1 Introduction

2 Preliminaries

2.1 Notation

Suppose $\mathcal{D}$ is a distribution over the set S, by $x \leftarrow \mathcal{D}$ we mean that x is chosen from the set S according to the distribution $\mathcal{D}$. When x is chosen uniformly we simply denote it as $x \leftarrow S$.

We describe the asymptotic behavior of functions by several notations. The *big O* notation $\mathcal{O}(\cdot)$ is used to bound the growth of a function in terms of another function. For example, we write $f(x) = \mathcal{O}(g(x))$ if there exists a constant $C > 0$ such that for sufficiently large x, $|f(x)| \leq Cg(x)$. We denote $f \sim g$ when f and g are asymptotically the same, that is

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1. \tag{1}$$

Throughout this report we donte base 2 logarithm by lg and natural logarithm by ln.

2.2 Prime Number Theorem

Theorem 1 ([3, Chapter 4]). *Let $\pi(x)$ denote the number of primes less than or equal to $x \geq 1$. The prime number theorem states that*

$$\lim_{x\to\infty} \frac{\pi(x)\ln x}{x} = 1 \tag{2}$$

That is, $\pi(x) \sim \dfrac{x}{\ln x}$. This implies that n-th prime p_n is asymptotically given by $p_n \sim n \ln n$.

Moreover, we may frequently use these exact bounds on $\pi(x)$.

Theorem 2 ([3, Theorem 4.6,4.7]). *For any $n \geq 2$,*

$$\frac{1}{6}\frac{n}{\ln n} \leq \pi(n) \leq 6\frac{n}{\ln n} \tag{3}$$

and for any $n \geq 1$,

$$\frac{1}{6} n \ln n \leq p_n \leq 12(n \ln n + n \ln \frac{12}{e}) \tag{4}$$

Theorem 3 ([9]). *Let $\omega(n)$ denote the number of distinct prime factors of n. For all $\epsilon > 0$ and sufficiently large values of n*

$$\omega(n) \leq \frac{\ln n}{\ln \ln n} + \epsilon \tag{5}$$

3 Prime Number Generation

3.1 Uniform prime generation

The original 3-step algorithm [2] generates two random prime numbers that are distributed uniformly in a given interval. Ahlswede *el al* use these prime numbers to encrypt the message which is represented by a number. This encryption achieves the channel capacity in the case of noiseless binary symmetric channel. Moreover, it is shown that the second kind error probability of this scheme tends to zero asymptotically.

To generate uniform primes on the interval $]1, p_K]$, Ahlswede *et al* pick a random index $k \leftarrow \{1, \ldots, K\}$ and then calculate p_k. Although this method minimizes the number random bits and generates exactly uniform primes, it is computationally expensive as the current best algorithms computing the first K primes in subexponential time in the size of the input, the number of bits of K. For example, the Atkin's sieve [4] runs in $\mathcal{O}(K/\lg\lg K)$. Even if we restrict ourselves to only computing p_k, the best algorithms are still subexponential. In [5, Chapter 9.9], it is shown that p_k can be computed efficiently given an oracle that compute $\pi(x)$ and vice versa. The current best

algorithm for computing $\pi(x)$ runs in $\mathcal{O}\left(x^{1/2+\epsilon}\right)$[10] for any $\epsilon > 0$ which is still subexponential in the number of bits of x.

To improve the time complexity of the encryption scheme, we look for polynomial time algorithms that produce *almost* uniform primes with the least number of random bits possible. A trivial algorithms for producing uniform primes is given in Algorithm 1.

Algorithm 1: Generating uniform primes

input : Positive integer n.
output: A uniformly choen prime number less than or equal to n.

```
1 repeat
2     p ← {2, ..., n};
3 until p is prime
4 return p
```

When we use a deterministic primality test in line 3 of Algorithm 1 the distribution of primes is exactly uniform. This algorithm may never terminate, however, we expect it to stop after $\mathcal{O}(\lg n)$ steps. Because by the prime number theorem, Theorem 1,

$$\frac{\pi(n)}{n} \sim \frac{1}{\ln n} \tag{6}$$

Hence, on average in $\mathcal{O}(\lg n)$ steps a prime number p is chosen. As a result, Algorithm 1 uses an average of $\mathcal{O}\left((\lg n)^2\right)$ random bits. The current state-of-the-art deterministic primality tests, AKS, runs in $\tilde{\mathcal{O}}\left((\lg n)^6\right)$ [11] which means that on average Algorithm 1 terminates in $\tilde{\mathcal{O}}\left((\lg n)^7\right)$.

We can further improve the time complexity of the Algorithm 1 if we use randomized primality tests. These tests can determine whether a number p is prime with high probability.

Algorithm 2: Generating uniform primes

input : Positive integer n.
output: A uniformly choen prime number less than or equal to n.

```
1 repeat
2     p ← {2, ..., n};
3 until p is probably a prime
4 return p
```

For example, the Miller-Rabin test might declare a composite number as a prime, however, this happens with low probability, as low as desired. The output of the Algorithm 2 is not a unifrom prime number as it can be composite, however, the distribution of prime numbers is equiprobable over all primes less than n. Each round of the Miller-Rabin test, it uses $\lg p$ random bits where p is the number that is to be tested. Therefore, we still use an average of $\mathcal{O}\left((\lg n)^2\right)$ random bits. The test itself runs in $\mathcal{O}\left((\lg n)^3\right)$ [5] thus, the Algorithm 2 terminates in $\mathcal{O}\left((\lg n)^4\right)$.

In this report, we implement the Miller-Rabin test since it is more efficient and easier to implement. Furthermore, by exectuing this test an appropriate number of rounds, we can ensure that the resulting distribution is statistically close to the uniform distribution over primes.

