE to give a pseudo-random OTP where the solution space is large enough that the resulting cryptographic system can be proved to be unbreakable in any realistically useful timeframe.

We will outline heuristic arguments to support a number of key assertions concerning the cryptographic security of ACORN-QRE. We believe that mathematically rigorous proofs are possible, but some further work is required to demonstrate this; equally importantly, we note that it has not proved possible to date to disprove them (for example, through an appropriate counter-example) despite significant efforts on the author's part. The format in section 5.1 will be to state each assertion and then to outline the supporting arguments. Taken singly or in combination these all lend support to the view that ACORN-QRE is cryptographically secure, and will remain so against current and future developments of conventional computers, as well as potential future developments of quantum computers.

For the purpose of this analysis, we will assume that the order and modulus of the underlying ACORN generator are known and that the seed is relatively prime with the modulus (so that the period length for ACORN is also known). In what follows we will use order $k$=8 and modulus $2^M$ where $M$=120, hence the seed is required to be odd and the resulting period length is $2^{123}$; as discussed in section 3, this choice of parameters has been shown to give excellent randomness properties in up to $k$ dimensions. Increasing either the modulus or the order will lead to further improvement in the randomness of the resulting ACORN sequences.

The choice of a smaller value of $M$=60 is also possible without seriously compromising the security of the method; this can be seen by repeating the analysis with appropriate minor adjustments (for example, on choosing $M$=$2^{60}$ the key length is reduced to 539, the ACORN period becomes $2^{63}$, the ACORN-QRE period is now approximately $2^{62}$, the time required for a brute force search of the solution space is much reduced but is still many times greater than the remaining lifetime of the universe, etc). In this case, the arguments in support of some of the assertions still apply, although they may not be quite as overwhelming as for $M$=120, and may need to be considered a bit more carefully; where appropriate, the potential need for such modification is addressed in the discussion.

We will assume that the ACORN generator has been initialised randomly (in the sense that each of the first ($M$-1) bits of the seed is independently initialised to 0 or 1 with probability 0.5, while bit $M$ is set to 1 to ensure that the seed takes an odd value; further, each of the $M$ bits of each initial value is also independently initialised to 0 or 1 with probability 0.5). The 'key', which comprises the seed together with the $k$ initial values (so that in the particular case where $k$=8 and $M$=120, the key length is 1079 bits) is assumed to be shared securely between the two participants, and is not known to a possible attacker.

The parameters used in the ACORN-QRE algorithm for the reduction (step 2) and the final reduction (step 3) to give the final fully reduced OTP need to be shared between the participants prior to starting the encryption process. Coupled with this, the participants need to agree and share details of the alphabet that will be used in the communications as well as the mapping (which is one-to-one and onto) that is to be used between the terms in the OTP and those in the alphabet. It does not matter at all if this information concerning parameters and mappings becomes public knowledge and finds its way into the hands of an attacker.

A final piece of information that must be shared between the participants at the time of transmitting a message is the starting point in the OTP that is used for encryption of that particular message. Care needs to be taken that individual messages are encrypted using distinct parts of the sequence and that there is no reuse of any part of any particular pad. Strictly speaking, this last requirement means that there should be two different OTPs used, for messages in the two directions; then each individual can keep track of exactly what parts of their pad have been used, and there is no risk of ambiguity about who should use which part of the pad. From a theoretical viewpoint it is best to keep the starting point in the OTP that was used for encrypting a particular message as a secret; however, in practice it does not greatly matter if the starting point in the pad used for a particular message becomes public knowledge and finds its way into the hands of an attacker, since it is of no benefit without a full knowledge of the contents of the pad.

## 5.1 Examples

### 5.1.1 Some Examples of OTP Generated Using ACORN-QRE

We include two examples of binary OTPs that have been generated using the ACORN-QRE algorithm. The main reason for including these example OTPs is for the benefit of cryptographers who might have an interest in looking in more detail at some examples of output data sets, as a starting point to help verify the correctness of our claims about the ACORN-QRE algorithm.

