Verification of the Winnow Algorithm for Error Correction in Quantum Key Distribution

S. Nicholls[1] and T. Foldy-Porto[1]
[1] *Department of Physics, Yale University*

(Dated: 9 April 2019)

In this study, we implement and verify the Winnow error correction algorithm that is presented in "Fast, efficient error reconciliation for quantum cryptography", by W. T. Buttler et al.[1] Using a pulsed laser, a variable polarizer, and measurement devices for two non-orthogonal states—as outlined in the B92 protocol presented by Charles Bennett in 1992—we verified the error correcting statistics that are claimed in the original Winnow paper.[2] Our results were consistent with the theoretical model proposed by Buttler et al., and we were able to reproduce their experimental results with a high degree of correlation.

I. INTRODUCTION

Throughout history, people have devised increasingly complicated means of communication. From smoke signals to electrical transmission lines, the goal is the same: transmit information across physical space. The differences between various means of communication can be found in metrics like speed, security, and efficiency. For example, the rate at which information can be transmitted using smoke signals is certainly slower than using electrical bits, and since anyone with eyes can observe the smoke signals, they are also less secure. The concepts of speed and efficiency were formalized by Claude Shannon in 1948 in his seminal paper, "A mathematical theory of communication".[3] In this paper, Shannon proposes a rigorous definition for information entropy, the amount of information revealed in a discrete, probabilistic event, and for metrics like channel capacity, the rate at which information can be sent over a channel.

The question of security falls under the field of cryptography, which primarily concerns itself with the encoding of information such that it can only be decoded by the 'right' person. The relationship between cryptography and information transmitted by a discrete statistical source, such as a computer, was addressed by Shannon in a paper a year later, in 1949, titled "Communication theory of secrecy systems."[4] This paper was the first to approach the creation of cryptographic algorithms from the standpoint of information theory. At that point in the twentieth century, recent developments in semiconductors allowed the digital electronics industry to boom, rendering Shannon's ideas on cryptography increasingly relevant.

By the early 1970s, society had become dependant on digital systems and the secure transfer of information, and it was necessary to develop *de jure* standards for encrypting information. The first of such standards, creatively named DES (Data Encryption Standard), was published in 1975. Since then, many other cryptographic schemes have been proposed: RSA, AES, and 3DES, among others. Both DES and AES are symmetric encryption algorithms, meaning if Alice is sending a message to Bob, then Alice encrypts the message using the same key, usually a large number, that Bob then uses to decrypt the message. Since the encryption algorithms are quite hard to break—the timescale on which modern computers can crack modern algorithms is generally years—the security of the algorithm as a whole rests on the distribution of this private key. This is where quantum mechanics is helpful.

A. Quantum Key Distribution

One method for securely distributing an encryption key is to exploit properties of quantum mechanics, namely that quantum states cannot be copied and are destroyed once measured. These properties allow two parties to create a random key, known only to them, such that any interception of the key produces an effect measureable by the sender and receiver. The security of the distribution method relies on the fact that measuring a quantum system disturbs the state of the system, collapsing it to a single value. If the sender ensures that, prior to any measurement, the initial state (of the data to be sent) meets certain criteria, then the receiver can expect certain statistical properties of the data they receive. As a quick example, supposed the sender sends the state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1}$$

Where $|0\rangle$ and $|1\rangle$ are two orthogonal states. A measurement of this state yields a 50% chance of measuring either $|0\rangle$ or $|1\rangle$. Any eavesdropper who intercepts this state cannot reproduce it faithfully. If the sender is careful in the states they choose to send, then any eavesdropping will yield receiving probabilities that deviate from what's expected.

