Close to Uniform Prime Number Generation
With Fewer Random Bits
Pierre-Alain Fouque and Mehdi Tibouchi

Abstract—In this paper, we analyze several variants of a simple method for generating prime numbers with fewer random bits. To generate a prime $p$ less than $x$, the basic idea is to fix a constant $q \sim x^{1-\varepsilon}$, pick a uniformly random $a < q$ coprime to $q$, and choose $p$ of the form $a + t \cdot q$, where only $t$ is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring a few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem. We argue that this approach has a number of desirable properties compared with the previous algorithms, at least in an asymptotic sense. In particular: 1) it uses much fewer random bits than both the “trivial algorithm” (testing random numbers less than $x$ for primality) and Maurer’s almost uniform prime generation algorithm; 2) the distance of its output distribution to uniform can be made arbitrarily small, unlike algorithms like PRIMEINC (studied by Brandt and Damgård), which we show exhibit significant biases; and 3) all quality measures (number of primality tests, output entropy, randomness, and so on) can be obtained under standard conjectures or even unconditionally, whereas most previous nontrivial algorithms can only be proved based on stronger, less standard assumptions like the Hardy–Littlewood prime tuple conjecture. Note, however, that our analysis involves non-explicit constants, and therefore does not establish the superiority of our approach for concrete parameter sizes.

Index Terms—Prime number generation, randomness, complexity, cryptography, analytic number theory.

I. INTRODUCTION

THERE are several ways to assess the quality of a random prime generation algorithm, such as its speed (time complexity), its accuracy (the probability that it outputs numbers that are in fact composite), its statistical properties (the regularity of the output distribution), and the number of bits of randomness it consumes to produce a prime number (as good randomness is crucial to key generation and not easy to come by [2]).

In a number of works in the literature, cryptographers have proposed fast prime generation algorithms [3]–[6] or algorithms providing a proof that the generated numbers are indeed prime numbers [7]–[9].

A number of these works also prove lower bounds on the entropy of the distribution of prime numbers they generate, usually based on very strong conjectures on the regularity of prime numbers, such as the prime $r$-tuple conjecture of Hardy and Littlewood [10]. However, such bounds on the entropy do not ensure that the resulting distribution is statistically close to the uniform distribution: for example, they do not preclude the existence of efficient distinguishers from the uniform distribution, which can indeed be shown to exist in most cases.

But some cryptographic protocols (including most schemes based on the Strong RSA assumption, such as Cramer–Shoup signatures [11]) specifically require uniformly distributed prime numbers for the security proofs to go through.

Moreover, some cryptographers, like Maurer [7], have argued that even for more common uses of prime number generation, like RSA key generation, one should preferably generate primes that are almost uniform, so as to avoid biases in the RSA moduli $N$ themselves, even if it is not immediately clear how such biases can help an adversary trying to factor $N$.

To the authors’ knowledge, the only known prime generation algorithms for which the statistical distance to the uniform distribution can be bounded are the one proposed by Maurer [7], [8] on the one hand, and the trivial algorithm (viz. pick a random odd integer in the desired interval, return it if it is prime, and try again otherwise) on the other hand. The output distribution of the trivial algorithm is exactly uniform (or at least statistically close, once one accounts for the compositeness probability of the underlying randomized primality checking algorithm), and the same can be said for at least some variants of Maurer’s algorithm, but both of those algorithms have the drawback of consuming a very large amount of random bits.

By contrast, the PRIMEINC algorithm studied by Brandt and Damgård [4] (basically, pick a random number and increase it until a prime is found) only consumes roughly as many random bits as the size of the output primes, but we can show that its output distribution, even if it can be shown...
to have high entropy if the prime $r$-tuple conjecture holds, is also provably quite far from uniform, as we demonstrate in §IV-A. It is likely that most algorithms that proceed deterministically beyond an initial random choice, including those of Joye et al. [5], [6], exhibit similar distributional biases.

The goal of this paper is to achieve in some sense the best of both worlds: construct a prime generation algorithm that consumes much fewer random bits than the trivial algorithm while being efficient and having an output distribution that is provably close to the uniform one.

We present such an algorithm in §III: to generate a prime $p$, the basic idea is to fix a constant $q \sim \epsilon^{-2}$, pick a uniformly random $a < q$ coprime to $q$, and choose $p$ of the form $a + t \cdot q$, where only $t$ is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H.L. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem.

We stress that our analysis is purely asymptotic. As a result, more work is needed to establish its superiority for concrete parameter sizes. We discuss prospects of obtaining explicit bounds in §V.

II. PRELIMINARIES

A. Regularity MEASURES of Finite Probability Distributions

In this subsection, we give some definitions on distances between random variables and the uniform distribution on a finite set. We also provide some relations which will be useful to bound the entropy of our prime generation algorithms. These results can be found in [13].

**Definition 1 (Entropy and Statistical Distance):** Let $X$ and $Y$ be two random variables on a finite set $S$. The statistical distance between them is defined as the $\ell_1$ distance$^2$:

$$\Delta_1(X; Y) = \sum_{s \in S} \left| \Pr[X = s] - \Pr[Y = s] \right|.$$  

We simply denote by $\Delta_1(X)$ the statistical distance between $X$ and the uniform distribution on $S$:

$$\Delta_1(X) = \sum_{s \in S} \left| \Pr[X = s] - \frac{1}{|S|} \right|,$$

and say that $X$ is statistically close to uniform when $\Delta_1(X)$ is negligible.$^3$

The squared Euclidean imbalance of $X$ is the square of the $\ell_2$ distance between $X$ and the uniform distribution on the same set:

$$\Delta_2^2(X) = \sum_{s \in S} \left| \Pr[X = s] - \frac{1}{|S|} \right|^2.$$

$^2$An alternate definition frequently found in the literature differs from this one by a constant factor $1/2$. That constant factor is irrelevant for our purposes.

$^3$For this to be well-defined, we of course need a family of random variables on increasingly large sets $S$. Usual abuses of language apply.

We also define the collision probability of $X$ as:

$$\beta(X) = \sum_{s \in S} \Pr[X = s]^2,$$

and the collision entropy (also known as the Rényi entropy) of $X$ is then $H_2(X) = -\log_2 \beta(X)$. Finally, the min-entropy of $X$ is $H_{\infty}(X) = -\log_2 \gamma(X)$, where $\gamma(X) = \max_{s \in S} \Pr[X = s]$.

**Lemma A:** Suppose $X$ is a random variable of a finite set $S$. The quantities defined above satisfy the following relations:

$$\gamma(X)^2 \leq \beta(X) = 1/|S| + \Delta_2^2(X) \leq \gamma(X) \leq 1/|S| + \Delta_1(X),$$

$$\Delta_1(X) \leq \Delta_2(X) \sqrt{|S|}.$$  

B. Prime Numbers in Arithmetic Progressions

All algorithms proposed in this paper are based on the key idea that, for any given integer $q > 1$, prime numbers are essentially equidistributed among invertible classes modulo $q$. The first formalization of that idea is de la Vallée Poussin’s prime number theorem for arithmetic progressions [14], which states that for any fixed $q > 1$ and any $a$ coprime to $q$, the number $\pi(x; q, a)$ of prime numbers $p \leq x$ such that $p \equiv a \pmod{q}$ satisfies:

$$\pi(x; q, a) \sim \frac{\pi(x)}{\varphi(q)},$$

where $\pi(x)$ is the number of primes $p \leq x$, and $\varphi$ is Euler’s totient function. De la Vallée Poussin established that estimate for constant $q$, but it is believed to hold uniformly in a very large range for $q$. In fact, H.L. Montgomery conjectured [15], [16] that for any $\varepsilon > 0^4$:

$$\left| \pi(x; q, a) - \frac{\pi(x)}{\varphi(q)} \right| \ll \varepsilon (x/q)^{1/2 + \varepsilon} \quad (q < x, \ (a, q) = 1),$$

which would imply that (3) holds uniformly for $q \ll x/\log^2 x$. However, Friedlander and Granville showed [17] that conjecture to be overly optimistic, and proposed the following corrected estimate.