For the first example (see Table 4), we have chosen $M$=60 and $k$=8, while for the second example (see Table 5) we have chosen $M$=120 and $k$=8. The seeds and initial values used in generating these OTP have been initialised randomly (in the sense that was discussed above). Each table includes the first one thousand elements, ordered by rows and arranged in 25 rows of 40 elements per row; rows and columns have been numbered to assist in reading the entries. The pads can be extended and/or provided in more easily machine-readable formats if required. We have in fact already generated and saved one million elements from each pad, so each table contains only the first 0.1% of the full data set that was generated in each case; the cpu time taken to generate each pad (comprising one million ACORN-QRE variates) was around 0.25 seconds on a standard laptop computer. The initialisations that were used are not included here, but we would be happy to share further details as appropriate.

Please contact the author to discuss further; full contact details are included on the title page of this report.

### 5.1.1 Some Examples of Encryption Using ACORN-QRE

We include some simple examples of encryption and decryption, see Figure 1, making use of a binary OTP that has been generated using the ACORN-QRE algorithm. The main reason for including these examples is for the benefit of mathematical readers who may have little prior knowledge or experience of cryptography, allowing them to get a feel for the way that encryption using an OTP works in practice.

Figure 1 has six parts, labelled (a) through (f). Each part takes the form of a 20 by 20 grid containing 400 binary values; cells taking the value 1 are coloured black, while those taking the value zero are coloured white.

(a) This is a geometric design which represents the original plaintext. It is divided into four quadrants, each containing a different design.
(b) This represents a section of an OTP generated using the ACORN-QRE algorithm; it consists of the first 400 entries from the example OTP that is included as Table 5 in this report.
(c) This is the cyphertext that results from combining the data from (a) and (b) using a bitwise XOR operation.
(d) This is the plaintext, which has been retrieved from the cyphertext using the correct section of the OTP; it results from combining the data from (c) and (b) using a bitwise XOR operation. It is clear that parts (a) and (d) are identical, confirming that the cyphertext has in fact been correctly decrypted.

(e) This represents a different section taken from the same OTP that was used for the encryption; it consists of the entries 401 to 800 from the example OTP that is included as Table 5 in this report.

(f) This is the result of a decryption attempt using an incorrect section of the OTP. It results from combining the data from (c) and (e) using a bitwise XOR operation. It is clear from a comparison of (c) and (f) that the result is another random binary array, and that it provides no useful information about the original plaintext.

## 5.2 Some Assertions Concerning ACORN-QRE and the Resulting OTP

In what follows we make a number of assertions concerning ACORN-QRE and the resulting OTP (assuming the parameter choices are as described above). We will include a number of heuristic arguments in support of each of the assertions, although we will not attempt to present formal mathematical proofs at this time. Taken together, we believe that the assertions provide a convincing justification for the effective cryptographic security of the ACORN-QRE algorithm with the chosen parameters. We note that any increase in either the order $k$ or the modulus $2^M$ used for the underlying ACORN sequence will result in an improvement in the randomness of the ACORN sequence, a further increase in period length of both the underlying ACORN sequence and the resulting OTP, and also a further increase in the cryptographic security of the resulting OTP.

**Assertion 1**. Given a section of OTP (however long or short) it is not possible to determine the initialisation that was used to generate it other than by a brute-force search of all possible initialisations and the resulting pads.

**Discussion**

Suppose we know the first $p$ terms from the OTP, where $p$ is an integer greater than or equal to $k$+1 (and where $k$ is the order of the underlying ACORN generator). We know that the OTP is binary, and has been derived from a reduced sequence (base 4) by deleting all terms with value 2 or 3 and retaining the terms with value 0 or 1. The reduced sequence in its turn was derived from the original ACORN sequence by truncating each ACORN variate from 120 bits to retain only the two leading bits.

Thus, the ACORN-QRE algorithm involves three steps:

1. Going from a known ACORN initialisation to an ACORN sequence; for a $k$-th order ACORN sequence it is simple to invert the process and derive the ACORN initialisation from any $k$ consecutive terms in the ACORN sequence.

2. Going from an ACORN sequence to a corresponding reduced sequence, is again straightforward; however, the inverse process is not. It may be possible to formulate the inverse problem as a set of simultaneous linear inequalities modulo 4; however, it is unclear whether this can be done and, if so, how many such inequalities would be needed in order to derive a unique solution. The computational requirements for finding any solution would be both immense and essentially unquantifiable.
3. Going from a reduced sequence to an OTP is again straightforward, involving deletion of approximately half the terms in the reduced sequence and mapping the remaining terms onto a binary sequence; however, this step cannot be formulated such that it is mathematically invertible (since there is no way of deducing from the terms remaining in the OTP either which terms in the reduced sequence were deleted or which of the possible values they took).