B. The B92 Protocol

The B92 protocol, invented in 1992 by Charles Bennett, is an algorithm for quantum key distribution (QKD).[2] The protocol is actually a simplification of the BB84 QKD protocol, which Bennett had worked on some

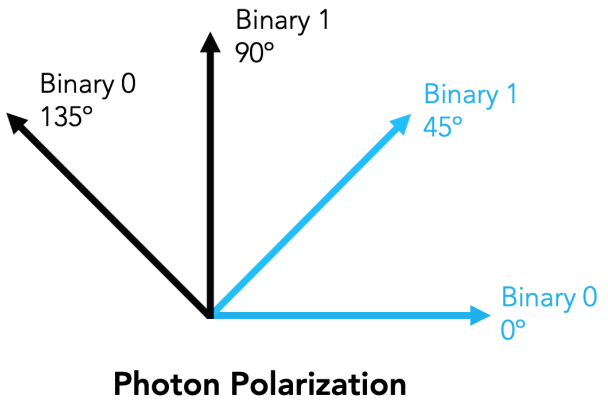

FIG. 1. The polarization states that are used in the B92 protocol. Our sender, Alice (blue), uses 0° and 45° polarization, while our receiver, Bob (black), uses 90° and 135° polarization. Due to this configuration, the bit that Alice sends has a 25% chance of being measured correctly by Bob, and a 0% chance of being measured incorrectly. For example, if Alice sends a 0 and Bob measures on the 90° channel, he will not detect any photons due to orthogonality of the two states. If he measures on the 135° channel, he will have a 50% chance of detecting a photon (by eq. 1).

years earlier, in 1984.[5] The two protocols are fundamentally the same, with the differences being that B92 is easier to implement but less secure than BB84. The protocol encodes bits into two non-orthogonal quantum states.

Our sender, Alice, sends her bits one-by-one in the form of polarized photons; typically, Alice's polarization states are 45° apart. The polarization state of each photon is determined by the bit value to be sent—a binary 1 means Alice sends a 0° polarized photon, a binary 0 means 90°. Our receiver, Bob, has a device which measures photons on a basis that is rotated 90° from Alice's, as shown in figure 1. In each frame where a photon is sent by Alice, Bob randomly chooses to measure on either the 90° channel (channel 1) or the 135° channel (channel 2). This means that there is a 25% chance that Bob detects a photon, and if he detects a photon there is a 100% chance he measured is correctly (assuming the states are truly orthogonal).

After the transmission of data is complete, Bob tells Alice—over an insecure channel—which bits he recorded a photon on. Because the probability of successful measurement is 100%, Alice and Bob now know for certain a string of bits that they agree on. Suppose, however, that there was an eavesdropper on the line, Eve, who siphoned off some of the photons and read them on her own photon detector. Assuming Alice only sends one photon per bit, any photon measured by Eve cannot possibly be measured by Bob, due to the no-cloning theorem which states that quantum states cannot be duplicated. Even if Eve sends a replacement photon to Bob, so that he doesn't get suspicious of the missing photons, there is only a 25% chance that the replacement photon encodes the information that Alice sent. Therefore, Bob will not only find that when he and Alice try to send information to each other, their keys do not match, but he will notice a drop in his receiving statistics.

C. The Winnow Error Correction Algorithm

Even if there is no eavesdropper on the communication between Alice and Bob, there is still the chance that Bob erroneously records a photon. This error can be attributed to stray photons that find their way into Bob's detectors (from exterior light sources) or from the fact that his measurement basis isn't perfectly perpindicular to Alice's sending basis. For example, suppose Alice sends a binary 1, which corresponds to a 45° polarized photon. Normally, if Bob happens to measure that photon on his binary 0 basis, which is at 135° (see figure 1), there is no chance he detects a photon since 45° and 135° are orthogonal. However, if his detector is slightly off, say 130°, there is a small probability that Bob measures a binary 0, even though Alice sent a 1.

Lucky for us, there are various classical error correction techniques we can use to flip erroneous bits. One of these methods is the Winnow algorithm, proposed by W. T. Buttler et al. in 2003.[1] Essentially, Winnow is a conditional application of the Hamming error correction algorithm, which detects errors using blocks of data called *syndromes*, shown below in figure 2.

