Security of Operations on Random Numbers: A Review

Tejas Sharma[1] and Ashish Kundu[2]

[1] Student, Computer Science, IIT Bombay, Mumbai, India, 22b0909@iitb.ac.in
[2] Cisco Research, San Jose, USA, ashkundu@cisco.com

Abstract. Random numbers are often used in cryptography algorithms, protocols, and in several security and non-security applications. Such usages often apply Arithmetic and Boolean operations on pseudorandom numbers, such as addition, XOR, NOT, bit shifts, and other operations, in order to achieve the desired amount of entropy and desired level of security. In this paper, we have reviewed, studied, and analyzed the security properties of these operations on random numbers: do Arithmetic and Boolean operations and other related operations on cryptographically secure pseudorandom numbers lead to cryptographically secure pseudorandom numbers; do they lead to loss of preservation of entropy?

Keywords: Entropy · Cryptographically Secure Random Numbers · Arithmetic Operations · Boolean Operations and Security Analysis.

1 Introduction

Random numbers are used in cryptography algorithms and protocols, their implementations, and other security scenarios. Sometimes cryptographers as well as non-cryptographers use Arithmetic and Boolean operations on cryptographically secure pseudorandom numbers (CSPRN) [14]. They use addition, XOR, NOT, bit shifts, and other operations in order to achieve the desired amount of entropy and the desired level of security of CSPRNs in that process. Cryptography protocols such as Skein Hash function [7], ChaCha20 [3] use addition operations. In other protocols, such as in the IEEE 802.11 standard for Wi-Fi [1], protocol steps split random numbers from n bits to $x < n$ bits.

The tacit assumption based on which such protocols and implementations sometimes split random numbers, add, subtract, and carry out arithmetic and boolean operations on random numbers, is that the output of such operations on CSPRNs remains cryptographically secure and is a CSPRN.

Research Questions: Here are some questions[3] in the context of entropy and random numbers suitable for cryptographic purposes.

[3] We assume these are research questions unless and until there is evidence to the contrary.

1. Do Arithmetic operations on CSPRNs result in a CSPRN?
2. Do Boolean operations on CSPRNs result in a CSPRN (XOR and NOT are known to result in a CSPRN; AND, OR do not.)
3. Do Splitting and bit selection on CSPRNs result in a CSPRN?

In this paper, we have attempted to study and review the security properties of these questions: Do Arithmetic and Boolean operations and other related operations on cryptographically secure pseudorandom numbers lead to cryptographically secure pseudorandom numbers? What operations lead to insecure outputs? It is out of scope for this paper to address these questions for pseudorandoms that need not be suitable for cryptographic operations but are used widely across different topics of Statistics and Mathematics.

We have also verified our answers to the above questions with NIST randomness test suite results.

Quantum Random Numbers: The results in this paper are applicable not only to classically generated random numbers/entropy but also to quantum random numbers/quantum entropy [10, 16, 18].

Summary of our Contributions: In Table 1, we have summarized the results of our review on the security of different operations on CSPRNs.

Operation on CSPRNs	Is the result a CSPRN?	Section
XOR	Yes	3.1
NOT	Yes	3.2
AND	No	3.3
OR	No	3.4
Addition (Signed and unsigned)	Yes	4.1
Subtraction (Unsigned)	Yes	4.2
Multiplication	No	4.3
Division	No	4.4
Modulo	No	4.5
Splitting	Yes	5.1
Bit selection	Yes	5.2

Table 1. Security of operations on CSRPNs

2 Background

2.1 Notations Used in Our Paper

In the rest of the paper, we use the terms randoms and CSPRNs interchangeably; the terms randomness, entropy and Shannon's entropy interchangeably.

– Binomial Theorem:

$$\forall a, b, n \qquad (a+b)^n = \sum_{i=0}^{n} \left(\binom{n}{i} a^i b^{n-i} \right)$$

where $\binom{n}{i}$ is the number of subsets of i elements off n elements or a number of n-bit numbers with i 1's. Clearly, $\forall n \ \binom{n}{0} = \binom{n}{n} = 1$ And,

$$\forall n, i \ \binom{n}{n-i} = \binom{n}{i} = \frac{n}{i}\binom{n-1}{i-1}$$

– r is a random number with digits denoted from LSB (r_0) to MSB(r_{n-1}) and i_j being bits taken off a constant i. r_1 and r_2 are also random numbers, typically used as input variables. r follows a distribution as the output.
– Integration:

$$\int_a^b \frac{1}{x} dx = ln\left(\frac{b}{a}\right)$$

2.2 Entropy an n-bit CSPRN

The notion of cryptographically secure randoms rely on Shannon's entropy [4].

Theorem 1. *The Shannon Entropy of a truly random n-bit number is n.*

Theorem 2. *The maximum entropy of a discrete random variable with n states is $log_2(n)$, when all states are equally likely.*

3 Review of Logic Operations and their Security

In this section, for the sake of completeness, we review known results such as for bitwise XOR, NOT, AND, and OR. We have provided proofs of the security claims for each of these operations on CSPRNs.