It is clear that step 3 is inherently one-way and not invertible - inclusion of this third step renders the inverse problem mathematically unsolvable, except possibly through a search of all the forward solutions. Thus, the only possible way to deduce the ACORN initialisation from the OTP is by brute force search of all initialisations (and this fact remains unchanged by the advent of quantum computers). We note that if $p$ is less than (or even equal to) the full period of the OTP, then there may still be more than one candidate solution – so it would still be necessary to go on and search all initialisations to make certain that all potential candidate solutions have been found.

**Assertion 2**. The time required to do a brute force search of all possible initialisations is so long as to make this approach totally impractical (even after allowing for potential improvements in the speed of conventional computing as well as massive parallelisation and potential future developments in quantum computing).

**Discussion**

The number of different initialisations can be calculated as follows, given the order $k$=8, and $M$=120 (modulus $2^{120}$) together with the constraint that the seed should take an odd value. Thus, there are $2^{119}$ values to choose from for the seed, and $2^{120}$ values to choose from for each of the eight initial values. The modulus and initial values can all be selected independently; hence the number of different initialisations is $2^{119}\times(2^{120})^8 = 2^{1079}$. This number is so large as to be almost impossible to comprehend, but some comparisons may prove helpful.

Table 1 shows the number of seconds in various time periods, from one second through 13.6 billion years (an estimate of the time since the big bang) to 225 trillion years (which is an estimate of the time remaining to the end of the universe).

Table 2 shows the number of micro-seconds (μs) in the same time periods; each call to ACORN-QRE takes somewhere between 0.1 and one micro-second on a typical processor on a standard laptop or desktop computer so the figures in Table 2 provide an order of magnitude approximation to the number of ACORN-QRE calls that might be made on a single processor in each time period.

Table 3 gives upper bound estimates on the number of ACORN-QRE calls that might be made in the same time periods, after allowing a generous factor $2^{30}$ (approximately one billion) potential speedup which might be argued as the result of massive parallelisation, faster processors and future quantum computing developments. From the final line of the table, the maximum number of ACORN-QRE calls that could be made before the end of the universe might be around $2^{122}$; even if we needed to make just a single ACORN-QRE call for each initialisation this is a negligibly small fraction of the $2^{1079}$ different initialisations that would need to be tested.

Reducing the modulus to $2^{60}$ does make a huge difference to the number of initialisations that need to be tested (reduced to $2^{539}$) and a small reduction (by a factor of 2) in the time per ACORN-QRE call. In spite of these changes, the number of initialisations that might be tested before the end of the universe remains a negligibly small fraction of the $2^{539}$ different initialisations that would need to be tested in this case.

**Assertion 3**. Given a relatively short cyphertext of length $p$ bits (where $p \leq r$ and where, for the case of order 8 and modulus $2^{120}$, $r$=1079 bits) which has been encrypted using ACORN-QRE, there are $2^p$ possible plaintext messages, each of which would imply a particular section of OTP. Given a longer cyphertext of length $q$ bits (where $q > r$) which has been encrypted using ACORN-QRE, there are $2^q$ "candidate" messages and $2^r$ different possible OTP, each of which would imply a possible corresponding message. Irrespective of the length of the cyphertext we assert that there is no practical way of identifying any of these "possible" messages as being more or less likely to be the true message than any other.

**Discussion**

The only possible exception might be by showing that there is no initialisation that would lead to certain particular sections of OTP (in which case the corresponding

candidate messages would be rejected). Note that by Assertion 2 we know that the time required to do this is impossibly long.

If we are able to find an initialisation leading to a particular section of OTP then either it is the only such initialisation, or it is one of two or more such initialisations. In the first case we would expect to test (on average) half the possible initialisations before finding the required solution; by Assertion 2, the time to do this would again be impossibly long. Further, we would still need to go on and test any remaining untested initialisations to confirm that there is indeed only one such solution. In the second case a possible approach might be to rank the candidate messages according to the number of initialisations that lead to each particular message; unfortunately, this would once again require all possible initialisations to be tested, and by Assertion 2 this would again take an impossibly long time.