3.2 Miller-Rabin analysis

Miller-Rabin is a well-known random primality test algorithm. Let $MR(n, k)$ be the distribution of Miller-Rabin algorithm on the prime candidate n with k tests. Let $\mathcal{P}$ be the set of primes, then from [5, Theorem 9.4.5] we have the following probabilities.

$$\Pr[MR(n,k) = 1 \mid n \in \mathcal{P}] = 1 \tag{7}$$

$$\Pr[MR(n,k) = 1 \mid n \notin \mathcal{P}] \leq 4^{-k} \tag{8}$$

Consider the following random prime number genrator, $GMR(N, s, k)$, as described in Algorithm 3. This algorithm, samples numbers uniformly and then checks if they are prime using the Miller-Rabin test. The parameter N is the upperbound, s is the maximum number of samples, and k is the number of repeats in the underlying Miller-Rabin test. We analyze the distribution of $GMR(N, s, k)$. Let n_i

Algorithm 3: $GMR(N, s, k)$

input : positive integers N, s, k
output: A uniformly chosen prime number less than or equal to N

```
for i = 1 → s do
    n ← {1, 2, ..., N}
    if MR(n, k) then
        return n
    end
end
return ⊥
```

denote the random variable n in the i_{th} iteration.

$$\Pr[GMR(M, s, k) = \perp] = \Pr[MR(n_1, k) = \cdots = MR(n_s, k) = 0] \tag{9}$$

$$= \prod_{i=1}^{s} \Pr[MR(n_i, k) = 0] \qquad \text{(Independence)} \tag{10}$$

$$= \prod_{i=1}^{s} \Pr[MR(n_i, k) = 0 \mid n_i \notin P] \Pr[n_i \notin P] \tag{11}$$

$$\leq \prod_{i=1}^{s} \left(1 - \frac{\pi(N)}{N}\right) \tag{12}$$

$$= \left(1 - \frac{\pi(N)}{N}\right)^s \tag{13}$$

$$\leq \left(1 - \frac{1}{6 \ln N}\right)^s \qquad \text{(Theorem 2)} \tag{14}$$

If we bound this error probability with ϵ, then we get the following bound on s.

$$s \geq \frac{\ln \epsilon}{\ln\left(1 - \frac{1}{6 \ln N}\right)} \approx -6 \ln N \ln \epsilon = -\frac{6}{(\lg e)^2} \lg N \lg \epsilon \tag{15}$$

for sufficiently large N by the Taylor series approximation.

The probability that the result of $GMR(N, s, k)$ is composite, given that it is not $\perp$ is as follows.

$$\Pr[GMR(N, s, k) \notin P \mid GMR(N, s, k) \neq \perp] \leq \sum_{i=1}^{s} \Pr[MR(n_i, k) = 1, n_i \notin P] \qquad \text{(Union bound)}$$

$$= \sum_{i=1}^{s} \Pr[MR(n_i, k) = 1 \mid n_i \notin P] \Pr[n_i \notin P] \tag{16}$$

$$\leq \sum_{i=1}^{s} 4^{-k} \left(1 - \frac{\pi(N)}{N}\right) \tag{17}$$

$$= s 4^{-k} \left(1 - \frac{\pi(N)}{N}\right) \tag{18}$$

$$\leq s 4^{-k} \left(1 - \frac{1}{6 \ln N}\right) \qquad \text{(Theorem 2)}$$

If we bound this error probability with δ, then we get the following bound on s.

$$s \leq \left(1 - \frac{1}{6 \ln N}\right)^{-1} 4^k \delta \approx 4^k \delta \tag{19}$$

for sufficiently large N. Let $\epsilon = 2^{-l}$ and $\delta = 2^{-q}$ with $l, q \geq 0$. Then,

$$\frac{6}{(\lg e)^2} l \lg N \leq 3l \lg N \leq s \leq 2^{2k-q} \tag{20}$$

Note that, $s = 3l \lg N$ and $k = \dfrac{\lg 3l + \lg\lg N + q}{2}$ satisfies both inequalities.

4 Identification Codes

Identification is a communication paradigm introduced by Ahlswede *et al* [1]. In identification schemes the receiver wants to know whether a certain message has been sent or not. This is in contrast to the Shannon's transmission schemes where the receiver wants to know the content of the message.

More formally, the sender and the receiver both have the message set $\mathcal{M}$ and the receiver is interested in some message $m \in \mathcal{M}$. Ofcourse, when the sender knows m, he can send a bit to indicate that he intends to send m or not. Thus, we assume that the sender does not know m.

This problem can be trivially addressed by transmission codes, the receiver decodes the received code to $\hat{m}$ and then decides if $\hat{m} = m$. However, the identification codes require exponentially shorter blocklength to identify the same number of messages. This improvement is achieved mainly by relaxing the condition that the decoding sets need be disjoint. By allowing the decoding sets to have slight overlap, Ahlswede *et al* [1] have shown that there exists coding schemes that can identify $2^{2^{nC}}$ messages where C is the Shannon capacity of the DMC channel.

There are two kinds of errors associated with an identification scheme. The first kind happens when the sender sends m but the receiver fails to identify it and hence *misses* the identification. The second kind happens when the sender send $m' \neq m$ and the receiver *falsely* identifies m instead.

Definition (Identification code): An identification code $(n, N, \lambda_1, \lambda_2)$ for a DMC channel $\mathcal{W}^n(\mathcal{X}^n|\mathcal{Y}^n)$ is a set $\{(Q(\cdot|i), \mathcal{D}_i)\}_{i\in[N]}$ where $Q(\cdot|i)$ is a distribution over $\mathcal{X}^n$ to that encodes the message i – for determinstic encoder $Q(x_i|i) = 1$ for some $x_i \in \mathcal{X}^n$– and $\mathcal{D}_i \subset \mathcal{Y}^n$ is the decoding set of the message i. The first and second kind errors are bounded by λ_1 and λ_2, respectively.

$$P_{e,1}(i) = \sum_{x^n \in \mathcal{X}^n} Q(x^n|i)\mathcal{W}^n(\mathcal{D}_i^c|x^n) \leq \lambda_1 \tag{21}$$

$$P_{e,2}(i,j) = \sum_{x^n \in \mathcal{X}^n} Q(x^n|j)\mathcal{W}^n(\mathcal{D}_i|x^n) \leq \lambda_2 \tag{22}$$

For all $1 \leq i, j \leq N$ and $j \neq i$.