**Conjecture A (Friedlander–Granville–Montgomery):** For $q < x$, $(a, q) = 1$ and all $\varepsilon > 0$, we have:

$$\left| \pi(x; q, a) - \frac{\pi(x)}{\varphi(q)} \right| \ll_{\varepsilon} (x/q)^{1/2} \cdot x^\varepsilon.$$

In particular, the estimate (3) holds uniformly for $q \ll x^{1-3\varepsilon}$.

That conjecture is much more precise than what can be proved using current techniques, however. The best unconditional result of the same form is the Siegel–Walfisz theorem [18], which only implies that (3) holds in the much smaller range $q \ll (\log x)^A$ (for any $A > 0$).

Stronger estimates can be established assuming the Extended Riemann Hypothesis (i.e. the Riemann Hypothesis

$^4$As is usual in analytic number theory and related subjects, we use the notations $f(a) \ll g(a)$ and $f(a) = O(g(a))$ interchangeably. A subscripted variable on $\ll$ or $O$ means that the implied constant depends only on that variable.
FOUQUE AND TIBOUCHI: CLOSE TO UNIFORM PRIME NUMBER GENERATION WITH FEWER RANDOM BITS

A simple method to construct obviously uniformly distributed prime numbers up to $x$ is to pick random numbers in $[1, x]$ and test until a prime is found. However, this method consumes a large amount of randomness to produce a prime, which is quite large.

As mentioned in the introduction, we propose the following algorithm.

1. Fix $g$. Let $q=\phi(g)\log x$.
2. Pick $a$ such that $g \not\equiv 1 \pmod{q}$ and $g \not\equiv 1 \pmod{q}$.
3. If $g$ is prime then return $p=1$.
4. $u=1$.
5. $r\leftarrow \log_2 q$.
6. If $g$ is prime then return $p=1$.
7. pseudorandomly choose $r\bmod q$.
8. $r'\leftarrow g^{3^u+q} \pmod{q}$.
9. $u\leftarrow u+1$.
10. If $u<\frac{r'}{2}$ then go to step 4.

where the implied constant is absolute.

Finally, we will need a few classical facts regarding the distribution of primes. For example, see [19, Ths. 328 and 330].

In order to obtain the result, we will use the following asymptotic estimates.

Let $\chi$ be any non-principal Dirichlet character. Then

$$\sum_{\chi \in \chi} \frac{\phi(q)}{\varphi(q)} \ll x^{2/7} \log x.$$  

And, for any integer $a$ such that $a \not\equiv 1 \pmod{q}$,

$$\sum_{\chi \in \chi} \frac{\phi(q)}{\varphi(q)} \ll x^{1/2} \log x.$$  

Furthermore, we can also show, under the same assumption, that the output distribution is statistically close to uniform.

One can use this method to obtain unconditional results of the same type. We can avoid these problems by modifying the algorithm slightly as we will discuss in Section 3.2.2. Finally, in Section 3.2.3, we will use the Barban-Davenport-Halberstam theorem to obtain unconditional results using the same method.

Algorithm 1 (Our Basic Algorithm)

1. Fix $g$. Let $q=\phi(g)\log x$.
2. Pick $a$ such that $g \not\equiv 1 \pmod{q}$.
3. $u=1$.
4. $r\leftarrow \log_2 q$.
5. $r'\leftarrow g^{3^u+q} \pmod{q}$.
6. If $g$ is prime then return $p=1$.
7. pseudorandomly choose $r\bmod q$.
8. $r''\leftarrow g^{3^u+q} \pmod{q}$.
9. $u\leftarrow u+1$.
10. If $u<\frac{r'}{2}$ then go to step 4.

This implies (3) in the range $q \ll x^{2/7} \log x$. However, the previous results on the character sums $\pi(x)$ and more careful character sum arguments allowed Turán to obtain the following theorem.

Theorem A (Turán): The Extended Riemann Hypothesis is an effective statement about the distribution of primes. The fact that it is an effective statement about the distribution of primes has close to maximal entropy.

Moreover, we can also show, under the same assumption, that the output distribution is statistically close to uniform.

We can avoid these problems by modifying the algorithm slightly as we will discuss in Section 3.2.2. Finally, in Section 3.2.3, we will use the Barban-Davenport-Halberstam theorem to obtain unconditional results using the same method.

For an extension of the results of the previous section, we can use Algorithm 1, and the algorithm should thus consume roughly $\frac{\phi(q)}{\varphi(q)} \log x$ bits of randomness on average much less than the trivial algorithm.

Therefore, we can show that the output distribution is statistically close to uniform and the algorithm slightly as discussed in Section 3.2.2. Finally, in Section 3.2.3, we will use the Barban-Davenport-Halberstam theorem to obtain unconditional results using the same method.

For an extension of the results of the previous section, we can use Algorithm 1, and the algorithm should thus consume roughly $\frac{\phi(q)}{\varphi(q)} \log x$ bits of randomness on average much less than the trivial algorithm.

Therefore, we can show that the output distribution is statistically close to uniform and the algorithm slightly as discussed in Section 3.2.2. Finally, in Section 3.2.3, we will use the Barban-Davenport-Halberstam theorem to obtain unconditional results using the same method.

For an extension of the results of the previous section, we can use Algorithm 1, and the algorithm should thus consume roughly $\frac{\phi(q)}{\varphi(q)} \log x$ bits of randomness on average much less than the trivial algorithm.

Therefore, we can show that the output distribution is statistically close to uniform and the algorithm slightly as discussed in Section 3.2.2. Finally, in Section 3.2.3, we will use the Barban-Davenport-Halberstam theorem to obtain unconditional results using the same method.
take advantage of the known factorization of \( q \) and use a unit generation algorithm such as the one proposed by Joye and Paillier [6], which we can show requires only \( O(1) \) iterations on average.

Finally, while we will not discuss the details of the primality test of Step 6, and shall pretend that it returns exact results (as the AKS algorithm [25] would, for example), we note that it is fine (and in practice preferable) to use a probabilistic compositeness test such as Rabin [26] instead, provided that the number of rounds is set sufficiently large as to make the error probability negligible. Indeed, the output distribution of our algorithm then stays statistically close to uniform, and the number of iterations is never larger.

B. Analysis Under the Friedlander–Granville–Montgomery Conjecture

As mentioned above, it is straightforward to deduce from the Friedlander–Granville–Montgomery conjecture that Algorithm 1 terminates almost surely, and to bound its expected number of iterations and amount of consumed randomness.