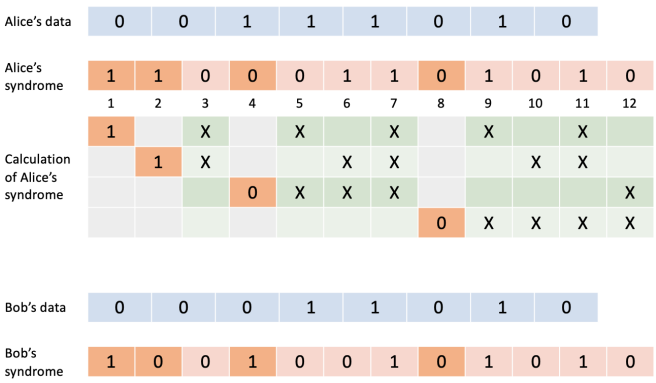

FIG. 2. Top: Alice's byte of data and the syndrome that she calculates from it. Middle, in green: how Alice's syndrome is calculated. Each of the four syndrome bits (in orange) is the row sum (modulo 2) of all the bits in which there is an 'X' in the corresponding column; e.g. bit 1 is the sum of bits 3, 5, 7, 9, and 11. Bottom: Bob's received data and syndrome, calculated in the same way as Alice's.

The syndromes are N-bit blocks of data that encode properties of a k-bit block of data, where N and k are integers that must satisfy the relationship

$$2^{N-k} \geq N + 1 \tag{2}$$

For example, in figure 2, Alice has an 8-bit block of data that she wishes to send to Bob. The smallest value of N that satisfies equation 2 is 12, so our syndrome is 12 bits long. The extra four bits that are added to Alice's data, called *redundant bits*, are located at positions that are multiples of two (slots 1, 2, 4, and 8) and are parity sums of various bits within Alice's byte. The first redundant bit is the sum (modulo 2) of all the bits in slots that have a 1 in the 1's place (odd numbers, in decimal). Similarly, the second redundant bit is the mod-2 sum of bits in slots that have a 1 in the 2's place (2, 3, 6, 7, 10, 11...). The third and fourth redundant bits are calculated according to the same logic. Note: the redundant bits do not include themselves in their parity sum.

Instead of sending her raw data, Alice instead sends her syndrome to Bob, who extracts the relevant byte of data (we'll call them the *useful bits*, from the appropriate slots. Assuming N is small enough that only one error occurs (one bit is flipped), we know the following things:

1. If it was a syndrome bit that got flipped, the byte of relevant information is unaffected.

2. If a useful bit is flipped, Bob will be able to recognize this because the syndrome will not be correctly calculated.

In order to figure out which bit is the source of the error, he simply XORs the redundant bits of his syndrome (having recalculated it to match the useful bits he received) with the redundant bits of the syndrome that Alice sent. For example, in figure 2, Alice has the byte 00111010 and corresponding syndrome 110001101010 (in this case, the redundant bits are 1100). When she sends it to Bob, the 6th bit of the syndrome (3rd bit of the useful byte) gets flipped and Bob receives 00011010 as the useful byte. When he recalculates his syndrome, he gets 100100101010 with redundant bits 1010. He XORs his redundant bits with Alice's:

$$1100 \oplus 1010 \rightarrow 0110 \tag{3}$$

The result, 0110 or 6, in decimal, points us to the fact that the 6th bit (counting from the right) is the source of our error. This may seem like a bit of black magic, but our result can be easily intuited from looking at the green table in figure 2, and working through the effect that a single bit flip has on the calculation of the redundant bits.

While Hamming's method of error correction increases the reliability of a channel, it decreases the channel efficiency. In our example, shown in figure 2 and described above, Alice wishes to send 8 bits to Bob, but to do so she must send a 12-bit syndrome. This means that if she was communicating with Bob purely using Hamming's algorithm, her channel efficiency—the ratio of useful bits sent to total bits sent—is $\frac{8}{12}$, or 0.67. Most of the time, however, the application of Hamming's algorithm is unnecessary because there were no errors in transmission.

The Winnow algorithm addresses this problem and increases the channel efficiency by conditionally applying Hamming's algorithm. Winnow first does a parity check of Alice's data and attaches a parity bit. In the example above, this means that every time Alice wishes to send 8 bits, she must send 9 bits, with the extra bit indicating total parity. Bob then calculates the total parity of the data he receives, and only if it doesn't match Alice's parity does he request her syndrome. If we assume that 0 errors occur 75% of the time, and 1 error occurs the remaining 25%, then our channel efficiency is:

$$0.75 \times (\frac{8}{9}) + 0.25 \times (\frac{8}{12}) = 0.83 \tag{4}$$

This is significantly better than the 0.67 channel efficiency given by a naive application of Hamming's algorithm. The Winnow algorithm succeeds at increasing the channel capacity without an increase in error rate when only zero or one errors occur in a given block of data, but fails if there are two or more errors.