5 3-step Algorithm

The 3-step algorithm as described in [2] defines the following parameters.

- Let $\mathcal{M} = \{1, 2, \ldots, M\}$ be the message set and $\alpha > 1$. Let $K = \lceil (\lg M)^\alpha \rceil$.
- Let p_i denote the i_{th} prime. Let $\mathcal{M}' = \{1, 2, 3, \ldots, p_K\}$ and $K' = \lceil (\lg p_K)^\alpha \rceil$.
- Let us denote the set $\{1, 2, \ldots, \pi_l\}$ by $\mathbb{Z}_l^+$. Define the function $\phi_l : \mathbb{N} \to \mathbb{Z}_l^+$ as follows.

  $$\phi_l(n) = [n \mod p_l] + 1 \tag{23}$$

  Where $[n \mod p_l]$ is equal to the remainder of the division of n by p_l.

A round of communication in this scheme is as follows.

1. The sender chooses a key $k \leftarrow \mathcal{K} = \{1, 2, \ldots, K\}$ uniformly and transmits it.
2. The sender chooses another key $l \leftarrow \mathcal{K}' = \{1, 2, \ldots, K'\}$ uniformly and transmits it.
3. Given a message $m \in \mathcal{M}$, the sender transmits $\phi_l(\phi_k(m))$. Assuming that receiver wishes to identify $\hat{m} \in \mathcal{M}$, he calculate $\phi_l(\phi_k(\hat{m}))$ and compares it with $\phi_l(\phi_k(m))$. He identifies the message as $\hat{m}$ whenever $\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m}))$.

Example 1. Suppose $M = 100$ and $\alpha = 1.5$. Then, $K = 18$, $\pi_K = 61$, and $K' = 15$. Assume that $k = 12$ and $l = 7$ are chosen, with $\pi_k = 31$ and $\pi_l = 17$. Given $m = 71$, the sender sends the following code.

$$\underbrace{1100}_{k=12}\,\underbrace{111}_{l=7}\quad\underbrace{01011}_{\phi_l(\phi_k(71))=11}$$

We immediately find this scheme to be problematic when the sender sends the code all at once. There needs to be some separator that allows the receiver to determine where each part of the code starts and ends. However, let us continue with the example. If $\hat{m}_1 = 71$, then clearly the receiver correctly identifies the message. If $\hat{m}_2 = 32$

$$\phi_l(\phi_k(\hat{m}_2)) = 16 = (10000)$$

and the receiver correctly does not identify the message. However, when $\hat{m}_3 = 9$, $\phi_l(\phi_k(\hat{m}_3)) = 11 = (01011)$ which means the receiver falsely identifies $m = 71$ as $\hat{m} = 9$.

Theorem 4 ([2]). *The 3-step algorithm produces a $(n = n(M, \alpha), M, \lambda_1 = 0, \lambda_2)$ identification code – per Definition 1– such that $\lambda_2 \to 0$ as $M \to \infty$. Moreover, this coding scheme is optimal.*

$$\lim_{M\to\infty} \frac{\lg\lg M}{n(M)} = \frac{1}{\alpha} \tag{24}$$

Proof. It is obvious that $\lambda_1 = 0$, that is the receiver will not miss an identification since the channel is noiseless. For the second kind error probability consider the following lemmas.

Lemma 5. *Any positive integers m has at most $\lfloor \lg m \rfloor$ unique prime factors.*

Proof. Suppose $q_1, \ldots, q_k$ are all the prime factors of m. Then for some $\alpha_1, \ldots, \alpha_k \geq 1$

$$m = \prod_{i=1}^{k} q_i^{\alpha_i} \geq \prod_{i=1}^{k} 2^{\alpha_i} \geq 2^k$$

As a result, $k \leq \lfloor \lg m \rfloor$ as required. ■

Lemma 6. *For any $m, m' \in \mathcal{M} = \{1, 2, \ldots, M\}$ such that $m \neq \hat{m}$*

$$\left| \left\{ k \in \{1, 2, \ldots, K\} \;\middle|\; \phi_k(m) = \phi_k(\hat{m}) \right\} \right| \leq \lg M \tag{25}$$

Proof. The given set is the set of common prime factors of m and $\hat{m}$ that are less than or equal to p_K. The inequality immediatly follows from the fact that $m, \hat{m} \leq M$ and M has at most $\lg M$ prime factors. ■

We can derive an upper bound for the second kind error.

$$\begin{aligned}
P_{e,2}(m, \hat{m}) &= \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid m \neq \hat{m}] && (26)\\
&= \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) = \phi_k(\hat{m})] \Pr[\phi_k(m) = \phi_k(\hat{m}) \mid m \neq \hat{m}] && (27)\\
&\quad + \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) \neq \phi_k(\hat{m})] \Pr[\phi_k(m) \neq \phi_k(\hat{m}) \mid m \neq \hat{m}] && (28)\\
&\leq \Pr[\phi_k(m) = \phi_k(\hat{m}) \mid m \neq \hat{m}] + \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) \neq \phi_k(\hat{m})] && (29)\\
&\leq \frac{\lg M}{K} + \frac{\lg M'}{K'} && (30)\\
&= \frac{\lg M}{\lceil (\lg M)^{\alpha} \rceil} + \frac{\lg p_K}{\lceil (\lg p_K)^{\alpha} \rceil} && (31)\\
&\leq \frac{1}{(\lg M)^{\alpha-1}} + \frac{1}{(\lg p_K)^{\alpha-1}} && (32)
\end{aligned}$$

By the prime number theorem, Theorem 1, $p_K \sim K \ln K$. As a result, $\lambda_2 \to 0$ as $M \to \infty$.

$$(\lg p_K)^{\alpha-1} \sim (\lg K + \lg\lg K - \lg\lg e)^{\alpha-1} \approx (\alpha \lg\lg M + \lg\lg\lg M + \lg\alpha - \lg\lg e)^{\alpha-1}$$