Theorem 3.2.1: Assume that Conjecture A holds. Then Algorithm 1 terminates almost surely, requires \((1 + o(1))\varphi(q) / q \cdot \log x\) iterations of the main loop on average, and consumes \((\varepsilon + o(1)) \cdot \varphi(q) / \log x + (\log x)^2 / x^2\) bits of randomness on average.

Proof: Indeed, fix \( q \propto x^{1-\varepsilon} \). Conjecture A implies uniformly over \( a \in (\mathbb{Z}/q\mathbb{Z})^* \):

\[
\left| \frac{\pi(x; q, a)}{\varphi(q)} - \frac{\pi(x)}{\varphi(q)} \right| \leq \left( \frac{x}{q} \right)^{1/2} \cdot x^{\varepsilon/4} \propto x^{3\varepsilon/4},
\]

which is negligible compared to \( \pi(x) / \varphi(q) \gg x^\varepsilon / \log x \).

As a result, we get \( \pi(x; q, a) = (1 + o(1))\pi(x) / \varphi(q) = (1 + o(1))\varphi(q) / \varphi(q) \cdot x / \log x \) uniformly over \( a \), and the success probability of the main loop becomes:

\[
\frac{\pi(x; q, a)}{1 + \left( \frac{a}{q} \right)} \cdot \frac{1 + o(1)}{\log x}
\]

which implies the stated results immediately. \( \square \)

Now let \( X \) be the output distribution of Algorithm 1, i.e. the distribution on the set of prime numbers \( \leq x \) such that Algorithm 1 outputs a prime \( p \) with probability exactly \( \Pr[X = p] \). Clearly, we have, for all \((a, q) = 1\) and all \( t \) such that \( a + t \cdot q \leq x \) is prime:

\[
\Pr[X = a + t \cdot q] = \frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x; q, a)}.
\]

As a result, the squared Euclidean imbalance of \( X \) is:

\[
\Delta^2(X) = \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} \sum_{a + t \cdot q \leq x \text{ prime}} \left( \frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x; q, a)} - \frac{1}{\pi(x)} \right)^2
\]

\[
+ \sum_{\substack{a \in (\mathbb{Z}/q\mathbb{Z})^* \ \rho \mid q}} \frac{1}{\pi(x)}
\]

\[
\propto \frac{1}{\pi(x)} \cdot \frac{1}{\varphi(q)} - \frac{1}{\pi(x)} \cdot \frac{1}{\varphi(q)}
\]

\[
\propto \frac{1}{\pi(x)} \cdot \frac{1}{\varphi(q)} \cdot \frac{(\log x)^2}{x^2} \cdot x^{3\varepsilon/2}
\]

We can then deduce the following.

Theorem 3.2.2: Assume that Conjecture A holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is only negligibly smaller than that of the uniform distribution.

Proof: Indeed, by (2), the statistical distance to the uniform distribution satisfies:

\[
\Delta_1(X) \leq \Delta_2(X) = \sqrt{\pi(x)} \ll \frac{\sqrt{x}}{x^{1/2 + \varepsilon/6}} \ll x^{-\varepsilon/6},
\]

which is negligible. Moreover, the collision probability is:

\[
\beta(X) = \frac{1}{\pi(x)} + \Delta_2(X) = \frac{1}{\pi(x)} \left( 1 + O \left( \frac{\pi(x)}{x^{1 + \varepsilon/3}} \right) \right)
\]

\[
= \frac{1}{\pi(x)} \left( 1 + o(x^{-\varepsilon/3}) \right).
\]

Hence:

\[
H_2(X) = \log_2 \left( \pi(x) \right) - \log_2 \left( 1 + o(x^{-\varepsilon/3}) \right)
\]

\[
= (H_2)_{\text{max}} - o(x^{-\varepsilon/3})
\]

as required. \( \square \)

C. Analysis Under the Extended Riemann Hypothesis

Assume the Extended Riemann Hypothesis, and denote by \( a \) the fraction of all possible choices of \( a \in (\mathbb{Z}/q\mathbb{Z})^* \) such that the error term \( E(x; q, a) := \left| \pi(x; q, a) - \pi(x) / \varphi(q) \right| \) satisfies \( E(x; q, a) > x^{3\varepsilon/4} \). Then, Turán’s theorem asserts that:

\[
\sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} E(x; q, a)^2 \ll x \log x \cdot \log x \cdot \frac{x^{3\varepsilon/2}}{\varphi(q)} \cdot \frac{1}{\pi(x)}
\]

and the left-hand side is greater or equal to \( a \varphi(q) \cdot x^{3\varepsilon/2} \) by definition of \( a \). As a result, we get:

\[
\alpha \ll \left( \frac{x^{1-3\varepsilon/2} \log x}{\varphi(q)} \cdot \frac{1}{\pi(x)} \right) \cdot \frac{1}{x^{3\varepsilon/2}}
\]

and hence \( \alpha \) is negligible. Therefore, for all except at most a negligible fraction of choices of \( a \in (\mathbb{Z}/q\mathbb{Z})^* \), we obtain that \( E(x; q, a) \leq x^{3\varepsilon/4} \), and since \( \pi(x) / \varphi(q) \gg x^\varepsilon / \log x \), this implies \( \pi(x; q, a) = (1 + o(1))\pi(x) / \varphi(q) \) as before. As a result, under the ERH, we obtain an analogue of Theorem 3.2.1 valid with overwhelming probability on the choice of \( a \).

Theorem 3.3.1: Assume the ERH holds. Then Algorithm 1 terminates with overwhelming probability. Moreover, except for a negligible fraction of choices of the class \( a \) mod \( q \), it requires \((1 + o(1))\varphi(q) / q \cdot \log x\) iterations of the main loop on average, and consumes \((\varepsilon + o(1)) \cdot \varphi(q) / \log x + (\log x)^2 / x^2\) bits of randomness on average.
Moreover, using Turán’s theorem and the Cauchy–Schwarz inequality, we can also establish under the ERH the following analogue of Theorem 3.2.2, regarding the output distribution of the algorithm.

**Theorem 3.3.2:** Assume the ERH holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is no more than \(O(\log \log x)\) bits smaller than that of the uniform distribution.

**Proof:** Algorithm 1 almost surely produces an output for a given choice of \(a\) if and only if \(x/q, a \neq 0\), and this inequality, although it holds under strong conjectures on the distribution of primes in arithmetic progression, has not been proved even assuming the ERH. Therefore, the probability that the algorithm outputs a prime \(p = a + tq \leq x\) becomes:

\[
\Pr[X = a + tq] = \frac{1}{\varphi^*_x(q)} \cdot \frac{1}{\pi(x; q, a)}
\]

where \(\varphi^*_x(q) = \#\{a \in \mathbb{Z}/q\mathbb{Z}^* \mid \pi(x; q, a) \neq 0\}\). By the previous discussion on the distribution of the values \(\pi(x; q, a)\), we know that \(\varphi^*_x(q) = \varphi(q) \cdot (1 - O(\alpha))\). As a result, a similar computation as in §III-B gives:

\[
\Delta^2_2(X) = \frac{1}{\pi(x)^2} \sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \left(\frac{1}{\varphi^*_x(q)} \cdot \frac{1}{\pi(x; q, a)}\right)^2 \leq \frac{S^2}{\pi(x)^2} + \omega(q)\pi(x)^2,
\]

where \(\omega(q)\) denotes as usual the number of prime factors of \(q\). Then, using the coarse lower bound \(\pi(x; q, a) \geq 1\) when \(\pi(x; q, a) \neq 0\), we get:

\[
\Delta^2_2(X) \leq \frac{S^2 + \omega(q)}{\pi(x)^2}
\]

where:

\[
S = \sqrt{\sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \left| \frac{\pi(x; q, a) - \pi(x)}{\varphi^*_x(q)} \right|^2}
\]

\[
\leq \sqrt{\sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \left| \frac{\pi(x)}{\varphi^*_x(q)} \right|^2}
\]

\[
+ \sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \frac{\varphi(x) \cdot \varphi(x)}{\varphi(q) \cdot \varphi^*_x(q)}
\]

\[
= O(x^{1/2}\log x) + \frac{\varphi(x)\pi(x)^2}{\varphi(q) \cdot \varphi^*_x(q)} \leq x^{1/2}\log x + \frac{x\log x}{\sqrt{x^{1/2} - \epsilon/2}} \cdot \frac{\alpha}{\phi(q)}\pi(x)^2
\]

\[
\ll x^{1/2}\log x + \frac{x\log x (\log x)^{3/2}}{x^{1/2} - \epsilon/2 \cdot \sqrt{x^{1/2}}}
\]

\[
\ll x^{1/2}\log x (\log x)^2
\]

by Turán’s theorem again. Hence:

\[
\Delta^2_2(X) \ll \frac{x^{2+\epsilon} \log x}{\pi(x)^2} \ll \frac{\log^{3+\epsilon} x}{\pi(x)}.
\]

This is enough to obtain a relatively good bound on the collision entropy:

\[
H_2(X) = \log_2(\pi(x)) - \log_2(\log^{3+\epsilon} x)
\]

\[
= (H_2)_{\max} - O(\log \log x)
\]

but isn’t sufficient for bounding the statistical distance. However, a direct computation using Turán’s theorem and the Cauchy–Schwarz inequality is enough:

\[
\Delta_1(X) = \left| \sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \frac{\pi(x; q, a)}{\varphi^*_x(q)} - \frac{\pi(x)}{\pi(x)} \right|
\]

\[
\ll \frac{1}{\pi(x)} \cdot \sum_{a \in \mathbb{Z}/q\mathbb{Z}^*} \left| \frac{\pi(x; q, a) - \pi(x)}{\varphi^*_x(q)} + \omega(q)\pi(x)^2 \right|
\]

\[
\ll \frac{\log x}{x^{1/2}} \cdot x^{1/2} \log x \cdot \sqrt{q}
\]

which proves that the distribution is indeed statistically close to uniform.

\[\square\]

### D. Achieving Almost Sure Termination Under the ERH

Theorem 3.3.1 above is somewhat unsatisfactory, as we have to ignore a negligible but possibly nonzero fraction of all values \(a \mod q\) to obtain a bound on the average number of iterations and on the randomness consumed by Algorithm 1 under the ERH. But this is unavoidable for that algorithm: as mentioned above, it is not known whether the ERH implies that for \(q \propto x^{1-\epsilon}\), all \(a \in \mathbb{Z}/q\mathbb{Z}^*\) satisfy \(\pi(x; q, a) \neq 0\). And if an \(a\) exists such that \(\pi(x; q, a) = 0\), the choice of that \(a\) in Step 2 of Algorithm 1, however unlikely, is a case of non-termination: as a result, the existence of such an \(a\) prevents any nontrivial bound on average running time or average randomness.

We propose to circumvent that problem by falling back to the trivial algorithm (pick a random \(p < x\), check whether it is prime and try again if not) in case too many iterations of the main loop have been completed. This variant is presented as Algorithm 2.

Clearly, since Algorithm 2 is the same as Algorithm 1 except for the possible fallback to the trivial algorithm, which has a perfectly uniform output distribution, the output distribution of the variant is at least as close to uniform as the original algorithm. In other words, the analogue of Theorem 3.3.2 holds, with the same proof.

**Theorem 3.4.1:** Assume the ERH holds. Then the output distribution of Algorithm 2 is statistically close to uniform, and its collision entropy is no more than \(O(\log \log x)\) bits smaller than that of the uniform distribution.

Moreover, as claimed above, we can obtain the following stronger analogue of Theorem 3.3.1.

**Theorem 3.4.2:** Assume the ERH holds. Then Algorithm 2 terminates almost surely, requires \((1 + o(1))\varphi(q)/q \cdot \log x\)
Algorithm 2 A Variant Which Terminates Almost Surely
Under the ERH

1: Fix $q \propto x^{1-\varepsilon}$
2: $a \leftarrow (\mathbb{Z}/q\mathbb{Z})^*$, seen as an element of $\{1, \ldots, q-1\}$
3: repeat $T = \log_2 x$ times
4: $t \leftarrow \{0, \ldots, \lfloor \frac{x-a}{q} \rfloor \}$
5: $p \leftarrow a + t \cdot q$
6: if $p$ is prime then return $p$
7: end repeat
8: repeat forever
9: $p \leftarrow \{1, \ldots, \lfloor x \rfloor \}$
10: if $p$ is prime then return $p$
11: end repeat

iterations of the main loop on average, and consumes $(\varepsilon + o(1)) \cdot \frac{\varphi(q)}{q} \cdot (\log x)^2$ bits of randomness on average.

Proof: Algorithm 2 terminates almost surely because the trivial algorithm does. One can estimate its average number of iterations as follows. Denote by $\omega(t)$ the probability that Algorithm 2 terminates after exactly $t$ iterations, and $\sigma_a(t)$ the probability of the same event conditionally to $a$ being chosen in Step 2. We have:

$$
\sigma_a(t) = \begin{cases} 
(1 - \frac{\pi(x; q, a)}{\lfloor x \rfloor})^{t-1} \cdot \frac{\pi(x; q, a)}{\lfloor x \rfloor} & (t \leq T) \\
(1 - \frac{\pi(x; q, a)}{\lfloor x \rfloor})^T (1 - \frac{\pi(x; q, a)}{\lfloor x \rfloor})^{T-1} \cdot \frac{\pi(x)}{\lfloor x \rfloor} & (t > T)
\end{cases}
$$

$$
\omega(t) = \frac{1}{\varphi(q)} \sum_a \sigma_a(t).
$$

Moreover, the expected number $N_a$ of iterations in Algorithm 2 is given by $N = \sum_{t \geq 1} t \omega(t)$. We can denote by $N_a = \sum_{t \geq 1} t \sigma_a(t)$ the contribution of a certain choice $a \in (\mathbb{Z}/q\mathbb{Z})^*$. Now, recall from §III-C that $\pi(x; q, a)$ is within a distance $\ll x^{3\varepsilon/4} \log x$ of $\pi(x)/\varphi(q)$, except for a fraction $\ll (\log x)^{3\varepsilon/2}$ of all possible choices of $a$. If we denote by $\mathcal{A}$ the set of “bad” choices of $a$, we can write, for all $a \in \mathcal{A}$:

$$
\omega_a(t) \leq \begin{cases} 
1 & (t \leq T) \\
(1 - \frac{\pi(x; q, a)}{\lfloor x \rfloor})^{T-1} \cdot \frac{\pi(x)}{\lfloor x \rfloor} & (t > T)
\end{cases}
$$