3.1 Bitwise XOR of Random Numbers

Theorem 3. *Bitwise XOR between 2 CSRPNs, as well as between a CSRPN and an n-bit constant number, preserves entropy and results in a CSRPN.*

3.2 Bitwise NOT

Theorem 4. *Bitwise NOT operation on a CSRPN results in a CSRPN, and it preserves entropy.*

Proof. Consider $\neg_n\, r$. Here, each bit is flipped.

$$\forall j \quad P\left((\neg_n r)_j = 1\right) = P(\neg r_j = 1) = P\left(r_j = 0\right) = \frac{1}{2}$$

$\neg r_j$ has only one dependency: r_j, implying that all bits of the results are independently random. This means that all numbers between 0 and $2^n - 1$ are still equally likely, all with a probability of 2^{-n}, as also evidenced by the fact that $f(x) = \neg_n x$ is a bijection on the n-bit domain. The entropy as computed remains n by a similar argument, thus preserving the randomness. □

3.3 Bitwise AND

Theorem 5. *Bitwise AND between 2 CSRPNs or between a CSRPN and an n-bit constant number does **not** preserve entropy.*

Proof. Take $r = r_1 \;\&\; r_2$, the bitwise AND. Here,

$$P(r_j = 1) = P\left(r_{1j} = 1 \;\&\; r_{2j} = 1\right) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

This already hints that all bits are biased towards 0, with only a 25% chance of being 1. The Shannon entropy is

$$\begin{aligned}
\mathbb{H}[\mathbf{R}] = & -\sum_{i=0}^{2^n-1} P(x_i) \cdot \log_2\left(P(x_i)\right) \\
= & -\sum_{i=0}^{2^n-1} \left(\prod_{j=0}^{n} \left(P(r_j = i_j)\right) \cdot \sum_{j=0}^{n} \log_2 P(r_j = i_j) \right) \\
= & -\sum_{j=0}^{n} \left(\binom{n}{j} \frac{3^{n-j}}{4^n} \cdot \left(-2n + (n-j) \cdot \log_2(3) \right) \right)
\end{aligned}$$

ordering numbers i by number of bits which are 1, a.k.a. j where $P(r_j = i_j) = \frac{1}{4}$

$$\begin{aligned}
= & \; (2 - \log_2 3)\, n \cdot \sum_{j=0}^{n} \left(\binom{n}{j} \left(\frac{1}{4}\right)^j \left(\frac{3}{4}\right)^{n-j} \right) \\
& + \; n \frac{\log_2 3}{4} \cdot \sum_{j=1}^{n} \left(\binom{n-1}{j-1} \left(\frac{1}{4}\right)^{j-1} \left(\frac{3}{4}\right)^{n-j} \right) \\
= & \; (2 - \log_2 3)\, n + n \frac{log_2 3}{4} = \; n \cdot \left(2 - \frac{3}{4} \log_2 3 \right) \sim 0.811 \cdot n
\end{aligned}$$

As evident, the entropy is less than n. Thus, the result of bitwise AND is **not** random.

Let us now consider $r = r_1 \;\&\; c$, where c is an n-bit constant.

$$\forall j \quad P(r_j = 1) \;=\; \begin{cases} 0\,, & c_j = 0 \implies r_j = 0 \quad \text{This bit is deterministic.} \\ \frac{1}{2}\,, & c_j = 1 \implies r_j = r_{1j} \quad \text{This bit is random.} \end{cases}$$

Suppose we cut out the 'deterministic' zero bits (where $c_j = 0$) and have $n_1 < n$ independent and random bits left over. These are the bits that contribute to the randomness. There are 2^{n_1} possible values these bits can take, taken all at a time. All of these are equally likely. This hints at an n_1-bit number and an entropy of n_1. Since there is a one-to-one mapping between possible results and these n_1-bit random numbers, we say that the entropy of the result is n_1.

This entropy is less than n unless $c = 1111..11111$ and depends on the value of c. Therefore, bitwise AND does **not** preserve Shannon's entropy. □

3.4 Bitwise OR

Theorem 6. *Bitwise OR between 2 CSRPNs or between a CSRPN and an n-bit constant number does **not** preserve entropy.*

Proof. Take $r = r_1 \mid r_2$, the bitwise OR. Here,

$$P(r_j = 0) = P\left(r_{1j} = 0 \;\&\; r_{2j} = 0\right) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

This already hints that all bits are biased towards 1, with only a 25% chance of being 0. By an exact argument, as in the case of bitwise AND, the entropy of the result is around $0.811n$, implying a decrease in Shannon entropy.