Finally, the blocklength is calculated by $n = \lceil \lg K \rceil + \lceil \lg K' \rceil + \lceil \lg p_{K'} \rceil$. By applying the prime number theorem 2

$$\begin{aligned} n &= \lceil \lg K \rceil + \lceil \lg K' \rceil + \lceil \lg p_{K'} \rceil \\ &= \lceil \lg \lceil (\lg M)^{\alpha} \rceil \rceil + \lceil \lg \lceil (\lg p_K)^{\alpha} \rceil \rceil + \lceil \lg p_{K'} \rceil \\ &\approx \alpha \lg \lg M + \alpha \lg \lg p_K + \lg p_{K'} \\ &\approx \alpha \lg \lg M + (1 + o(1)) \lg \lg \lg M + (\alpha + o(1)) \lg \lg \pi_K \\ &\approx \alpha (1 + o(1)) \lg \lg M \end{aligned}$$

which was what was wanted. ■

This scheme requires both sender and receiver to have access to a prime generation algorithm that given k computes the k_{th} prime, p_k. As we have mentioned earlier, this problem does not have polynomial algorithm yet. To alleviate these inefficiencies we propose the following modifications.

5.1 Modified 3-step algorithm

Consider the following parameters.

- Let $\mathcal{M} = \{1, 2, \ldots, M\}$ be the message set and let $K = \lceil (\lg M)^{\alpha} \rceil$, $K' = \lceil (\lg K)^{\alpha} \rceil$, for some constant $\alpha > 1$.
- Let us denote the set $\{1, 2, \ldots, l\}$ by $\mathbb{Z}_l^+$. Define the function $\phi_l : \mathbb{N} \to \mathbb{Z}_l^+$ as follows.

$$\phi_l(n) = [n \mod l] + 1 \tag{33}$$

  Where $[n \mod l]$ is equal to the remainder of the division of n by l.

A round of communication in this scheme is as follows.

1. The sender chooses a probabilistic prime k from the set $\mathcal{K} = \{1, 2, \ldots, K\}$ by a prime number generator and transmits it.
2. The sender chooses another probabilistic prime l from the set $\mathcal{K}' = \{1, 2, \ldots, K'\}$ by the same prime number generator and transmits it.
3. Given a message $m \in \mathcal{M}$, the sender transmits $\phi_l(\phi_k(m))$. Assuming that receiver wishes to identify $\hat{m} \in \mathcal{M}$, he calculates $\phi_l(\phi_k(\hat{m}))$ and compares it with $\phi_l(\phi_k(m))$. He identifies the message as $\hat{m}$ whenever $\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m}))$.

Theorem 7. *This coding scheme is optimal.*

$$\lim_{M \to \infty} \frac{\lg \lg M}{n(M)} = \frac{1}{\alpha} \tag{34}$$

Proof. The blocklength is calculated by $n = \lceil \lg K \rceil + \lceil \lg K' \rceil + \lceil \lg l \rceil$.

$$\begin{aligned} n &= \lceil \lg K \rceil + \lceil \lg K' \rceil + \lceil \lg l \rceil && (35) \\ &\approx \lceil \lg K \rceil + 2 \lceil \lg K' \rceil && (36) \\ &\approx \alpha \lg \lg M + 2\alpha \lg \lg K && (37) \\ &\approx \alpha \lg \lg M + 2\alpha \lg \lg \lg M + 2\alpha \lg \alpha && (38) \\ &\approx \alpha (1 + o(1)) \lg \lg M && (39) \end{aligned}$$

which was what was wanted. ■

The error analysis this code depends on the prime number generator. We will be using the simple prime number generator Algorithm 3 based on the Miller-Rabin primality test.

Theorem 8. *The modified 3-step algorithm produces a $(n = n(M, \alpha), M, \lambda_1 = 0, \lambda_2)$ identification code – per Definition 1– such that $\lambda_2 \to 0$ as $M \to \infty$. Moreover, this coding scheme is optimal.*

$$\lim_{M \to \infty} \frac{\lg \lg M}{n(M)} = \frac{1}{\alpha} \tag{40}$$

We employ the same techniques used by Ahlswede by first calculating the following probability.

Lemma 9. *Suppose $m, \hat{m} \in \mathcal{M} = \{1, 2, \ldots, M\}$ and $p \leftarrow \mathcal{A}(K, \epsilon)$ is a probabilistic prime generated by randomized algorithm $\mathcal{A}$ such that all primes less than or equal to K are generated equiprobably and*

$$\Pr[p \notin \mathcal{P}] \leq \epsilon$$

Then,

$$\Pr[\phi_p(m) = \phi_p(\hat{m}) \mid m \neq \hat{m}] \leq \frac{\lg M}{\pi(K) \lg \lg M} + \epsilon$$

Proof. From Theorem 3, for all $\delta > 0$ and sufficiently large n,

$$\omega(n) \leq \frac{\ln n}{\ln \ln n} + \delta \tag{41}$$

Then, by the previous argument

$$\frac{|\{\pi \leq K \mid \phi_\pi(m) = \phi_\pi(\hat{m})\}|}{|\{\pi \leq K\}|} \leq \frac{\max_{c \in \mathcal{M}} \omega(c)}{\pi(K)} \leq \frac{\lg M}{\pi(K) \lg \lg M} \tag{42}$$

Therefore,

$$\begin{aligned}
\Pr[\phi_p(m) = \phi_p(\hat{m}) \mid m \neq \hat{m}] &= \Pr[\phi_p(m) = \phi_p(\hat{m}) \mid p \in \mathcal{P}, m \neq \hat{m}] \Pr[p \in \mathcal{P}] && (43)\\
&\quad + \Pr[\phi_p(m) = \phi_p(\hat{m}) \mid p \notin \mathcal{P}, m \neq \hat{m}] \Pr[p \notin \mathcal{P}] && (44)\\
&\leq \Pr[\phi_p(m) = \phi_p(\hat{m}) \mid p \in \mathcal{P}, m \neq \hat{m}] + \epsilon && (45)\\
&\leq \frac{\lg M}{\pi(K) \lg \lg M} + \epsilon && (46)
\end{aligned}$$

■

The proof of Theorem 8 works as follows.