II. METHODS

A. Setting the Pulse Intensity

In order to make the channel secure, it is necessary to minimize the probability that Alice sends more than a single photon per pulse. The reason is that if Alice sends multiple photons in a pulse (all of which would be at the same polarization), this opens up the possibility that Eve intercepts these photons, and by measuring a number of them, gains certain knowledge of the polarization state of the photons. If Eve has her measurement axis aligned 45 degrees from the polarization axis of an incoming photon, then whether or not she makes a positive measurement is a Bernoulli random variable with parameter $\frac{1}{2}$. Therefore, the larger the sample size that Eve is able to take, the probability that she does not make a positive measurement decreases. But so long as Eve makes one positive measurement, she knows the polarization of all the photons in a pulse, and can re-send an exact copy of the original pulse to Bob, without Alice or Bob being aware that she intercepted the message. The security of the communication channel is therefore inversely proportional to the average number of photons per pulse that Alice sends.

Due to randomness involved in the intensity of light produced by the laser, and the amount of light that makes it through the attenuators, we will model the number of photons per pulse as a Poisson random variable n.

$$P_{\bar{n}}(n) = \frac{\bar{n}^n e^{-\bar{n}}}{n!} \tag{5}$$

Since the detectors at Ch1 and Ch2 only tell us whether or not there is a measurement made, and not

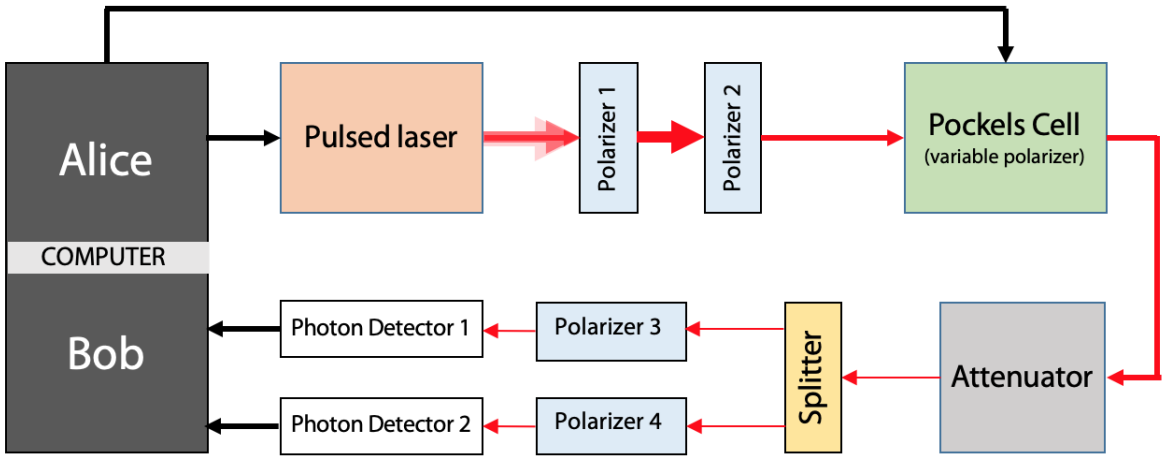

FIG. 3. Schematic diagram of our implementation of the B92 QKD technique. Alice sends a data stream to Bob via a pulsed laser (in the IR range). The laser pulses are attenuated by crossed polarizers (1 and 2) before being variably polarized to 0° or 45° by a Pockels cell, which is controlled by Alice. After that, the beam is further attenuated to achieve a single photon per pulse (on average). The photon is then split, with a 1/2 probability of going in either direction, and passed to either photon detector 1 or to photon detector 2, having passed through polarizers 3 or 4 respectively. Note: black arrows indicate electrical connections, while red arrows indicate information sent via the laser beam.