Let us now consider $r = r_1 \mid c$, where c is an n-bit constant.

$$\forall j \quad P(r_j = 1) \;=\; \begin{cases} \frac{1}{2}\,, & c_j = 0 \implies r_j = r_{1j} \quad \text{This bit is random.} \\ 1\,, & c_j = 1 \implies r_j = 1 \quad \text{This bit is deterministic.} \end{cases}$$

Following an identical argument with bitwise AND (except that it is now the constant '1' bits that are pruned), we conclude that bitwise OR does not **not** preserve randomness.

□

4 Arithmetic Operations and their Security

4.1 Addition of Random Numbers

The objective is to determine that the resultant number on the addition of two random numbers is also random according to the definition of randomness used throughout this paper, only for cryptography purposes.

We define the addition of two numbers $r = r_1 + r_2$ in a bitwise manner, with the help of the full adder [13].

$$\begin{aligned} \forall j \quad & r_j = r_{1j} \oplus r_{2j} \oplus c_j \\ \forall j \quad & c_{j+1} = (r_{1j} \ \& \ r_{2j}) \mid (r_{2j} \ \& \ c_j) \mid (c_j \ \& \ r_{1j}) \\ & c_0 = 0 \end{aligned}$$

We study the entropy of each bit and, thus, the entire number, assuming both numbers are random and that any one is random.

Theorem 7. *The least significant bit of the result is truly random iff at least one of the numbers has a random LSB.*

Proof. For the LSB,

$$r_0 = r_{10} \oplus r_{20} \oplus 0 = r_{10} \oplus r_{20}$$

We use Theorem 2 to say that the LSB is computationally indistinguishable from being random if at least one of r_{10} and r_{20} is computationally indistinguishable from being truly random or even is a truly random bit. □

Theorem 8. *If at least one of the bits r_{1j} and r_{2j} is random for any index j, we say, regardless of the value and the entropy of the bit c_j, that r_j remains random.*

Proof. Without loss of generality, let r_{1j} be the random bit, at least computationally indistinguishable from being truly random and independent of all other bits. We treat

$$\text{temp}_j = (r_{2j} \oplus c_j)$$

as a 1-bit number. If p were the probability that temp_j were equal to 1, then the chance that the result bit is 1 is

$$P(r_{1j} = 1 \ \& \ \text{temp}_j = 0) + P(r_{1j} = 0 \ \& \ \text{temp}_j = 1) = \frac{1-p}{2} \ + \ \frac{p}{2} = \frac{1}{2}$$

$$P(r_{1j}) = \frac{1}{2} + \eta \implies P(r_j = 1) = \frac{1}{2} + (1-2p)\eta$$

not any further from a 50 % chance, by more than the value η (so it does not matter if p depends on other bits of r_2 or not). This means that if at least one of the two input numbers is computationally indistinguishable from truly random numbers, so is the result. □

Theorem 9. *The carry bits c_j are not truly random, although they approach a 50 % randomness as the index of the bit increases if both r_1 and r_2 are random.*

Proof. Let $P(c_j = 1) = 1/2 - e$ for any $e < 1/2$. With the help of the inclusion and exclusion principle applied to $P(c_{j+1} = 1)$, denoted $P(c_{j+1})$, we say that

$$P(c_{j+1}) \;=\; P(c_j \;\&\; r_{1j}) + P(r_{1j} \;\&\; r_{2j}) \;+\; P(c_j \;\&\; r_{2j}) - 2P(c_j \;\&\; r_{1j} \;\&\; r_{2j})$$

Without loss of generality, let r_1 be truly random. Then, this evaluates to

$$1/2\,(P(r_{2j}) + P(c_j))$$

which could lie anywhere between 0 and 1. If r_2 is also truly random, we say that $P(c_0) = 0$ snf

$$P(c_{j+1}) \;=\; 1/2\,(1/2 + P(c_j)) \quad \Longrightarrow \quad P(c_j = 1) \;=\; 1/2\left(1 - 2^{-j}\right)$$

In particular, if the result is not kept within the domain of n-bit numbers but is allowed to take an extra bit ('BigInteger' addition), then the n^{th} bit, $r_n = c_n$, while it approaches a random bit as n increases, the sharing of dependencies with r_{n-1} on $r_{1,n-1}$, $r_{2,n-1}$ and c_{n-1} makes the pair of bits and as a result, the entire number **not** random. □

Theorem 10. *Addition of 2 CSRPNs, as well as a CSRPN and an n-bit constant number, results in a CSRPN and preserves entropy, provided the result is truncated to n bits, but BigInteger addition with overflow to the $n+1^{th}$ bit does not result in a CSRPN.*

Proof. **For Integer Addition**: Without loss of generality, let r_1 be the random. This is a direct consequence of the above three Lemmas. An alternate approach would be to weigh the chances of each result. Then,

$$P(r = i) \;=\; \sum_{i_1=0}^{2^n-1} P(r_1 = i_1)P(r_2 = i - i_1) \;=\; \frac{1}{2^n}\sum_{i_1=0}^{2^n-1} P(r_2 = i - i_1)$$

Note that $f(x) = i - x$ is a bijection in the n-bit number domain, the same for $x = i_2$. Meaning: our probability is

$$P(r = i) \;=\; 2^{-n}\sum_{i_2=0}^{2^n-1} P(r_2 = i_2) \;=\; 2^{-n}$$

This is the probability regardless of the i value. This means that r is truly random, and Shannon entropy is preserved in addition and subtraction.