Hence, if we let $\xi := \pi(x)/\lfloor x \rfloor$, we get:

$$
N_a \leq \sum_{t = 1}^T t + \sum_{t = T+1}^{+\infty} t(1-\xi)^{t-1} \xi
= T(T+1)/2 + \sum_{k = 1}^{+\infty} (T+k)(1-\xi)^{k-1} \xi
= T(T+1)/2 + \sum_{k = 1}^{+\infty} (T+k)(1-\xi)^{k-1} \xi
\leq T(T+3)/2 + \frac{1}{\xi} \ll \log^4 x.
$$

On the other hand, for $a \notin \mathcal{A}$, we have $\xi_a := \pi(x; q, a) = \frac{q}{\varphi(q)} \cdot \frac{1+o(1)}{\log x}$. Therefore:

$$
N_a = \sum_{t = 1}^T \frac{T}{1-\xi_a} + (1-\xi_a)^T \sum_{t = T+1}^{+\infty} (1-\xi_a)^{T-t-1} \xi_a
= \frac{1}{\xi_a} - \sum_{t = T+1}^{+\infty} (1-\xi_a)^{T-t-1} \xi_a
\leq \frac{T}{1-\xi_a} + (1-\xi_a)^T \sum_{t = T+1}^{+\infty} (T+k)(1-\xi_a)^{k-1} \xi_a
\leq \frac{T}{1-\xi_a} + \exp(-T(1-\xi_a)) \cdot (2T + 1/\xi + 1/\xi_a)
\leq \frac{T}{1-\xi_a} + \exp \left( (1 + o(1)) \frac{q}{\varphi(q)} \log x \right)
\ll \frac{1}{x^{1-\varepsilon}}.
$$

As a result, we obtain:

$$
N = \frac{1}{\varphi(q)} \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} N_a
= \left(1 - O(\log^3 x/\xi^{\varepsilon/2}) \cdot \frac{1}{\xi_a} + O(1/x^{1-\varepsilon}) \right)
+ O(\log^3 x/\xi^{\varepsilon/2}) \cdot O(\log^4 x)
= \frac{1}{\xi_a} + O(\log^7 x)
= (1 + o(1)) \frac{\varphi(q)}{q} \cdot \log x
$$

as required. For the expected number $R$ of random bits consumed by the algorithm, it is given (ignoring the negligible amount necessary to pick $a$) by:

$$
R = \frac{\log x}{\log 2} \left( \sum_{t = 1}^T t \cdot \omega(t) + \sum_{t = T+1}^{+\infty} (eT + t - T) \cdot \omega(t) \right)
$$

and the stated estimate is obtained by an exactly analogous computation.

$\square$

E. An Unconditional Algorithm

Finally, we propose yet another variant of our algorithm for which both almost sure termination and uniformity bounds can be established unconditionally. The idea is to no longer use a fixed modulus $q$, but to pick it uniformly at random instead in the range $[1, \ldots, Q]$ where $Q \propto x \log x$ for some constant $A > 0$. Uniformity bounds can then be deduced from the Barban–Davenport–Halberstam theorem. Unfortunately, since $Q$ is only polynomially smaller than $x$, we can no longer prove
Algorithm 3 An Unconditional Variant

1: Fix $Q \propto x(\log x)^{-A}$ even
2: $q \leftarrow \{Q/2 + 1, \ldots, Q\}$
3: $a \leftarrow \{0, \ldots, Q - 1\}$
4: if $a \geq q$ or gcd$(a, q) \neq 1$ then goto step 2
5: repeat $T = \log^2 x$ times
6: \hspace{1em} $t \leftarrow \{0, \ldots, \left\lfloor \frac{x-a}{q}\right\rfloor\}$
7: \hspace{1em} $p \leftarrow a + t \cdot q$
8: \hspace{1em} if $p$ is prime then return $p$
9: end repeat
10: repeat forever
11: \hspace{1em} $p \leftarrow \{1, \ldots, |x|\}$
12: \hspace{1em} if $p$ is prime then return $p$
13: end repeat

This allows us to deduce the analogue of Theorem 3.4.2 for Algorithm 3.

Theorem 3.5.1: Algorithm 3 with $A > 4$ terminates almost surely, requires at most $(1 + o(1)) \cdot \log x$ iterations of the main loop on average, and consumes at most:

$$(A + o(1)) \cdot \frac{\log x \log \log x}{\log 2}$$

bits of randomness on average.

Proof: Algorithm 3 terminates almost surely because the trivial algorithm does. One can estimate its average number of iterations as follows. Denote by $\sigma(t)$ the probability that Algorithm 3 terminates after exactly $t$ iterations, and $\sigma_{q,a}(t)$ the probability of the same event conditionally to the pair $(q, a)$ being chosen in Steps 2–4. We have:

$$\sigma_{q,a}(t) = \left\{ \begin{array}{ll}
(1 - \pi(x/q,a))^{t-1} & \pi(x/q,a) \\
(1 - \pi(x/q,a))^{t-1} & \pi(x/q,a)
\end{array} \right. \frac{\pi(x/q,a)}{|x|}$$

for $t \leq T$

$$\sigma(t) = \frac{1}{F(Q)} \sum_{Q/2 < q \leq Q} \sum_{a \in \mathbb{Z}/q\mathbb{Z}} \sigma_{q,a}(t).$$

Moreover, the expected number $N$ of iterations in Algorithm 2 is given by $N = \sum_{t \geq 1} t \sigma(t).$ We can denote by $N_{q,a} = \sum_{t \geq 1} t \sigma_{q,a}(t)$ the contribution of a certain choice $(q, a).$

As we have just seen, $\pi(x; q, a) = (1 + o(1)) \pi(x)/\phi(q),$ except perhaps for a fraction $\alpha \ll (\log x)^{-A/2}$ of all choices of $(q, a).$ If we denote by $\mathcal{A}$ the set of “bad” choices of $(q, a),$ we can write, as before, that for all $(q, a) \in \mathcal{A}:$

$$\sigma_{q,a}(t) \leq \left\{ \begin{array}{ll}
1 & t \leq T;
\left(1 - \frac{\pi(x)}{|x|}\right)^{t-T-1} & \pi(x)/|x|
\end{array} \right.$$

otherwise.

Hence, if we let $\xi = \pi(x)/|x|,$ we again obtain:

$$N_{q,a} \leq \sum_{t=1}^{T} t + \sum_{t=T+1}^{+\infty} t(1 - \xi)^{t-T-1}$$

$$\leq T(T + 3)/2 + \frac{1}{\xi} \ll \log^4 x.$$
Algorithm 4 A Variant of Algorithm 3 With no Fallback

1: Fix $Q \propto x(\log x)^{-\delta}$ even
2: $a \leftarrow \{Q/2 + 1, \ldots, Q\}$
3: $a \leftarrow \{0, \ldots, Q - 1\}$
4: if $a \geq q$ or $\gcd(a, q) \neq 1$ then goto step 2
5: repeat forever
6: $t \leftarrow \{0, \ldots, \lfloor x/a \rfloor\}$
7: $p \leftarrow a + t \cdot q$
8: if $p$ is prime then return $p$
9: end repeat