Proof. Let $m, \hat{m} \in \mathcal{M} = \{1, 2, \ldots, M\}$ be given such that $m \neq \hat{m}$. Let $\mathcal{A}$ be the randomized prime generator as described above. Suppose the probable primes $k \leftarrow \mathcal{A}(K, \epsilon)$ and $l \leftarrow \mathcal{A}(K', \epsilon)$ are generated with $K = \lceil (\lg M)^\alpha \rceil$ and $K' = \lceil (\lg K)^\alpha \rceil$. The probability of the second kind error is given as follow

$$\begin{aligned}
P_{e,2}(m, \hat{m}) &= \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid m \neq \hat{m}] && (47)\\
&= \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) = \phi_k(\hat{m}), m \neq \hat{m}] \\
&\quad \Pr[\phi_k(m) = \phi_k(\hat{m}) \mid m \neq \hat{m}] \\
&\quad + \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) \neq \phi_k(\hat{m}), m \neq \hat{m}] && (48)\\
&\quad \Pr[\phi_k(m) \neq \phi_k(\hat{m}) \mid m \neq \hat{m}] \\
&\leq \Pr[\phi_k(m) = \phi_k(\hat{m}) \mid m \neq \hat{m}] + \Pr[\phi_l(\phi_k(m)) = \phi_l(\phi_k(\hat{m})) \mid \phi_k(m) \neq \phi_k(\hat{m})] && (49)\\
&\leq \frac{\lg M}{\pi(K) \lg \lg M} + \frac{\lg K}{\pi(K') \lg \lg K} + 2\epsilon && \text{(Lemma 9)}\\
&\leq \frac{6 \lg(M) \lg(K)}{K \lg \lg M} + \frac{6 \lg(K) \lg(K')}{K' \lg \lg K} + 2\epsilon && \text{(Theorem 2)}\\
&\approx \frac{6\alpha \lg \lg M}{(\lg M)^{\alpha-1} \lg \lg M} + \frac{6\alpha \lg \lg K}{(\lg K)^{\alpha-1} \lg \lg K} + 2\epsilon && (50)\\
&= \frac{6\alpha}{(\lg M)^{\alpha-1}} + \frac{6\alpha}{(\lg K)^{\alpha-1}} + 2\epsilon && (51)
\end{aligned}$$

Since ϵ can be chosen as small as possible, then $\lambda_2 \to 0$ as $M \to \infty$. ■

5.2 Performance

As oppose to the original 3-step algorithm of Ahlswede, in our revision, most of the work is done by the transmitter. Specifically, the transmitter runs the prime number generator $\mathcal{A}$ twice. Suppose the simple Algorithm 3 is used with Miller-Rabin primality test. By the analysis given in the Section 3.1,

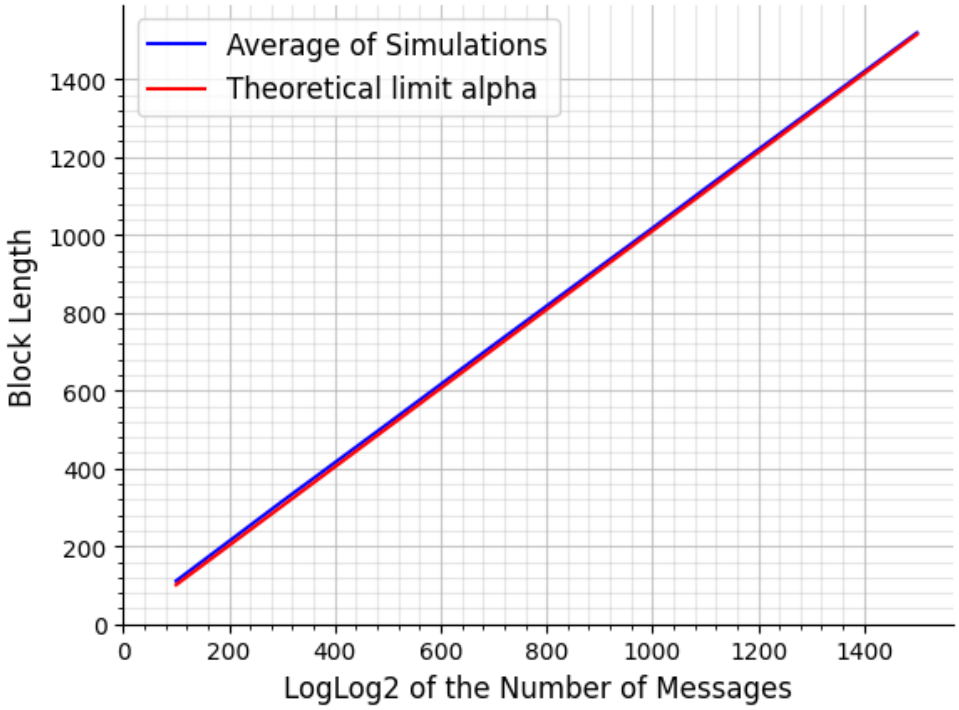

Figure 1: The block length vs lg lg of the number of messages curve. The block length is calculated as the $\lceil \lg k \rceil + 2\lceil \lg l \rceil$ where k, l are the probable primes generated. The lg lg of the number of messages are incremented by 100.

this algorithm on average terminates in $\mathcal{O}\Big((\lg n)^4\Big)$ where n is its input. Therefore, on input K, the algorithm runs in

$$\mathcal{O}\Big((\lg K)^4\Big) = \mathcal{O}\Big((\alpha \lg \lg M)^4\Big) = \mathcal{O}\big(n^4\big)$$

and on input K', it runs in

$$\mathcal{O}\Big((\lg K')^4\Big) = \mathcal{O}\Big((\alpha \lg \lg K)^4\Big) = \mathcal{O}\Big((\lg n)^4\Big)$$

All together, we have significantly improved time complexity of the scheme without affecting its error probability or blocklength. We should mention that this is the time complexity to generate prime numbers. To compute the actual code, we need to divide a 2^n-bit message by an n-bit prime. However, this is common in all identification schemes as we are required to process exponentially larges messages.