For Unsigned BigInteger Addition: The result would follow a triangular distribution with

$$P(r = i) \quad = \sum_{i_1=max(0,i-2^n+1)}^{min(i,2^n-1)} P(r_1 = i_1)P(r_2 = i - i_1)$$

Since i_2 is an n-bit number, $i_2 = i - i_1 \leq 2^n - 1 \implies i_1 \geq i - 2^n + 1$ and $i_1 \geq 0$. Combining these, we get $i_1 \geq \max(i - 2^n + 1,\ 0)$. Likewise, $i_2 \geq 0 \implies i_1 \leq i$ since $i_1 \leq 2^n - 1$, the maximum number in the domain.

$$\begin{aligned} P(r = i) &= \sum_{i_1 = \max(0, i - 2^n + 1)}^{\min(i, 2^n - 1)} \frac{1}{2^n} \frac{1}{2^n} \\ &= \frac{1}{2^{2n}} \cdot (\min(i + 1, 2^n) - \max(0, i - 2^n + 1)) \\ P(r = i) &= \begin{cases} i + 1, & i < 2^n \\ 2^{n+1} - i - 1, & i \geq 2^n \end{cases} \end{aligned}$$

This means the distribution is not uniform, and the entropy would be less than $n + 1$. However, it would be more than n as truncating one bit in the result only reduces net information (and therefore entropy). So, while entropy increases, the result is not random. □

4.2 Unsigned Integer Subtraction

Theorem 11. *Subtraction of 2 CSRPNs, as well as a CSRPN and an n-bit constant number, results in a CSRPN and preserves entropy, provided overflow is taken into account.*

Proof. Subtracting a number r from any given number x has the same effect of adding the 2-s complement of r, $2^n - r$ to x; the reason being that

$$(x - r)\%2^n = (x + 2^n - r)\%2^n$$

From the previous result, if x is truly random, so is the result. If r were random but x were not, then $2^n - r$ would have and identical but inverted probability distribution and therefore would be random; so would $x + 2^n - r$. □

4.3 Integer Multiplication

We define the integer product of two numbers $r_0 = r_1 \cdot r_2$ by these definitions, taking

$$r_{1j} = r_{2j} = 0 \quad \forall\ j \geq n$$

where the two numbers are n-bit – adding zeroes to the left. Note that truncating the result to n bits would involve ignoring the values of all bits left-shifted or carried to and beyond the n^{th} place, taking into account the definition of addition, as described above.

Theorem 12. *Multiplication of 2 CSRPNs does not result in a CSRPN, and the entropy of the result is reduced.*

Proof.

$$P(r_{00} = 1) = P(r_{10} = 1 \text{ and } r_{20} = 1) = P(r_{10} = 1) \cdot P(r_{20} = 1) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

This itself disproves the randomness of the product. □

Distribution of Probability of Result: Here we do not follow a bitwise approach, but rather an approach that evaluates the probability distribution of each n-bit or $2n$-bit result, among the domain of n-bit or $2n$-bit numbers. The exact distribution is more haphazard, for prime numbers have a very low probability, and the likelihood that the result equals any number (for $2n$ bit result) is proportional to the number of factors; however, it follows a decreasing trend which approaches the continuous distribution

$$f_X(x) = -ln(x) \text{ [6]}$$

That is, it is more likely for the product to be a small number within the $2n$-bit space. The variance is even more random if the result is truncated to n bits.

Calculated Entropy: With the help of a brute-force algorithm, the entropy values for multiplying two truly random numbers, with the results truncated to n bits and the results kept at the full $2n$ bits, were computed.

n, #**bits**	**Entropy of** r_1, r_2	$2n - bit$ **product entropy**	$n - bit$ **product entropy**
1	1	1.500000	0.811278
2	2	3.077820	1.750000
3	3	4.788910	2.727217
4	4	6.446142	3.718139
5	5	8.267194	4.714364
6	6	10.053356	5.712749
7	7	11.894136	6.712042
8	8	13.754086	7.711729
9	9	15.647446	8.711588
10	10	17.548259	9.711524
11	11	19.461689	10.711494
12	12	21.384701	11.711481

Table 2. Entropy on multiplication of random numbers

Remarks: While it is true that multiplication is repeated addition, when we perform $a * b$, the number of times we add a to itself is dependent on b and is not fixed. Moreover, $a + a = 2 * a$ is not as random as just a within an n-bit integer field, since the least significant bit is 0. Therefore, we cannot say that $a * b$ is as random as a or b.