In addition, we note that for a longer message (having length $q>r$) the number of candidate messages is $2^q$; in this case we know that at least $(2^q-2^r)$ of the candidate messages will be impossible due to there being an insufficient number of different initialisations. However, the time required to classify even one single candidate message as "possible" or "impossible" would already take an impossibly long time (by Assertion 2).

We note that in the case where $M$=60 (and with order 8, as before) the above discussion still holds, but with the modification that the value of $r$ is reduced to 539 in both the statement of the Assertion and in the ensuing discussion.

**Assertion 4**. The time required to encrypt or decrypt a Megabyte of data (equal to $2^{23}$ bits, since one byte is equal to 8 bits) using the ACORN-QRE algorithm with any specified initialisation on a current (2023) standard processor is estimated to be ~4 seconds, as follows.

**Discussion**

We assume the encryption or decryption time is dominated by the generation of the OTP. We can generate more than $2^{20}$ (~$10^6$) ACORN-QRE variates per second on a standard laptop or desktop processor (taking the data in Table 4 and Table 5 as examples, each case took about 0.25 seconds to generate one million binary ACORN-QRE variates). Hence, we can estimate approximately 4 seconds per Megabyte of encryption or decryption (this converts to about 1 hour per Gigabyte).

In the case where $M$=60 there is a potential speedup factor of 2 in encryption or decryption time compared with these estimates, although we did not take any benefit from this in generating the data for Table 4.

**Assertion 5**. The period length of an ACORN-QRE sequence is so long that there is no reason to be concerned about ever exhausting an OTP generated using this method.

**Discussion**

The period length for each OTP can be shown to be approximately $2^{122}$, based on the specified parameter values $k$=8, modulus $2^{120}$, together with an odd seed value (the period is approximately half of the period of the underlying ACORN sequence, which is known to be $2^{123}$). Given sufficient time, the resulting OTP could potentially be used to encrypt up to $2^{119}$ bytes (or $2^{79}$ Terabytes) of data. *QED*.

For the case where the modulus is reduced to $2^{60}$, an analogous argument leads to the conclusion that the resulting OTP could still encrypt up to $2^{59}$ bytes ($2^{19}$ Terabytes) of data. Although this is much reduced, it is still in excess of half a million Terabytes.

**Assertion 6**. The number of ACORN-QRE sequences with order $k$=8 and modulus $2^{120}$ that are available to choose from is (vastly) more than enough for a different pad to be selected for communication between every pair of individuals in the world without any duplication.

**Discussion**

A 2023 estimate of the world population is 8 billion or $2^{33}$; hence, there are ~$2^{66}$ different ways to select a pair of individuals; thus, there are over a quadrillion ($10^{15}$ or ~$2^{50}$) times more ACORN-QRE pads to choose from than would be required for this purpose, and choosing the keys randomly would effectively ensure no likelihood of duplication.

# 6 CONCLUSIONS

The ACORN algorithm is a method for generating sequences of uniformly distributed pseudo-random numbers on the unit interval. The resulting sequences have long period and good uniformity which has been demonstrated both using mathematical analysis and by empirical testing. The ACORN sequences are not themselves suitable for cryptographic purposes because an exact knowledge of just ($k$+1) successive terms in a $k$-th order ACORN sequence allows the entire sequence to be reproduced.

The ACORN-QRE algorithm is a straightforward modification of the ACORN algorithm which allows the generation of secure one-time pads (OTP) for use in cryptography. The OTP can be defined using any desired alphabet (including binary, octal, hexadecimal, or any chosen set of alphanumeric and/or special characters). The security of the OTP (against current hardware and future developments in conventional computers, as well as possible future developments in quantum computers) has been convincingly demonstrated using heuristic arguments. We believe that if required these arguments can be refined to make them mathematically more rigorous. Significant efforts over an extended time have failed to refute the heuristic arguments (for example, failing to find any counter-examples).

Based on our analysis, ACORN-QRE provides a family of secure encryption algorithms that will remain secure against possible future developments in both conventional and quantum computers. As a result, files encrypted using this method can be safely transmitted over secure or insecure communications channels or left for collection on publicly accessible web-sites without fear of their contents being deciphered and read by any attacker, including both commercial rivals, bad actors and members of security services from any country, either now or at any time in the future. Only somebody in possession of the appropriate key will be able to decrypt the contents of the file and read the message contained in it.