how many measurements were made, we can only measure the probability that we get a zero count, which is

$$P_{\bar{n}}(0) = e^{-\bar{n}} \tag{6}$$

And the probability that we get a non-zero count is

$$P_{\bar{n}}(n > 0) = 1 - e^{-\bar{n}} \tag{7}$$

From the attenuation of light when passing through crossed polarizers, we can deduce that

$$\bar{n} = n_0 \cos^2(\phi_1 - \phi_2) \tag{8}$$

Where n_0 is a background pulse intensity determined by the laser intensity, the angle between the first polarizer and the polarization axis of the laser beam, and the attenuator. $\phi_1 - \phi_2$ is the angle between Alice's polarizers (polarizers 1 and 2 in figure 3). By adjusting the angle between Alice's polarizers, we were able to fine tune the number of photons per pulse.

When Bob receives a single photon, there is a 25% chance that he will detect it. Therefore, for the average number of photons that Bob receives to be one, we want the proportion of times when a photon is detected at Bob's end to be 0.25.

$$P_{\bar{n}}(n > 0) = 1 - e^{-\bar{n}} = 0.25 \tag{9}$$

To calibrate our apparatus, we sent 5000 pulses from Alice to Bob and measured the number of times that Bob detected a photon on either channel 1 or channel 2. We found that an angle of 54° between polarizer 1 and polarizer 2 yielded 1250 ± 50 detected photons out of a total of 5000 sent photons, which gives the required ratio of 0.25.

B. Implementing B92

If B92 were to be implemented in real life, Alice would have one computer, Bob would have another, and they would communicate solely through a secure laser channel. In our case however, space and resources are limited, so we implemented B92 using a single computer, functioning as both Alice and Bob and sending information to itself, as shown in figure 3.

Essentially, the "Alice" half of the computer randomly generates a binary string; in practice, this is actually a succession of 8-bit sequences. For each bit, Alice pulses an infrared laser to generate packets of photons, with each photon in a packet encoding that one bit. The packets are passed through a pair of cross polarizers in order to reduce the average number of photons per packet, as described in the previous section. Next, the photon packets are conditionally polarized using a Pockels cell, which polarizes them in two passes (22.5° per pass). Finally, Alice passes the packet through a 10^{-9} attenuator so that only a single photon remains. She then "sends" this photon to Bob.

When Bob receives the photon, he passes it though a beam splitter which either sends the photon to a measurement device polarized 90° or to one polarized 315°. This effectively chooses at random a polarization basis on which to measure the photon. In reality, our polar-

ized measurement devices are polarized lenses followed by photon detectors. In our experimental set up, we called channel 1 the 90° lens (binary 0) and channel 2 the 315° lens (binary 1).

In order to measure the relationship between the post-error-correction error rate, p_{on}, and the original error rate, p_0, it was necessary to artificially introduce errors into our system. We achieved this by tuning Bob's polarized lenses so that they weren't completely orthogonal. The less orthogonal his lenses were, the higher the error rate. Conveniently, we found that p_0 varied linearly with $90 - \phi$, where ϕ is the angle between Bob's polarizers.

C. Implementing the Winnow Algorithm

Due to the fact that our QKD algorithm was implemented on a single computer, our implementation of the Winnow algorithm differed slightly from what is described in the "Methods" section. Unlike in a legitimate B92 set-up, where Alice's and Bob's computers could not share information aside from the secure channel, we were able to immediately compare the received data with the sent data. This allowed us to cheat a little bit in computing the channel efficiency. For each transmission of N bytes, we had a running tally of a variable called wastedBits, which kept track of every non-useful bit that was sent. Instead of sending the entire 12-bit syndrome each time, we simply transmitted the raw bytes and then checked the parity afterwards. There were two cases:

1. If the parity matched, then 1 was added to wastedBits (indicated the single parity bit)

2. If the parity didn't match, then the 4-bit abbreviated syndrome was computed (the orange bits in figure 2), and 4 was added to wastedBits

Every time that the parity didn't match, our algorithm corrected the error identified by the syndrome. Then each 8-bit block was checked against the byte that Alice originally sent. For each transmission, our algorithm kept a running tally of the number of erroneous bits. At the end, we divided this number by the number of useful bits sent (total bits sent minus wasted bits) to get our error rate. To get our channel capacity, we divided the number of useful bits sent by the number of total bits sent.