4.4 Integer Division

Theorem 13. *Integer Division of 2 CSRPNs does not result in a CSRPN, and entropy is not preserved.*

Proof. Let us consider the floor division r_1/r_2 or the Python $r_1//r_2$ since we deal with integers. The result is always zero whenever $r_1 < r_2$ by magnitude (minus one in Python if r_1 is negative). This means there is an almost 50 % chance that the result is zero. If we stick to positive (aka unsigned) numbers, the probability that the result is zero is $(2^n - 1)/2^{n+1}$.

This itself proves that division does not preserve randomness. □

For the probability that $r = i$, Let $k = 2n/i$, the floor, and $2n = ki + c$. The ordered pairs

$$\begin{aligned}&(1, i);\\&(2, 2i); (2, 2i + 1);\\&\quad\vdots\\&(k, ki); (k, ki + 1); ...(k, ki + k - 1)\end{aligned}$$

consist of k sequences, with less than or equal to $k(k + 1)/2$ terms, less than if some of the second numbers in these ordered pairs go above $2^n - 1$. Each ordered pair (a, b) represents one possibility for r_2 and r_1 in this order, where the result, r_1/r_2, would be i. Plus, the likelihood of any ordered pair (a, b) is 2^{-2n}. This means that

$$P\left(\frac{r_1}{r_2} = i\right) \leq \frac{\left(\frac{2^n}{i}\left(\frac{2^n}{i} + 1\right)\right)}{(2^{2n+1})}$$

This gives a probability distribution centered at zero and decreases approximately quadratically with i, the result. This is not a uniform random distribution; the entropy is less than n.

4.5 Modulo

Theorem 14. *The Modulo (Remainder) of 2 CSRPNs is not a CSRPN, and entropy is not preserved.*

Proof. The modulo operation on randoms behaves differently. For the result $r = r_1 \% r_2 = i$, the divisor r_2 should be greater than i. Let it be $i_2 > i$. Then, the number of possibilities of i_1 to result in modulo i, is

$$\left\lceil \frac{2^n - i}{i_2} \right\rceil$$

These possibilities result from the series $i_1 = i,\ i + i_2,\ i + 2i_2, ...$
Summed up across all different values of i_2, the number of (i_1, i_2) pairs is

$$\sum_{j=i+1}^{2^n-1} \left\lceil \frac{2^n - i}{j} \right\rceil$$

One lower bound for this number of ordered pairs is obtained on adding the fractions, and an upper bound is obtained on adding 1 to each fraction [or adding $2^n - i - 1$ to the lower bound]. Dividing by the total number of ordered pairs (i_1, i_2) possible in the domain of n-bit numbers space gives us the probability distribution.

$$\begin{aligned} No.(r = i) &\geq \sum_{j=i+1}^{2^n-1} \frac{2^n - i}{j} \\ &\geq \int_{j=i+1}^{2^n} \frac{2^n - i}{j} dj \\ &\qquad \text{Upper bounds on decreasing function rectangle areas} \\ &= (2^n - i) ln \left(\frac{2^n}{i+1} \right) \end{aligned}$$

$$\begin{aligned} No.(r = i) &\leq 2^n - i - 1 + \sum_{j=i+1}^{2^n-1} \frac{2^n - i}{j} \\ &\leq 2^n - i - 1 + \int_{j=i}^{2^n-1} \frac{2^n - i}{j} dj \\ &\qquad \text{Lower bounds on decreasing function} \\ &= 1 + (2^n - i) \left(1 + ln \left(\frac{2^n - 1}{i} \right) \right) \end{aligned}$$

Therefore,

$$P[r = i] \in \left[\frac{2^n - i}{2^{2n}} ln \left(\frac{2^n}{i+1} \right),\ \frac{1}{2^{2n}} + \frac{2^n - i}{2^{2n}} \left(1 + ln \left(\frac{2^n - 1}{i} \right) \right) \right]$$

This is the required range. The probability decreases with i, the modulo itself, agreeing with the fact that for the modulo to be a small number, there are many divisors possible and many dividends for many of these divisors. These numbers shrink as the remainder needed increases, reducing the number of 'acceptable' ordered pairs for the values of r_1, r_2, and the probability that the result $r = i$.

Thus, this is not a uniform distribution and the 'randomness' of the remainder is reduced, with entropy less than n. □

5 Splitting and Bit Selection and their Security

5.1 Splitting of CSPRNs

Theorem 15. *For a truly random number or number with all bits computationally indistinguishable from truly random numbers, the entropy on splitting the random number or choosing any k bits out of n remains random.*

Proof. Let the k bits be at positions $j_1, j_2, \ldots j_k$. Each of these bits has a 50 % chance of being 1. This means that of the 2^k numbers possible by only those k bits, each equally likely with a probability of 2^{-k}. The computed Shannon Entropy is k, meaning the number is truly random.

This holds *regardless* of the value of $k \leq n$ and the actual choice of k bits. This also means splitting a random number into two would result in two random k-bit numbers and one random $n-k$-bit number. For the special case $k = \frac{n}{2}$, the number is split into two halves, two equally random numbers.