As a result, we get:

$$N = \frac{1}{F(x)} \sum_{Q/2 < q \leq Q} \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} N_{q, a}$$

$$\leq (1 - \alpha) \cdot (1 + o(1)) \cdot \log x + a \cdot O(\log^4 x)$$

$$= (1 + o(1)) \cdot \log x + O((\log x)^{-A/2 - 1}) \cdot O(\log^4 x)$$

$$= (1 + o(1)) \cdot \log x + O((\log x)^{3 - A/2})$$

$$= (1 + o(1)) \cdot \log x$$

as required, since $3 - A/2 < 1$. As for the expected number $R$ of random bits consumed by the algorithm, it is now given by:

$$R = A \log \log x \sum_{t=1}^{T} t \cdot \omega(t)$$

$$+ \sum_{t=T+1}^{\infty} \left(T \cdot \frac{A \log \log x}{\log 2} + (t - T) \cdot \frac{\log x}{\log 2}\right) \cdot \omega(t)$$

where we have again ignored the random bits necessary to pick the pair $(q, a)$, since we need only $Q^2/Q \sim 2^{2/9}$ iterations of the loop from Step 2 to Step 4 to select it, and hence $O(\log x)$ random bits. The stated bound is obtained by essentially the same computation as for $N$. \square

Remark 1: As noted at the beginning of this section, the number of random bits used by Algorithm 3 is asymptotically always smaller than for Algorithm 1 and 2, even if they use a primorial $q$. Indeed, the latter is given by:

$$(\varepsilon + o(1)) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2} \gg (\log x)^2 \cdot \log \log x$$

which is always asymptotically greater than $O(\log x \log \log x)$. We now turn to estimates on the uniformity of the output distribution of the algorithm. For that purpose, we consider instead the output distribution $X$ of Algorithm 4, the variant of Algorithm 3 in which no bound is set to the number of iterations of the main loop (Steps 5–9), i.e. with no fallback to the trivial algorithm. Clearly, since the trivial algorithm has a perfectly uniform output distribution, the output distribution of Algorithm 3 is at least as close to uniform as $X$.

Theorem 3.5.2: The output distribution of Algorithm 4 (and hence also Algorithm 3) has a statistical distance $\Delta_1 \ll (\log x)^{(1 - A)/2}$ to the uniform distribution. In particular, this distance is polynomially small as long as $A > 1$.

Proof: Algorithm 4 produces an output (almost surely) for exactly those choices of $(q, a)$ such that $\pi(x; q, a) \neq 0$.

Let us denote by $F_x^*(Q)$ the number of such choices. Clearly, with notation from above, we have:

$$F_x^*(Q) = \sum_{Q/2 < q \leq Q} f_x^*(q) \quad \text{and} \quad 1 - a \leq \frac{F_x^*(Q)}{F(Q)} \leq 1.$$ 

Then, the probability that Algorithm 4 outputs a given prime $p \leq x$ can be written as:

$$\Pr[X = p] = \frac{1}{F_x^*(Q)} \sum_{p \mid q \leq Q} \frac{1}{\pi(x; q, p)}.$$ 

Therefore, we have:

$$\Delta_1(X) = \sum_{p \leq x} \left| \Pr[X = p] - \frac{1}{\pi(x)} \right|$$

$$= \sum_{p \leq x} \left| \frac{1}{F_x^*(Q)} \sum_{Q/2 < q \leq Q} \frac{1}{\pi(x; q, p)} - \frac{1}{\pi(x)} \right|$$

$$= \sum_{p \leq x} \frac{1}{F_x^*(Q)} \sum_{Q/2 < q \leq Q} \frac{1}{\pi(x; q, p)} - \sum_{Q/2 < q \leq Q} \frac{\phi_x^*(q)}{\pi(x)}$$

$$\leq \frac{1}{F_x^*(Q)} \sum_{Q/2 < q \leq Q} \sum_{p \mid q \leq Q} \frac{1}{\pi(x; q, p)} - \frac{\phi_x^*(q)}{\pi(x)}$$

$$+ \frac{1}{F_x^*(Q)} \sum_{Q/2 < q \leq Q} \sum_{p \mid q \leq Q} \frac{\phi_x^*(q)}{\pi(x)}$$

$$\leq \frac{1}{F_x^*(Q)} \sum_{Q/2 < q \leq Q} \sum_{p \mid q \leq Q} \left(\pi(x) - \phi_x^*(q) \pi(x; q, a)\right)$$

$$+ \frac{O(\log Q)}{\pi(x)}.$$ 

We can then bound the sum over $(q, a)$ of $\left| \pi(x) - \phi_x^*(q) \pi(x; q, a)\right|$ as $D + D^*$, where:

$$D = \sum_{Q/2 < q \leq Q} \left| \pi(x) - \phi_x^*(q) \pi(x; q, a)\right|,$$

$$D^* = \sum_{Q/2 < q \leq Q} \left| \phi_x^*(q) \cdot \pi(x; q, a)\right|.$$

Now, on the one hand:

$$D^* = \sum_{Q/2 < q \leq Q} \left| \phi_x^*(q) \right| \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} \pi(x; q, a)$$

$$\leq (F(Q) - F_x^*(Q)) \cdot \pi(x)$$

$$\leq a F(Q) \pi(x) \ll x^3 \left(\log x\right)^{-5A/2 - 2}.$$
and on the other hand, applying the Cauchy–Schwarz inequality and the Barban–Davenport–Halberstam theorem:

\[
D = \sum_{Q/2 \leq q \leq Q} \varphi(q) \cdot |\pi(x; q, a) - \frac{x}{\varphi(q)}| \\
\leq Q \sum_{Q/2 \leq q \leq Q} |\pi(x; q, a) - \frac{x}{\varphi(q)}| \\
\leq Q \cdot \sqrt{\frac{F(Q)}{\log Q}} \sqrt{\frac{xQ}{\log Q}} \\
\ll x^3 (\log x)^{-\frac{5A}{2} - 1/2}.
\]

As a result, we obtain:

\[
\Delta_1(X) \ll \frac{1}{F^6(Q)\pi(x)} \cdot (D + D^*) \\
\ll \frac{(\log x)^{2A+1}}{x^3} \cdot \frac{x^3}{(\log x)^{5A/2+1/2}} \\
\ll \frac{1}{(\log x)^{(A-1)/2}}.
\]

IV. COMPARISON WITH OTHER PRIME NUMBER GENERATION ALGORITHMS

In this section, we compare other prime number generation algorithms to our method. In §IV-A we show that the output distribution of Brandt and Damgård’s PRIMEINC algorithm exhibits significant biases (a fact that is intuitively clear, but which we make quite precise), and in §IV-B, we discuss the advantages and drawbacks of our method compared to that of Maurer, as the only previous work emphasizing a close to uniform output distribution.

A. Primeinc

Previous works on the generation of prime numbers, such as [4] and [6], provide a proof (based on rather strong assumptions) that the output distribution of their algorithm has an entropy not much smaller than the entropy of the uniform distribution. This is a reasonable measure of the inability of an adversary to guess which particular prime was output by the algorithm, but it doesn’t rule out the possibility of gaining some information about the generated primes. In particular, it doesn’t rule out the existence of an efficient distinguisher between the output distribution and the uniform one.

Consider for example the PRIMEINC algorithm studied by Brandt and Damgård [4]. In essence, it consists in picking a random integer \( y < x \) and returning the smallest prime greater or equal to \( y \). There are some slight technical differences between this description and the actual PRIMEINC,\(^5\) but they have essentially no bearing on the following discussion, so we can safely ignore them.