III. RESULTS

p_0 is the error rate without the Winnow error correction algorithm running, while p_{on} is the error rate *with* Winnow turned on. Both are calculated the same way:

$$p_{on/0} = \frac{\text{no. errors}}{\text{no. useful bits}} \tag{10}$$

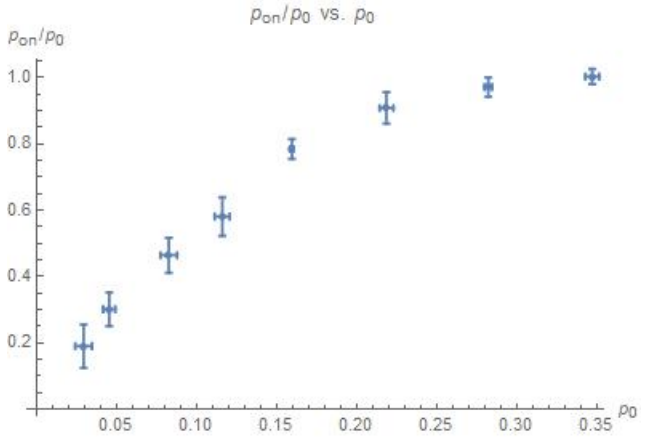

FIG. 4. Showing the ratio between the error rate after error correction (p_{on}) and the background error rate (p_0), versus the background error rate. The last data point ($0.347 \pm 0.004, 1.00 \pm 0.02$) is the lowest (and only) value of p_0 which we measured for which the Winnow correction algorithm increased the number of errors over the background error rate. This is in line with the results in Fig. 1. in Buttler et. al.[1]

For each data point, We measured p_0 five times, so the uncertainty in p_0 was calculated as the standard deviation of these five values. Similarly, the uncertainty in p_{on} is the standard deviation of the five measured values of p_{on}. Since we did not measure p_{on} and p_0 on the same tests, the uncertainty in p_{on}/p_0 is found by propagating the error of each p_{on} and p_0 in quadrature.

$$\frac{\delta(\frac{p_{on}}{p_0})}{\frac{p_{on}}{p_0}} = \sqrt{(\frac{\delta p_{on}}{p_{on}})^2 + (\frac{\delta p_0}{p_0})^2} \tag{11}$$

where p_0 and p_{on} are the average of the five measured values and δp_0 and δp_{on} are the standard deviations of these five measured values.

The expected functional form of the relationship between p_{on} and p_0 is not known, so there is little use in fitting a curve through these data points. In Buttler et. al.[1] a numerical approach is used to plot the theoretical relationship between p_{on}/p_0 and p_0 for a range of values (Fig. 1 in that paper). To verify that our results are consistent with the theory, we have plotted in Fig 5 the theoretical relationship between p_{on}/p_0 and p_0 and the data points that we measured. We can see that the theoretical curve passes through all of the error bars, indicating that our data supports the theoretical predictions of Buttler et. al.

We also measured the relationship between the channel efficiency, C, and the underlying error rate (p_0). We defined channel efficiency as

$$C = 1 - \frac{\text{no. bits discarded}}{\text{total no. bits sent}} \tag{12}$$

where no. bits discarded is the number of bits discarded by the error correction algorithm, and the remain-