5.3 Simulation

We have implemented a code that mainly runs the Algorithm 2 to produce two probable prime numbers. Then it calculates the blocklength, error probability, and the running time of the simulation. On $\alpha = 1.01$ we get the following curves.

6 Generalizations of 3-step algorithm

In the 3-step algorithm, we produce two random prime numbers and then compute a function of the message based on these two primes. In general, this is viewed as producing a local randomness j and computing a message specific tag function T_i with input j, $T_i(j)$ [8]. When working with a noiseless channel, the tagging function determines the second kind error probability. For example, the probability that the receiver identifies $\hat{i}$ when i is sent is calaculated as

$$P_{e,2}\Big(i, \hat{i}\Big) = \Pr[T_i(j) = T_{\hat{i}}(j)]. \tag{52}$$

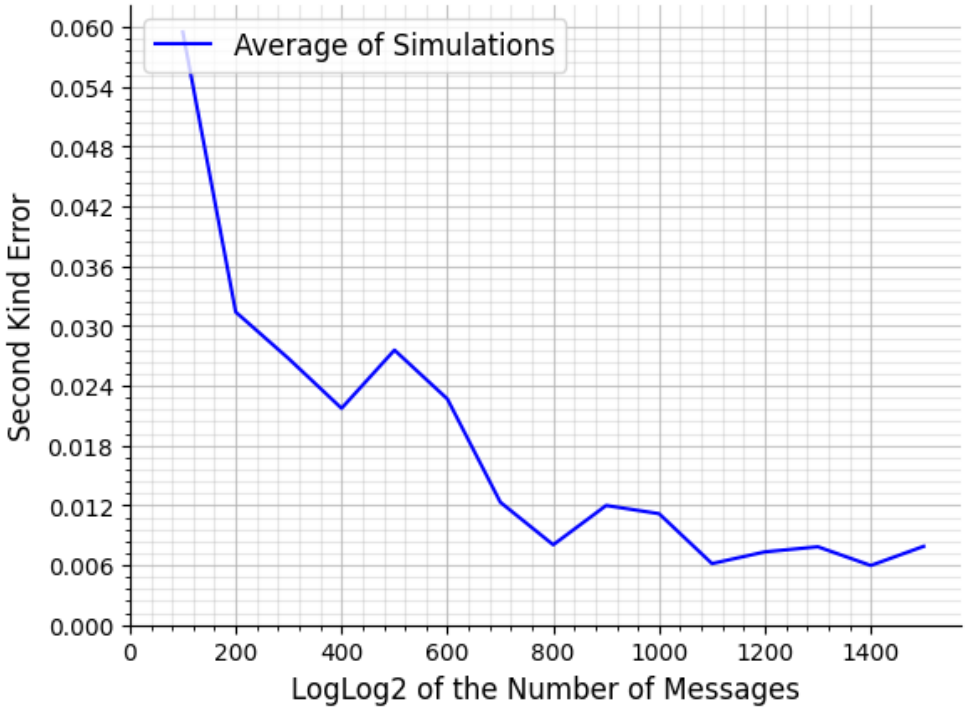

Figure 2: The average second kind error vs $\lg\lg$ of the number of messages curve. The second kind error is calculated as $1/l$ where l is the second probable prime generated. Furthermore, the average taken over 600 rounds of simulation. The $\lg\lg$ of the number of messages are incremented by 100.

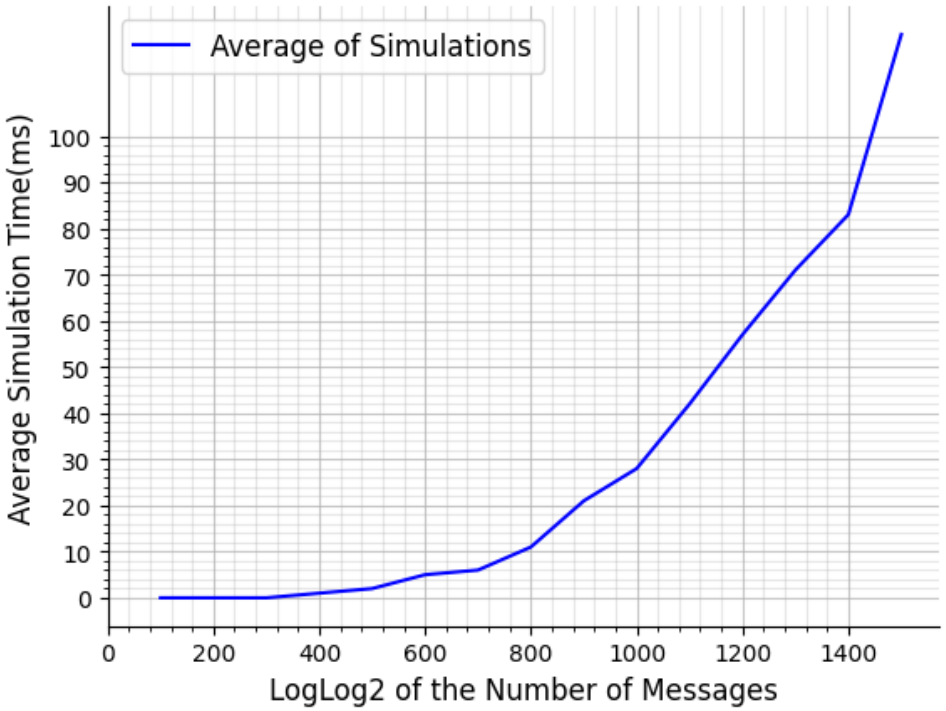

Figure 3: The average time vs $\lg\lg$ of the number of messages curve. The time it takes to generate two probable primes and the average taken over 600 rounds of simulation. The $\lg\lg$ of the number of messages are incremented by 100.

When j is chosen uniformly, this probability can be viewed as the probability of the collision of the hash function $h_j(x) = T_x(j)$. The families of hash functions for which this probability of collision is bounded are called *almost universal*. A formal definition was first given in [7] which is as follows.