ACORN-QRE provides what is effectively the same level of security as provided by a randomly generated OTP, while avoiding some crucial drawbacks/limitations

(i) A randomly generated OTP still needs to be tested to ensure the randomness of the pad. Pads generated by ACORN-QRE do not need to be tested individually because the algorithm can be proved to give the required randomness properties for all initialisations provided certain straightforward rules are followed.

(ii) In order to use a randomly generated OTP the entire pad (which must be at least as large as the message) has to be shared securely with the intended recipient of the message. With ACORN-QRE it is only necessary to share the key securely with the recipient; the key length (at least 500 and typically up to a few thousand

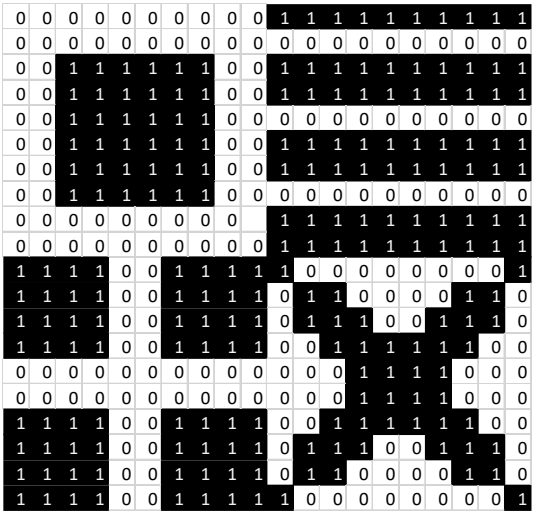
bits) is much shorter than the length of the resulting pad (at least $2^{60}$ and typically of the order $2^{120}$ bits), and is therefore easier to communicate securely.

(iii) To maintain the security of the encryption, no section of a pad should ever be reused (since re-use of the same section of a pad for multiple messages leads to a potential vulnerability to attack). The length of an ACORN-QRE pad (at least $2^{60}$ and typically of the order $2^{120}$ bits) means that it will never be exhausted in practice, and it can continue to be used as long as the key remains secure.

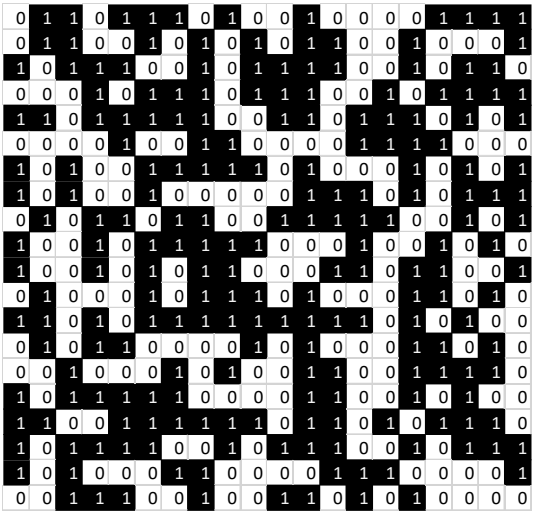
(iv) The number of different pads that are available to choose from using ACORN-QRE with order $k$=8 and modulus $2^{120}$ is so huge as to be more than enough to allow a different pad to be selected for communication between every pair of individuals in the world without any duplication.

(v) Should it ever turn out that the pad length, the number of different pads to choose from, or the randomness of the pads is insufficient for some future requirement, the ACORN-QRE algorithm scales in a natural way: increasing the modulus to $2^{240}$ (i.e. squaring the modulus) increases the time needed to generate each variate by a factor of just two, while at the same time increasing the size of the solution space (and the time for a brute force search) by a factor $2^{120}$ and also further improving the randomness of the resulting OTPs.