Caveat: If the pseudorandom number generated is not computationally indistinguishable [9] from a truly random number, a.k.a.

$$\exists\ j, \text{polynomial } Q(j) \text{ such that } \|P(r_j = 1) - 0.5\| > \frac{1}{Q(j)}$$

then either the j^{th} bit is dependent on the previous bit or it is simply not random enough. In these cases, truncating the number could reduce entropy by up to $n-k$ [for removing $n-k$ bits], but then again, the entropy of the original number would be less than n. □

5.2 Bit Selection from CSPRNs

Selection of bits at random indices of a CSPRN such that the number of such selected bits is sufficiently large. Entropy extraction has been studied earlier [17]. In the following, we are reviewing the security of this bit selection operation and providing proof of its security.

Theorem 16. *Choosing k random indices between 0 and $n-1$ and taking the corresponding bits off an n-bit random number would give a random number.*

Proof. There are $\binom{n}{k}$ possible choices of bits. Multiplied by $k!$, we get the number of choices with the order of bits taken into account to form our k-bit number. Let our permutation be p. Then,

$$P(p) = 1/\left(k!\binom{n}{k}\right)$$

And, $P(i|p) = 2^{-n}$ from Theorem 11. From Bayes' [2] Theorem,

$$P(i) \;=\; \sum_p P(p)P(i|P) \;=\; \sum_p \left(\frac{1}{k!\binom{n}{k}} \frac{1}{2^k} \right) \;=\; \frac{1}{2^k} \sum_p \frac{1}{k!\binom{n}{k}} = \frac{1}{2^k}$$

From this, it follows that $\forall i \; P(r = i) = 2^{-k}$ and therefore, the obtained number is truly random. □

6 Implementation and Experimental Analysis

With the help of the NIST Test suite [11], we tested the results using 128 bitstreams of 1.5 million bits per stream taken off the hardware-accelerated random generator (`/dev/random`) on an M1 MacBook Air with 8 GB RAM and 256 GB SSD. The suit consists of 15 types of tests, some repeated many times but with different hyperparameters involved, totalling around 150 types of tests. 2 types of results are produced:

- **Passing rate**: the number of bitstreams that get a P-value more than a given threshold. This could be around 0.01. If too many bitstreams get P-values less than the significance level α, then it means that the chances that (True RNG does better than bitstream) are too high, too many bitstreams produced are significantly less random according to the test.
- **P-value**: this reported number ranging from 0.0 to 1.0 is **not** the average of P-value or P-value of one bitstream. This is the analysis of the P-values distribution across bitstreams; if it deviates too much from uniformity (as against a perfect RNG), then this test reports a failure (low P-value reported). Here, 0.0001 is taken as the threshold; even if all bitstreams individually have high P-values, a distribution too far from uniform indicates that there is something less than random about it, another means of testing.

6.1 Procedure to get Files for Operation Testing

- Open the input file containing random bits.
- Read 128 bits at a time into 2 64-bit numbers a and b.
- Perform the operation like $a \oplus b$, $a + b$, $a - b$. Write the results as 64 bits into the corresponding output file. For bitwise NOT, write $\neg a$ and $\neg b$ both, that is 128 bits.
- Repeat for the entire input file. We get the output file size:
  $\text{size(output\_file)} = \eta \cdot \text{size(input\_file)}$, where
  $$\eta = \begin{cases} 1, & \text{Bitwise NOT and unary operations} \\ 0.5, & \text{Binary operations like bitwise AND, OR, XOR and } +, \text{-}, *, /, \% \end{cases}$$

6.2 Analysis of Experimental Results

Operations	**Tests passed**	**P-values**	**Inference**	**Corroborates Analysis?**
Operands	All tests passed bitstreams passed: $\geq$ 124 / 128, 91 / 94	$\geq$ 0.01	Random	Agrees with Definition
Bitwise NOT	All tests passed bitstreams passed: $\geq$ 124 / 128, 90 / 94	$\geq$ 0.03	Random	Agrees with Theorem 4
Bitwise AND	All tests failed except rank tests bitstreams passed: 0 / 128	All 0.0	Not Random Too many 0s	Agrees with Theorem 5
Bitwise OR	All tests failed except rank tests bitstreams passed: 0 / 128	All 0.0	Not random Too many 1s	Agrees with Theorem 6
Bitwise XOR	All tests except one passed Failed test: 122 / 128 threshold: 123 / 128	0.45 for that 'failed' test acceptable for the rest	Random	Agrees with Theorem 3
Addition	Random Ecursions Variant one test failed All other tests: $\geq$ 123 / 128 bitstreams passed Random Excursions: $\geq$ 86 / 88 passed	$\geq$ 0.01	Random	Agrees with Theorem 10
Subtraction	Non-overlapping Template Matching one test failed rest: $\geq$ 123 / 128	$\geq$ 0.03	Random	Agrees with Theorem 11
Multiplication	Longest run, Rank, Universal, Linear complexity tests passed 1 / 2 serial tests failed (71 / 128 bitstreams passed) many Template matching and most other tests failed	Several 0.0s	Not random	Agrees with Theorem 12
Division	All tests failed with 0 / 128 bitstreams passed	All 0.0	Not random	Agrees with Theorem 13
Modulo	Longest run, Rank, Universal, Linear complexity tests passed 1 / 2 serial tests failed (71 / 128 bitstreams passed) many Template matching and most other tests failed	Several 0.0s	Not random	Agrees with Theorem 14