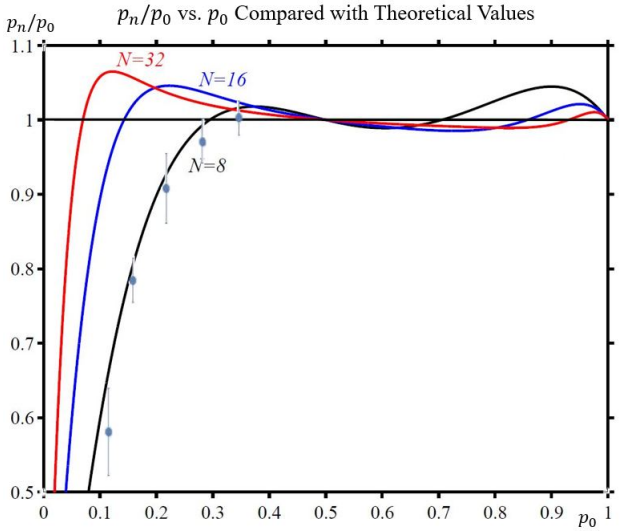

FIG. 5. Experimental measurements plotted alongside the theoretical curve of p_{on}/p_0 against p_0. The black $N = 8$ curve is the relevant comparison for our experiment. As Buttler et. al.[1] only plotted this relationship for $p_{on}/p_0 > 0.5$, only part of our data is shown. We can see that the theoretical curve passes through all of our measurement error bars, indicating that the experimental data is in line with theoretical predictions, and the chi-squared test would yield $\chi^2 \approx 1$. Since we don't know the functional form of the theoretical curve, we cannot precisely perform the chi-squared test for this fit.

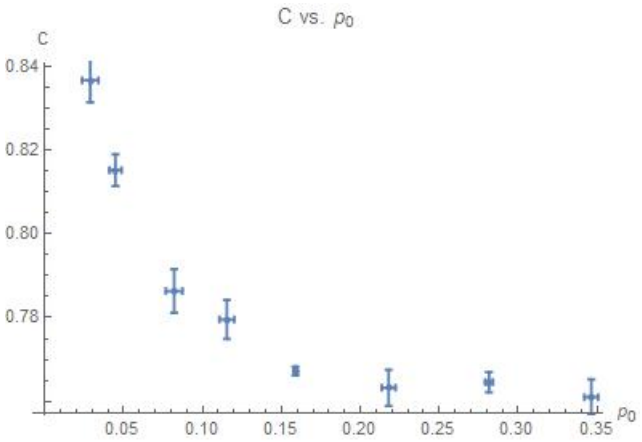

FIG. 6. Showing the relationship between the channel efficiency and the underlying error rate p_0. As expected, the higher the underlying error rate, the greater the number of bits discarded by the error correction algorithm, and so the lower the channel efficiency.

ing bits may be either correct or in error. This value is the same as the wastedBits variable described in section 2.C, "Implementing the Winnow Algorithm." It is useful to compare figure 4 and figure 6 to determine simultaneously how accurate and how efficient the Winnow algorithm makes our channel. The error bars are simply calculated as the standard deviation of the five measured values of C and p_0.

IV. CONCLUSION

The data presented here validate the theoretical predictions of the Winnow paper. Our measured error rate, shown in figure 4, matches the data presented by Buttler et al. within error bars. In figure 5, we overlay our data onto the Winnow data and see that they closely align for N = 8, which was the size of data that we tested with. To further verify the predictions of the Winnow paper, future researchers might run our algorithm with N = 16 and N = 32 to test those curves as well. They also might generate artificial error rates p_0 exceeding 0.35 so that the remainder of the N = 8 curve can be verified. It is important to note that although the error correction techniques presented here remove the majority of errors, an encryption free must be completely free of errors to be of any use. In order to achieve that, one can repeatedly shuffle the bits that are sent and reapply the Winnow algorithm. One can also adjust the block size (to N = 16 or N = 32) and reapply Winnow to obtain an error-free key.

In addition to measuring the error rates achieved by Winnow, we measured the relationship between error rate and channel efficiency, which we defined as the ratio of useful information sent to total information sent. We found that as the underlying error rate p_0 increases, the channel efficiency C drops sharply from 0.85 before leveling off at 0.77. However, this metric is slightly misleading, because when calculating the channel efficiency, the algorithm does not distinguish between useful bits that are correct and useful bits that are wrong. This as well suggests a further point of research.

V. REFERENCES

[1] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. Nickel, C. Donahue, and C. G. Peterson, "Fast, efficient error reconciliation for quantum cryptography," Physical Review A **67**, 052303 (2003).

[2] C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," Physical review letters **68**, 3121 (1992).

[3] C. E. Shannon, "A mathematical theory of communication," Bell system technical journal **27**, 379–423 (1948).

[4] C. E. Shannon, "Communication theory of secrecy systems," Bell system technical journal **28**, 656–715 (1949).

[5] C. BENNETT, "Quantum crytography," in *Proc. IEEE Int. Conf. Computers, Systems, and Signal Processing, Bangalore, India, 1984* (1984) pp. 175–179.