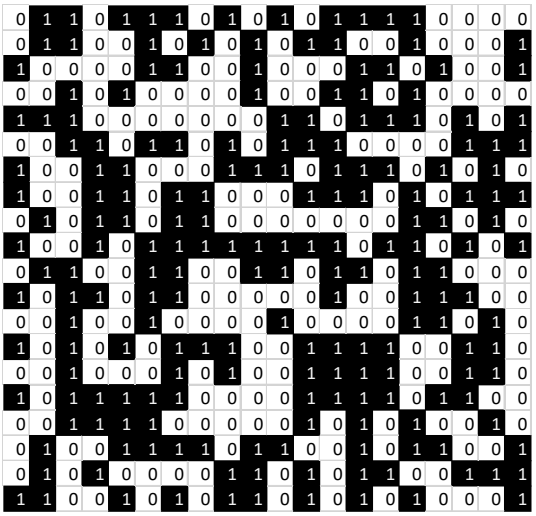
 



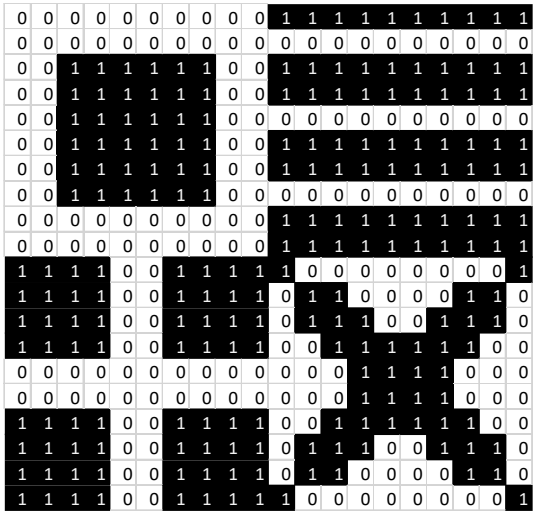
(a) Plaintext of length 400, corresponding to a 20 by 20 "picture"



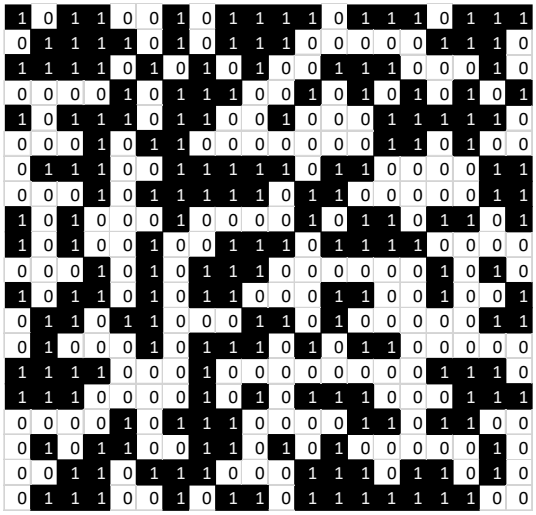
(b) First 400 entries from the OTP in Table 5 (entries 1-400)



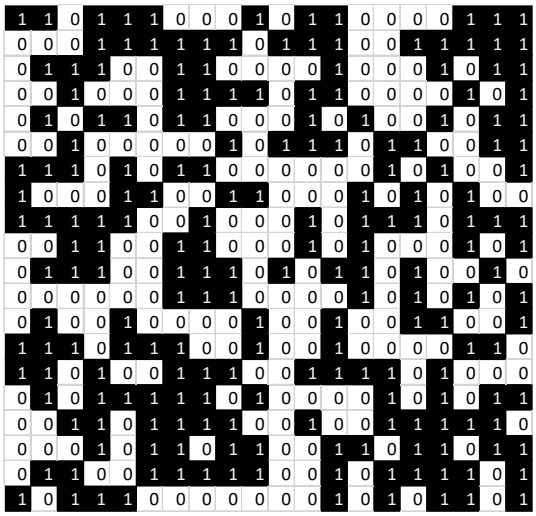
(c) Cyphertext resulting from bitwise XOR on data shown in (a) and (b)



(d) Plaintext, retrieved from cyphertext by bitwise XOR on data shown in (c) and (b)



(e) A different selection of entries from the OTP in Table 5 (entries 401-800)



(f) Incorrect decryption attempt using wrong section of OTP, by bitwise XOR on data shown in (c) and (e)