Table 3. Summary of Experimental Results

7 Related Work

Arithmetic operations such as the addition of cryptographically random numbers have been used in some form in [3] and [7]. Random variables that need not support security requirements and their algebraic constructs have been studied nu Dale [6]. For randoms that are not cryptographically secure, [12] discusses Arithmetic operations on randoms. XOR operations on randoms support security [14]. The proofs used in this paper rely on Shannon's Entropy, as described in [4]. We also used the mathematical logic behind the full adder as described in [13]. The NIST test suite, as described in [11], is used in our experiments.

To the best of our knowledge, we have yet to find a closely related work that performs a detailed security analysis of a set of arithmetic and boolean operations on random numbers used for cryptography.

8 Conclusion

In this paper we reviewed the security of Arithmetic, Boolean, Splitting and Bit Selection operations on CSPRNs. NOT, XOR, ddition and subtraction, with results truncated to n bit numbers, preserve randomness, but the same cannot be said for AND, OR, multiplication, division, and modulo of random numbers.

This should be taken into account when designing random number generators; using XOR or addition with random numbers increases security of pseudo-random hashing algorithms.

References

1. Iso/iec international standard - information technology telecommunications and information exchange between systems local and metropolitan area networks specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Medium access control (mac) security enhancements. *ISO/IEC 8802-11, Second edition: 2005/Amendment 6 2006: IEEE STD 802.11i-2004 (Amendment to IEEE Std 802.11-1999)*, pages c1–178, 2004.
2. Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1763.
3. Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Citeseer, 2008.
4. C.E.Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.
5. C.E.Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 1949.
6. Melvin Dale. The algebra of random variables. *New York [ua]: Wiley*, 1979.

7. Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The skein hash function family. *Submission to NIST (round 3)*, 7(7.5):3, 2010.
8. E.K. Godunova. Jensen inequality. *Encyclopedia of Mathematics*, 2001.
9. Oded Goldreich; Shafi Goldwasser; and Silvio Micali. How to construct random functions. *Journal of the Association of Computing Machinery*, 1986.
10. Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. Quantum random number generators. *Rev. Mod. Phys.*, 89:015004, Feb 2017.
11. https://csrc.nist.gov/pubs/sp/800/22/r1/upd1/final. NIST Test Suite.
12. Szymon Jaroszewicz and Marcin Korzeń. Arithmetic operations on independent random variables: a numerical approach. *SIAM Journal on Scientific Computing*, 34(3):A1241–A1265, 2012.
13. Charles W Kann. Digital circuit projects: an overview of digital circuits through implementing integrated circuits. 2014.
14. Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.
15. Frank Miller. Telegraphic code to ensure privacy and secrecy in the transmission of telegrams. cm cornwell,. *C.M. Cornwell*, 1882.
16. M. Stipčević. Quantum random number generators and their use in cryptography. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 1474–1479, 2011.
17. Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 32–42. IEEE, 2000.
18. Jiapeng Zhao, Eneet Kaur, Michael Kilzer, Yihan Liu, Stephen DiAdamo, Hassan Shapourian, Ramana Kompella, and Reza Nejabati. Over 40 gbps tri-type quantum random number generator. In *Quantum Computing, Communication, and Simulation V*, volume 13391, pages 225–231. SPIE, 2025.

Appendix:
Proof of Theorem 1:

Proof. There are 2^n possible numbers, all of which are equally likely. When looked at bitwise, each bit is independent of all other bits, be it pairwise, three at a time, $k < n$ at a time, or all at a time; it has a 50 % chance of being 1, regardless of the other bit values. □

Proof of Theorem 2:

Proof. Let the probabilities of the states $x_0, x_1, \ldots x_{n-1}$ be $p_0, p_1, \ldots p_{n-1}$. Then, the entropy is

$$\mathbb{H}[X] = -\sum_{i=0}^{n-1} p_i \log_2(p_i) \quad \Longrightarrow \quad 2^{-\mathbb{H}[x]} = \prod_{i=0}^{n-1} p_i^{p_i}$$

Further,

$$\sum_{i=0}^{n-1} p_i = 1$$

Let

$$f(\boldsymbol{p}) = 2^{-\mathbb{H}[X]}$$

We require a global minimum for $f(\boldsymbol{p})$, which is continuous in the hypercube between $(0, 0, \ldots 0)$ and $(1, 1, \ldots 1)$ on the n-dimensional hyperspace for $\boldsymbol{p}$, constrained by the fact that the sum of probabilities is one.