It is natural to suspect that the distribution of the output of this algorithm is quite different from the uniform distribution:

\(^5\)To wit, Brandt and Damgård restrict their attention to odd numbers \( x/2 < y < x \), and set an upper bound to the number of iterations in the algorithm for generating the second prime of a twin prime pair, i.e. a prime \( p \) such that \( p - 2 \) is also prime, is abnormally unlikely. More precisely, if one believes the twin prime conjecture, the proportion of primes of that form among all primes up to \( x \) should be \( \sim 2c_2/\log x \), where \( c_2 \approx 0.66 \) is the twin prime constant. On the other hand, PRIMEINC outputs such a \( p \) if and only if it initially picks \( y = p \) or \( p - 1 \).

Therefore, we expect the frequency of such primes \( p \) in the output of PRIMEINC to be much smaller, about \( 4c_2/(\log x)^2 \). This provides an efficient distinguisher between the output of PRIMEINC and the uniform distribution.

More generally, it is easy to see that the method used in [4] to obtain the lower bound on the entropy of the output distribution of PRIMEINC can also provide a relatively large constant lower bound on the statistical distance to the uniform distribution. Indeed, the method relies on the following result, obtained by a technique first proposed by Gallagher [27].

Lemma C [4, Lemma 5]: Assume the prime \( r \)-tuple conjecture, and let \( F_r(x) \) denote the number of primes \( p \leq x \) such that the largest prime \( q \) less than \( p \) satisfies \( p - q \leq h \). Then for any constant \( \lambda \),

\[
F_{2\log x}(x) = \frac{x}{\log x}(1 - \frac{1}{e^2})(1 + o(1))
\]

as \( x \rightarrow +\infty \).

Now, the probability that the PRIMEINC algorithm outputs a fixed \( p \) can clearly be written as \( d(p)/x \), where \( d(p) \) is the distance between \( p \) and the prime that immediately precedes it. Let \( \Delta'_1 \) be the statistical distance between the output distribution of PRIMEINC and the uniform distribution. We have:

\[
\Delta'_1 = \sum_{p \leq x} \left| \frac{d(p)}{x} - \frac{1}{\pi(x)} \right| > \sum_{p \leq x, d(p) > \log x} \left| \frac{2\log x}{x} - \frac{\log x}{x} \right|
\]

for \( x \geq 17 \) in view of the already mentioned classical bound \( \pi(x) = x/\log x \) for \( x \geq 17 \). By Lemma C, this gives:

\[
\Delta'_1 > \frac{\log x}{x} F_{2\log x}(x) = \left(1 - \frac{1}{e^2}\right)(1 + o(1)) > 0.86 + o(1)
\]

as \( x \rightarrow +\infty \), and in particular, \( \Delta'_1 \) admits a relatively large constant lower bound (at least if one believes the prime \( r \)-tuple conjecture on which the entropy bound is based).

Since the entropy lower bounds for other prime generation algorithms like [6] are based on the same techniques, it is very likely that their output distributions can similarly be shown to be quite far from uniform.

B. Maurer’s Algorithm

Maurer [7], [8] proposed a prime generation algorithm based on a similar principle as Bach’s technique to produce random numbers with known factorization [28] (a simpler variant of which was later proposed by Kalai [29]). This algorithm has the major advantage of producing a primality certificate as part of its output; as a result, it generates only provable primes, contrary to our method.

Additionally, in the standard version of the algorithm, the distribution of generated primes is heuristically close to uniform. More precisely, this distribution would be exactly uniform if the following assertions held:
1) The distribution of the relative sizes of the prime factors of an integer \( x \) of given length is conditional to \( 2x + 1 \) being prime is the same as the distribution of the relative sizes of the prime factors of a uniformly random integer of the same length.

2) That distribution is, moreover, the same as the asymptotic one (i.e. as the length goes to \( +\infty \)), as computed using Dickson’s \( p \) function.

Assertion 1 is a rather nonstandard statement about prime numbers, but Maurer points to some heuristic arguments that renders it quite plausible [30], at least asymptotically. Assertion 2 is of course not exactly true, and it seems difficult to quantify how much this affects the output distribution, but probably not to a considerable extent.

That standard version of the algorithm, however, has a small probability of not terminating, as discussed in [8, Sec. §3.3]. To circumvent that problem, Maurer suggests a modification to the algorithm which introduces biases in the output distribution. If it is used, the algorithm will in particular never generate primes \( p \) such that \( (p − 1)/2 \) has a large prime factor. This provides an efficient distinguisher from the uniform distribution (exactly analogous to the “twin prime” distinguisher of the previous section), since e.g. safe primes have non negligible density. There are other ways to avoid cases of non-termination that do not affect the output distribution to such an extent (such as some form of backtracking), but Maurer advises against them because of the large performance penalty they may incur.

Furthermore, the paper [8] also describes faster variants of the algorithm that are less concerned with the quality of the output distribution, and they do indeed have outputs that are very far from uniform, typically reaching only around 10% of all prime numbers.

More importantly, the main drawback of Maurer’s algorithm compared to our method is likely the amount of randomness it consumes: it is within a small constant factor of the amount consumed by the trivial algorithm. In fact, if we neglect everything but the last loop of the topmost step in the recursion, and count as zero the cases when the recursion has more than two steps, we see that the average amount of randomness used by the algorithm is bounded below by \( c \) times the corresponding amount for the trivial algorithm, where (with the notations of [8, Appendix 1]):

\[
c \geq \int_{1/2}^{1} (1-x) \, dF_1(x) = \int_{1/2}^{1} (1-x) \, \frac{dx}{x} = \log 2 - \frac{1}{2} \geq 0.19
\]

and we only counted a very small fraction of the actual randomness used!

As a result, in context where random bits are scarce and primality certificates are not required, it seems that our method offers a rather better trade-off than Maurer’s algorithm: the uniformity of the distribution is easier to estimate in our case, and we use much less randomness. Moreover, Maurer finds [8, Sec. §4.3] that an efficient implementation of his algorithm is about 40% slower than the trivial algorithm, whereas Algorithm 1 is easily 5 times as fast in practice.

On the other hand, if proofs of primality are considered important, it is clearly better to rely on Maurer’s algorithm than to use our method and then primality proving.

V. CONCLUSION AND FUTURE WORK

In this paper, we have described several variants of a prime generation algorithm which achieves a close to uniform distribution while using a comparatively small number of random bits, and we have analyzed and proved it under increasingly less expensive assumptions (first the Friedlander–Granville–Montgomery conjecture, then the ERH, and finally fully unconditionally). However, our analysis is purely asymptotic. For practical applications to cryptography and security, it would be desirable to make the statements of our theorems effective.

Regarding the analysis of §III-B based on the Friedlander–Granville–Montgomery conjecture, this seems difficult to achieve, since the conjecture itself involves a non-explicit constant depending on the exponent \( c \). Therefore, one would need to formulate a stronger variant of the conjecture with explicit constants, presumably backed by concrete numerical experiments, to obtain effective variants of theorems 3.2.1 and 3.2.2.

On the other hand, the analysis of §III-C and §III-D based on the ERH could be made fully effective given a version of Turán’s theorem with explicit constants. This seems more tractable: one could obtain such a version based on explicit bounds on the numbers of primes less than \( x \) in arithmetic progressions, and similar results have been established in the literature [31].