**Figure 1 Examples of encryption and decryption attempts using an OTP generated with ACORN-QRE**

| Time Period | Number of seconds, $N_s$ | $\log_{10}(N_s)$ | $\log_2(N_s)$ |
|---|---|---|---|
| second | 1 | 0.00 | 0.00 |
| minute | 60 | 1.78 | 5.91 |
| hour | 3600 | 3.56 | 11.81 |
| day | 86400 | 4.94 | 16.40 |
| year | 31557600 | 7.50 | 24.91 |
| millenium | 3.16E+10 | 10.50 | 34.88 |
| million years | 3.16E+13 | 13.50 | 44.84 |
| billion years | 3.16E+16 | 16.50 | 54.81 |
| big bang ~13.8 billion years ago | 4.35E+17 | 17.64 | 58.60 |
| trillion years | 3.16E+19 | 19.50 | 64.77 |
| end of universe ~225 trillion years from now | 7.10E+21 | 21.85 | 72.59 |

**Table 1 Number of seconds in different time periods**

| Time Period | Number of ms, $N_m$ | $\log_{10}(N_m)$ | $\log_2(N_m)$ |
|---|---|---|---|
| second | 1000000 | 6.00 | 19.93 |
| minute | 60000000 | 7.78 | 25.84 |
| hour | 3600000000 | 9.56 | 31.75 |
| day | 8.64E+10 | 10.94 | 36.33 |
| year | 3.16E+13 | 13.50 | 44.84 |
| millenium | 3.16E+16 | 16.50 | 54.81 |
| million years | 3.16E+19 | 19.50 | 64.77 |
| billion years | 3.16E+22 | 22.50 | 74.74 |
| big bang ~13.8 billion years ago | 4.35E+23 | 23.64 | 78.53 |
| trillion years | 3.16E+25 | 25.50 | 84.71 |
| end of universe ~225 trillion years from now | 7.10E+27 | 27.85 | 92.52 |

**Table 2 Number of micro-seconds(μs) in different time periods (1 second = $10^6$ μs)**

| Time Period | Number of calls ($N_c$) | $\log_{10}(N_c)$ | $\log_2(N_c)$ |
|---|---|---|---|
| second | 1.07E+15 | 15.03 | 49.93 |
| minute | 6.44E+16 | 16.81 | 55.84 |
| hour | 3.87E+18 | 18.59 | 61.75 |
| day | 9.28E+19 | 19.97 | 66.33 |
| year | 3.39E+22 | 22.53 | 74.84 |
| millenium | 3.39E+25 | 25.53 | 84.81 |
| million years | 3.39E+28 | 28.53 | 94.77 |
| billion years | 3.39E+31 | 31.53 | 104.74 |
| big bang ~13.8 billion years ago | 4.68E+32 | 32.67 | 108.53 |
| trillion years | 3.39E+34 | 34.53 | 114.71 |
| end of universe ~225 trillion years from now | 7.62E+36 | 36.88 | 122.52 |

**Table 3 Potential number of ACORN calls in different time periods; based on assumption of $10^6$ ACORN calls per second on single standard laptop processor and allowing for potential future speedup of order of $2^{30}$ (approximately one billion) which might be attributed to possible algorithmic refinements, faster processors and massive parallelisation.**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| r 2 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| r 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| r 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| r 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| r 6 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| r 7 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| r 8 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| r 9 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| r 10 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| r 11 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| r 12 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| r 13 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| r 14 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| r 15 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| r 16 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| r 17 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| r 18 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| r 19 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| r 20 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| r 21 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| r 22 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| r 23 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| r 24 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| r 25 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

**Table 4 Example of a binary OTP generated using ACORN-QRE with $M$=60 and $k$=8. The pad has 25 rows of 40 elements per row; it is ordered by rows, with row 1 containing the first 40 elements, row 2 the next 40 elements, etc.**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| r 2 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| r 3 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| r 4 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| r 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| r 6 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| r 7 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| r 8 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| r 9 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| r 10 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| r 11 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| r 12 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| r 13 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| r 14 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| r 15 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| r 16 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| r 17 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| r 18 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| r 19 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| r 20 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| r 21 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| r 22 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| r 23 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| r 24 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| r 25 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

**Table 5 Example of a binary OTP generated using ACORN-QRE with $M$=120 and $k$=8. The pad has 25 rows of 40 elements per row; it is ordered by rows, with row 1 containing the first 40 elements, row 2 the next 40 elements, etc.**

## REFERENCES

NOTE. Further discussion of ACORN sequences is available at the ACORN website http://acorn.wikramaratna.org/index.html. Included on that website there is a page with a more comprehensive list of relevant ACORN references as well as links, pointing to downloadable versions, or to other sites where those references can be accessed and downloaded (see http://acorn.wikramaratna.org/references.html). Recent ACORN references (including REAMC Limited reports, papers and presentations) are available for download from the publications page of the REAMC Limited website, https://www.reamc-limited.com.