In particular, let us look at $g(\boldsymbol{p}_1)$ in which the value of p_0 is filled automatically by the values of $p_1, p_2, \ldots p_n$, which form the vector $\boldsymbol{p}_1$ and $g(\boldsymbol{p}_1) = f(\boldsymbol{p})$. We require a global minima for $g(\boldsymbol{p}_1)$ which means that $\boldsymbol{\nabla} g(\boldsymbol{p}_1) = 0$ and $\nabla^2 g(\boldsymbol{p}_1) > 0$. For each $i > 0$,

$$\begin{aligned}\frac{\partial}{\partial p_i} g(\boldsymbol{p}_1) &= g(\boldsymbol{p}_1)(1 + ln(p_i)) - (1 + ln(K - p_i))g(\boldsymbol{p}_1) \\ &= g(\boldsymbol{p}_1)\left(ln(p_i) - ln\left(1 - \sum_{i=1}^{n-1} p_i\right)\right)\end{aligned}$$

This expression has to be zero for all i, so that the condition for minima $\boldsymbol{\nabla} g(\boldsymbol{p}_1) = \mathbf{0}$ is met; which implies that

$$\forall i \quad ln(p_i) = ln\left(1 - \sum_{i=1}^{n-1} p_i\right)$$

which is possible only if

$$p_1 = p_2 = \ldots p_{n-1} = \frac{1}{n}$$

For this point,

$$\frac{\partial^2}{\partial p_i^2} g(\boldsymbol{p}_1) \;=\; 2n g(\boldsymbol{p}_1)$$

When added for all the i values, we get

$$\nabla^2 g(\boldsymbol{p}_1) = 2n(n-1) g(\boldsymbol{p}_1)$$

which is a positive value, meaning that g has a minima. From this, we also get $p_0 = 1/n$, which coincides with the obvious, symmetric result – this is the only minima, and therefore the global minima within the plane.

With all probabilities being $1/n$,

$$f(\boldsymbol{p}) = 2^{-\mathbb{H}[X]}$$

subject to the constraint of total probability being unity, reaches a minimum, meaning that the entropy reaches a maximum and the only maximum. At this point, the entropy is

$$-\sum_{i=0}^{n-1} \frac{1}{n} log_2 \frac{1}{n} = +log_2 n$$

Therefore, this is the maximum possible entropy for the given discrete r.v.
This can also be verified by Jensen's inequality [8] , with $f(x) = -\log_2(x)$:

$$\begin{aligned} f_{\text{convex}}\left(\sum_{i=0}^{n-1}\lambda_i x_i\right) &\leq \sum_{i=0}^{n-1}\lambda_i f_{\text{convex}}(x_i) \\ -\log_2\left(\sum_{i=0}^{n-1} p_i \cdot \frac{1}{p_i}\right) &\leq \sum_{i=0}^{n-1} p_i \cdot \left(-\log_2\left(\frac{1}{p_i}\right)\right) = \sum_{i=0}^{n-1} p_i \cdot \log_2(p_i) \\ \mathbb{H}[\mathbf{X}] &= \sum_{i=0}^{n-1} p_i \cdot (-\log_2(p_i)) \leq \log_2(n) \end{aligned}$$

□

Proof of Theorem 3:

Proof. Let the two random numbers be r_1 and r_2. Once again, we assume they are truly random and independent. Entropy is maximum when all events are equally likely, given the number of events. That value equals $\log_2$(no. of events), which equals n, corresponding to a truly random number. If $r = r_1 \oplus r_2$ is the result of bitwise XOR,

$$P(r_j = 1) = P\Big((r_{1j} = 1 \textit{ and } r_{2j} = 0) \textit{ or } (r_{1j} = 0 \textit{ and } r_{2j} = 1)\Big) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

for all j, due to the mutual exclusiveness of events separated by *or*.
In the case of one random number r_1 and one constant c, consider bit r_j.

$$\forall j \quad P(r_j = 1) = \begin{cases} \frac{1}{2}\,, & c_j = 0 \implies r_j = r_{1j} \quad \text{This bit is random.} \\ \frac{1}{2}\,, & c_j = 1 \implies r_j = \neg\, r_{1j} \quad \text{This bit is random.} \end{cases}$$

In all these cases, the result bit r_j depends only on r_{1j} and r_{2j}/c_j; and is independent of any other bit in either of the two random numbers, consequently any other bit $r_k \forall k \neq j$. All bits are independent, so the likelihood of result i is

$$P(r = i) = \prod_{i=0}^{n-1} P(r_j = i_j) = \prod_{i=0}^{n-1} \frac{1}{2} = 2^{-n}$$

Just like the truly random number, the entropy is n.
The case with one constant (the message) and one random number (for the key) is the principle behind the One Time Pad. [5, 15]. □