Definition: Let $H = \{h : X \to Y\}$ be a family of hash functions from X to Y and let ϵ be a positive real number. H is said to be ϵ-almost universal if for all distinct $x_1, x_2 \in X$

$$|\{h \in H \,|\, h(x_1) = h(x_2)\}| \leq \epsilon |H| \tag{53}$$

If $h \leftarrow H$ uniformly,

$$\Pr[h(x_1) = h(x_2)] \leq \epsilon \tag{54}$$

Consider the following identification scheme for noiseless channels that uses almost universal functions.

Definition: Let $\mathcal{M} = \{1, \ldots, M\}$ and $H = \{h_a : X \to Y\}_{a \in [I]}$ be a family of ϵ-almost universal hash functions indexed by the set $[I] = \{0, 1, \ldots, I-1\}$. A round of communication round works as follows.

1. The sender chooses $a \leftarrow [I]$ uniformly. Then, he sends the code $(a, h_a(m))$ to the receiver.
2. Upon receiving the code $(a, h_a(m))$, the receiver calculates $h_a(\hat{m})$ and identifies $\hat{m}$ whenever $h_a(m) = h_a(\hat{m})$.

The blocklength of the identification scheme in Definition 3 is

$$n = \lg I + \lg |Y| \tag{55}$$

The number of random bits required is

$$r = \lg I \tag{56}$$

And the second kind error probability is given by

$$\Pr[h_a(m) = h_a(\hat{m}) \mid m \neq \hat{m}] \leq \epsilon \tag{57}$$

Thus, we have shown the following statement.

Theorem 10. *If there exists a class of ϵ-almost universal hash functions $H = \left\{h_a : \mathbb{F}_q^n \to \mathbb{F}_q\right\}_{a \in [I]}$ then there exists a $(\lg I + \lg q, q^n, 0, \epsilon)$ identification code for noiseless binary channel.*

6.1 Formulation of 3-step algorithm in terms of universal hash function

We now give an explicit constuction of the Definition 3 according to the Section 5.1. Let $A(n) = \{1, 2, \ldots, 2^n\}$, $B(n) = \{1, 2, \ldots, n^\alpha\}$ for some $\alpha > 1$, $K = \left\{2, 3, \ldots, p_{\pi(n^\alpha)}\right\}$, and define hash family $H(n) = \{h_k : A \to B \,|\, k \in K\}$ given as follows.

$$h_p(a) = [a \mod p] + 1 \tag{58}$$

Lemma 11. *The class $H(n)$ described above is $\frac{\alpha}{n^{\alpha-1}}$-almost universal.*

Proof. Suppose $x, y \in A$ are distinct.

$$\Pr[h_p(x) = h_p(y)] = \frac{1}{\pi(n^\alpha)} |\{p \in K : p \mid |x - y|\}| \tag{59}$$

$$\leq \frac{\omega(x-y)}{\pi(n^\alpha)} \tag{60}$$

$$\approx \frac{\alpha}{n^{\alpha-1}} \tag{61}$$

■

Note that, H digests input exponentially, therefore, if we use H twice we can achieve double exponential compression. Firstly, consider the follwoing composition lemma.

Lemma 12. *Suppose $H_1 : A \to B$ and $H_2 : B \to C$ are ϵ_1 and ϵ_2-almost universal, respectively. The hash famliy $H = H_2 \circ H_1 : A \to C = \{h_2 \circ h_1 \,|\, h_1 \in H_1, h_2 \in H_2\}$ is $\epsilon = (\epsilon_1 + \epsilon_2)$-almost unviersal. It is shown that H is $(\epsilon_1 + \epsilon_2 - \epsilon_1\epsilon_2)$-almost universal [6].*

Proof. Let $x, y \in A$ be two distinct elements.

$$\Pr[h_2(h_1(x)) = h_2(h_1(y))] \leq \Pr[h_2(h_1(x)) = h_2(h_1(y)) \mid h_1(x) \neq h_1(y)] + \Pr[h_1(x) = h_1(y)] \quad (62)$$

$$\leq \epsilon_2 + \epsilon_1 \quad (63)$$

■

By Lemma 11 and Lemma 12, the hash family $H^2(n) = H(\alpha n) \circ H(2^n)$ is

$$\epsilon = \frac{\alpha}{(\alpha n)^{\alpha-1}} + \frac{\alpha}{2^{n(\alpha-1)}} = \mathcal{O}\left(\frac{1}{n^{\alpha-1}}\right) \quad (64)$$

-almost universal which maps $A = \left\{0, 1, \ldots, 2^{2^n}\right\}$ to $C = \{1, \ldots, (\alpha n)^\alpha\}$.

6.2 Current Directions

One way to construct identification codes is by re-purposing error correcting codes [12, 8]. The methods described in [12, 8] use the error correction codes to construct a tag function. As a matter of fact, Bierbrauer *et al* [6] has shown the equivalence between error correcting codes and almost universal hash function.

Theorem 13. *If there is a $[n, k, d]_q$ code, then there exists a $(1 - \frac{d}{n})$-almost universal hash family $H = \left\{h_a : \mathbb{F}_q^k \to \mathbb{F}_q\right\}_{a\in[n]}$ and conversely, if there is a ϵ-almost universal $H = \left\{h_a : \mathbb{F}_q^k \to \mathbb{F}_q\right\}_{a\in[n]}$ then there exists a $[n, k, n(1-\epsilon)]_q$ code.*

This theorem gives us the framework needed to compare code-based identification schemes such as [8].