---

1 R.S. Wikramaratna, ACORN - A New Method for Generating Sequences of Uniformly Distributed Pseudo-random Numbers, *J. Comput. Phys.*, **83**, pp16-31, 1989.

2 R.S. Wikramaratna, Theoretical Background for the ACORN Random Number Generator, Report AEA-APS-0244, AEA Technology, Winfrith, Dorset, UK, 1992.

3 R.S. Wikramaratna, The Additive Congruential Random Number Generator – A Special Case of a Multiple Recursive Generator, *J. Comput. and Appl. Mathematics*, **261**, pp371–387, 2008. [doi: 10.1016/j.cam.2007.05.018].

4 R.S. Wikramaratna, Theoretical and Empirical Convergence Results for Additive Congruential Random Number Generators, *J. Comput. and Appl. Math.*, **233**, pp2302-2311, 2010. [doi: 10.1016/j.cam.2009.10.015].

5 R.S. Wikramaratna, Statistical Performance of Additive Congruential Random Number Generators Part 2 - Conjectures Concerning Seed Values Chosen Uniformly at Random, REAMC Report-003, Issue 2, August 2021, REAMC Limited, UK. *Note that this report was originally published as Issue 1 in January 2021; Issue 2 is unchanged apart from a few minor typographic corrections.* [Link for download is available at https://www.reamc-limited.com ]

6 R.S. Wikramaratna, Two Conjectures on Statistical Performance of ACORN Generators: Evidence for Orders 11 - 15, REAMC Report-004, August 2021, REAMC Limited, UK. [Link for download is available at https://www.reamc-limited.com ]

7 R.S. Wikramaratna, Statistical Performance of ACORN Generators: Evidence for Selected Orders 16 – 101, REAMC Report-006, May 2023, REAMC Limited, UK. [Link for download is available at https://www.reamc-limited.com ]

8 C. Shannon, Communication Theory of Secrecy Systems, *Bell System Technical Journal*. 28 (4): 656–715, 1949. [doi:10.1002/j.1538-7305.1949.tb00928.x]

9 R.S. Wikramaratna, The Centro-invertible Matrix: A New Type of Matrix Arising in Pseudo-random Number Generation, *Linear Algebra and Its Applications*, **434**, pp144-151, 2011. [doi: 10.1016/j.laa.2010.08.011].

10 R.S. Wikramaratna, Periodicity of ACORN Sequences with Arbitrary Order and Modulus, REAMC Report-001, March 2020. REAMC Limited, UK. [Link for download is available at https://www.reamc-limited.com ]

11 G. Marsaglia, The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, Florida State University, Florida, USA, 1995. (originally made available from http://stat.fsu.edu/pub/diehard ; since November 2019 has been available from https://github.com/jeffThompson/DiehardCDROM )

12 P. L'Ecuyer and R. Simard, TestU01: A C Library for Empirical Testing of Random Number Generators, *ACM Transactions on Mathematical Software*, **33**, 4, Article 22, 2007.

13 R.S. Wikramaratna, Statistical Testing of Additive Congruential Random Number (ACORN) Generators, Meeting on 'Numerical algorithms for high-performance computational science', April 2019, The Royal Society, London, UK. [Link for download is available at https://www.reamc-limited.com ]

14 R.S. Wikramaratna, The Additive Congruential Random Number (ACORN) Generator - pseudo-random sequences that are well distributed in k dimensions, University of Oxford Numerical Analysis Group Internal Seminar, June 2019. [Link for download is available at https://www.reamc-limited.com ]

15 R.S. Wikramaratna, Statistical Performance of Additive Congruential Random Number Generators Part 1 - Results of Testing Some Specific Seed Values, REAMC Report-002, November 2020. REAMC Limited, UK. [Link for download is available at https://www.reamc-limited.com ]