Finally, the difficulty of making the analysis of §III-E based on the Barban–Davenport–Halberstam theorem effective is probably intermediate between the previous two: in principle, this can be done by making every step in the proof of Barban and Davenport–Halberstam explicit. This would be a quite significant endeavour in analytic number theory.

In addition, we mention that this paper has focused on the problem of generating prime numbers distributed close to uniformly among all prime numbers below a certain bound. Some cryptographic constructions use primes with additional properties, such as primes satisfying specific congruence relations, or so-called safe primes (i.e. primes \( p \) such that \( (p − 1)/2 \) is also prime, and hence a Sophie Germain prime). Extending our results to support generating primes satisfying congruence relations modulo small primes (i.e. generating close to uniform primes up to some bound in specific arithmetic progressions) is relatively straightforward. In contrast, generating primes distributed close to uniformly among safe primes in a provable way would at the very least require strong results on the distribution of safe primes, which is much less well-studied, as even the existence of infinitely many safe primes is an open problem.

REFERENCES

[1] P.-A. Fouque and M. Tibouchi, “Close to uniform prime number generation with fewer random bits,” in International Colloquium on Automata, Languages, and Programming (Lecture Notes in Computer Science), vol. 8572, J. Esparza, P. Fraigniaud, T. Hufedelt, and E. Koutsoupias, Eds. Berlin, Germany: Springer, 2014, pp. 991–1002.

[2] D. Eastlake, III, J. Schiller, and S. Crocker, “Randomness requirements for security,” document RFC 4086, (Best Current Practice), Internet Engineering Task Force, Jun. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4086.txt
[3] J. Brandt, I. Damgård, and P. Landrock, “Speeding up prime number generation,” in Proc. Int. Conf. Theory Appl. Cryptol., in Lecture Notes in Computer Science, vol. 739, H. Imai, R. L. Rivest, and T. Matsumoto, Eds. Berlin, Germany: Springer, 1991, pp. 440–449.

[4] J. Brandt and I. Damgård, “On generation of probable primes by incremental search,” in Proc. Ann. Int. Cryptol. Conf., in Lecture Notes in Computer Science, vol. 740, E. F. Brickell, Ed. Berlin, Germany: Springer, 1992, pp. 358–370.

[5] M. Joye, P. Paillier, and S. Vaudenay, “Efficient generation of prime numbers,” in Proc. Int. Workshop Cryptograph. Hards. Embedded Syst., in Lecture Notes in Computer Science, vol. 1965, Ç. K Koç and C. Paar, Eds. Berlin, Germany: Springer, 2000, pp. 340–354.

[6] M. Joye and P. Paillier, “Fast generation of prime numbers on portable devices: An update,” in Proc. Int. Workshop Cryptograph. Hards. Embedded Syst., in Lecture Notes in Computer Science, vol. 4249, L. Goubin and M. Matsui, Eds. Berlin, Germany: Springer, 2006, pp. 160–173.

[7] U. M. Maurer, “Fast generation of secure RSA-moduli with almost maximal diversity,” in Proc. Workshop Theory Appl. Cryptograph. Techn., J. Quisquater and J. Vandewalle, Eds., 1989, pp. 636–647.

[8] U. M. Maurer, “Fast generation of prime numbers and secure public-key cryptographic parameters,” J. Cryptol., vol. 8, no. 3, pp. 123–155, 1995.

[9] P. Mihăilescu, “Fast generation of provable primes using search in arithmetic progressions,” in Proc. Ann. Int. Cryptol. Conf., in Lecture Notes in Computer Science, vol. 839, J. Quisquater and J. Vandewalle, Eds., 1994, pp. 282–293.

[10] G. H. Hardy and J. E. Littlewood, “Some problems of ‘partitio numerorum’ : III. On the expression of a number as a sum of primes,” Acta Math., vol. 44, no. 1, pp. 1–70, 1923.

[11] R. Cramer and V. Shoup, “Signature schemes based on the strong RSA assumption,” ACM Trans. Inf. Syst. Secur., vol. 3, no. 3, pp. 161–185, 2000.

[12] P. Mihăilescu, “Security of biased sources for cryptographic keys,” in Cryptography and Computational Number Theory (Progress in Computer Science and Applied Logic), vol. 20. Basel, Switzerland: Birkhäuser, 2001, pp. 287–302.

[13] V. Shoup, A Computational Introduction to Number Theory and Algebra. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[14] C. J. de La Vallée Poussin, “Recherches analytiques sur la théorie des nombres premiers,” Ann. Soc. Sci. Bruxelles, vol. 2, pp. 281–362, 1896.

[15] H. L. Montgomery, Topics in Multiplicative Number Theory (Lecture Notes in Mathematics), vol. 227. Berlin, Germany: Springer, 1971.

[16] H. L. Montgomery, “Problems concerning prime numbers,” in Proc. Symp. Pure Math., vol. 28, 1976, pp. 307–310

[17] J. B. Friedlander and A. Granville, “Limitations to the equi-distribution of primes I,” Ann. Math., vol. 129, no. 2, pp. 363–382, 1989.

[18] A. Walfisz, “Zur additiven Zahlentheorie. II,” Mathematische Zeitschrift, vol. 40, no. 1, pp. 592–607, 1936.

[19] H. Davenport, Multiplicative Number Theory (Graduate Texts in Mathematics), vol. 74, 2nd ed. Berlin, Germany: Springer, 1980.

[20] P. Turán, “Über die Primzahlen der arithmetischen Progression,” Acta Sci. Math. Szeged, vol. 8, no. 4, pp. 226–235, 1936.

[21] M. B. Barban, “The ‘large sieve’ method and its applications in the theory of numbers,” Russian Math. Surv., vol. 21, no. 1, p. 49, 1966.

[22] H. Davenport and H. Halberstam, “Primes in arithmetic progressions,” Michigan Math. J., vol. 13, no. 4, pp. 485–489, 1966.

[23] P. X. Gallagher, “The large sieve,” Mathematika, vol. 14, no. 1, pp. 14–20, 1967.

[24] G. H. Hardy and E. M. Wright, An Introduction to the Theory of Numbers, 4th ed. Oxford, U.K.: Clarendon, 1960.

[25] M. Agrawal, N. Kayal, and N. Saxena, “PRIMES is in P,” Ann. Math., vol. 160, no. 2, pp. 781–793, 2004.

[26] M. O. Rabin, “Probabilistic algorithm for testing primality,” J. Number Theory, vol. 12, no. 1, pp. 128–138, 1980.

[27] P. X. Gallagher, “On the distribution of primes in short intervals,” Mathematika, vol. 23, no. 1, pp. 4–9, 1976.

[28] E. Bach, “How to generate factored random numbers,” SIAM J. Comput., vol. 17, no. 2, pp. 179–193, 1988.

[29] A. Kalai, “Generating random factored numbers, easily,” J. Cryptol., vol. 16, no. 4, pp. 287–289, 2003.

[30] U. M. Maurer, “Some number-theoretic conjectures and their relation to the generation of cryptographic primes,” in Cryptography and Coding, vol. 129, no. 2, pp. 363–382, 1999.

[31] P. Desaut, “Estimates of $\theta(x; k, l)$ for large values of $x$,” Math. Comp., vol. 71, no. 239, pp. 1137–1168, 2002.

Pierre-Alain Fouque, biography not available at the time of publication.

Mehdi Tibouchi, biography not available at the time of publication.