Example 2. The 3-step algorithm is equivalent to almost universal hash family H^2, as described above. The hash functions are indexed by

$$I = \{(p_i, p_j) \,|\, 1 \leq i \leq \pi(2^{n\alpha}),\ 1 \leq j \leq (\alpha n)^\alpha\} \quad (65)$$

Instead of transmitting the indicies (i, j) we transmit the actual key (p_i, p_j). Thus, to transmit a key pair we need send n_1 bits.

$$n_1 = \lg p_{\pi(2^{n\alpha})} + \lg p_{\pi((\alpha n)^\alpha)} \quad (66)$$

$$\sim \lg \pi(2^{n\alpha}) + \lg \ln \pi(2^{n\alpha}) + \lg \pi((\alpha n)^\alpha) + \lg \ln \pi((\alpha n)^\alpha) \quad (67)$$

$$= (1 + o(1)) \lg \pi(2^{n\alpha}) \quad (68)$$

$$\sim (1 + o(1))(\lg 2^{n\alpha} - \lg \ln 2^{n\alpha}) \quad (69)$$

$$= (1 + o(1))\alpha n \quad (70)$$

The output set is $\{1, \ldots, (\alpha n)^\alpha\}$ which requires $\alpha \lg \alpha n$ bits to transmit. Therefore, the total block-length is $n' = (1 + o(1))\alpha n$. The input set is $\left\{1, \ldots, 2^{2^n}\right\}$ and the probability of collision is bounded by $\mathcal{O}\left(\frac{1}{n^{\alpha-1}}\right)$. As a result, we have an

$$\left((1 + o(1))\alpha n,\ 2^{2^n},\ 0,\ \mathcal{O}\left(\frac{1}{n^{\alpha-1}}\right)\right) \quad (71)$$

identification scheme.

Example 3. The identification code in [8] is constructed from a double Reed-Solomon code

$$[q^{k+1}, kq^{k-\delta}, (q - k + 1)(q^k - q^{k-\delta} + 1)]_q \quad (72)$$

By Theorem 13, there exists an almost universal $H = \left\{h_a : \mathbb{F}_q^{kq^{k-\delta}} \to \mathbb{F}_q\right\}_{a\in[q^{k+1}]}$ with the collision probability bounded by

$$1 - \frac{(q-k+1)(q^k - q^{k-\delta}+1)}{q^{k+1}} \leq \frac{k-1}{q} + \frac{1}{q^\delta} \tag{73}$$

Therefore, the resulting identification code is

$$\left((k+2)\lg q,\ q^{kq^{k-\delta}},\ 0,\ \frac{k-1}{q} + q^{-\delta}\right) \tag{74}$$

References

[1] R. Ahlswede and G. Dueck. Identification via channels. *IEEE Transactions on Information Theory*, 35(1):15–29, 1989.

[2] R. Ahlswede and B. Verboven. On identification via multiway channels with feedback. *IEEE Transactions on Information Theory*, 37(6):1519–1526, 1991.

[3] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer New York, 1976.

[4] Arthur Atkin and Daniel Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, 73(246):1023–1030, 2004.

[5] E. Bach, J.O. Shallit, and P.J. Shallit. *Algorithmic Number Theory: Efficient algorithms*. Number v. 1 in Algorithmic Number Theory. MIT Press, 1996.

[6] Jürgen Bierbrauer, Thomas Johansson, Gregory Kabatianskii, and Ben Smeets. On families of hash functions via geometric codes and concatenation. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 331–342, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[7] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[8] Sencer Derebeyoglu, Christian Deppe, and Roberto Ferrara. Performance analysis of identification codes. *CoRR*, abs/2007.06372, 2020.

[9] G. H. Hardy and S. Ramanujan. The normal number of prime factors of a number n. *Quart. J.*, 48:76–92, 1917.

[10] Jeffrey C Lagarias and Andrew M. Odlyzko. Computing π (x): An analytic method. *Journal of Algorithms*, 8(2):173–191, 1987.

[11] Hendrik W Lenstra Jr and Carl B Pomerance. Primality testing with gaussian periods. *Journal of the European Mathematical Society*, 21(4):1229–1269, 2019.

[12] S. Verdu and V.K. Wei. Explicit construction of optimal constant-weight codes for identification via channels. *IEEE Transactions on Information Theory*, 39(1):30–36, 1993.

A Simulation Code

```
1  void uniform_primes(mpz_ptr rop, mp_bitcnt_t n, uint32_t s, uint32_t
       k)
2  {
3      for (uint32_t i = 0; i < s; i++)
4      {
5      mpz_urandomb(rop, gmp_generator, n);
6          if (mpz_probab_prime_p(rop,k))
7              return;
8      }
9  }
10
11 uint64_t simulate(uint64_t loglog_number_of_messages,  uint64_t
       number_of_encoding_iterations, long double alpha, mpq_ptr
       avg_error)
12 {
13     uint32_t l = 10, q = 10;
14     uint64_t bit1 = loglog_number_of_messages + ((uint64_t)(log2l(
          alpha))) + 1;
15     uint64_t bit2 = (uint64_t)(log2l(alpha) + log2l(bit1)) + 1;
16
17     uint32_t s1 = l / 2 * bit1, s2 = l / 2 * bit2;
18     uint32_t k1 = (3 + (uint64_t)log2l(bit1) + q) / 2,
19              k2 = (3 + (uint64_t)log2l(bit2) + q) / 2;
20
21     mpz_t prime1;
22     mpz_t prime2;
23          mpz_init(prime1);
24          mpz_init(prime2);
25
26
27     uniform_primes(prime1, bit1, s1, k1);
28     uniform_primes(prime2, bit2, s2, k2);
29
30     mpq_t error;
31     mpq_init(error);
32     mpq_set_z(error,prime2);
33     mpq_canonicalize(error);
34     mpq_inv(error,error);
35     mpq_add(avg_error,avg_error,error);
36     uint64_t block_length =  mpz_sizeinbase(prime1,2) + 2 *
          mpz_sizeinbase(prime2,2);
37     return block_length;
38 }
39
40 uint64_t simulate_nonprime(uint64_t loglog_number_of_messages,
       uint64_t number_of_encoding_iterations, long double alpha, mpq_ptr
        avg_error)
41 {
42     uint64_t bit1 = loglog_number_of_messages + ((uint64_t)(log2l(
          alpha))) + 1;
43     uint64_t bit2 = (uint64_t)(log2l(alpha) + log2l(bit1)) + 1;
```