An Accurate Analysis of the BINARY Information Reconciliation Protocol by Generating Functions

Sean Seet
DSO National Laboratories
Singapore S118230
Email: flamingarrow13@gmail.com

Ruth Ng Ii-Yung
University of Chicago
Chicago, IL 60637
Email: ruthfrancisng@uchicago.edu
(This research was done while the author was at DSO National Laboratories)

Khoongming Khoo
DSO National Laboratories
Singapore S118230
Email: kkhoongm@dso.org.sg

***Abstract*—This Paper is eligible for the student paper award. Information Reconciliation (IR) protocols, which achieve error correction of shared secrets by public discussion, is an important process in Quantum Key Distribution (QKD). We provide an analysis of Brassard's BINARY and CASCADE IR protocols, two protocols commonly used in QKD. Using generating functions, we give an accurate result on BINARY. We derive the error probability distribution at each pass, which allows us to compute the decoding error probability and the number of "leaked" bits; two quantities crucial in the proof of security for QKD. We then corroborate the probability distribution computed by our formulas with actual simulation results. Finally we show that our formulas give better estimate for the decoding error probability of BINARY than the upper bound derived by Brassard for CASCADE. Because CASCADE should have better decoding performance than BINARY, this shows that Brassard's estimate of CASCADE may be too loose and can be improved. Our accurate formulas for BINARY can also be used as a basis on which to derive more accurate formulas for CASCADE.**

I. Introduction

Quantum Key Distribution (QKD) is an important technique to establish secret keys for secure communications. Its advantage over traditional cryptographic key distribution scheme is that QKD offers unconditional security while conventional schemes can only offer computational security.

In QKD, Alice and Bob first performs a photon exchange to share a secret string, called the raw key. However, there will be errors in the raw key they share because of:

1) Quantum channel noise.
2) Quantum eavesdropping by the adversary Eve.

Thus we need Information Reconciliation (IR), a 2-party error correction protocol, to correct the raw key through exchanging parity bits by public discussion. Because the adversary will learn some information of the raw key through quantum eavesdropping and observing the public exchange of parity bits, privacy amplification is used to compress the corrected raw key to a shorter final key to remove the adversary's knowledge.

The IR protocol used in the first practical instantiation of QKD is the BINARY IR protocol in [1]. It chops up a secret string into blocks, and perform binary search on each block to search out an error bit. The secret string is then permuted and this binary search error correction is performed over several passes until all error bits are corrected. Later, Brassard [3] introduced an improved version of BINARY, called CASCADE, which does backtracking to correct errors that was missed in binary search error correction of the earlier passes.

The CASCADE IR protocol is commonly studied and implemented because it corrects all errors by revealing the least number of parity bits (close to the theoretical limit), e.g. see [2], [3], [4], [5]. To analyze the security of a QKD system that uses the CASCADE IR protocol, we need to (e.g. see [6], [7]):

1) Measure the information leak of the raw key because of the exchange of parity bits.
2) Measure the decoding error probability, which is the probability that not all errors are corrected after completion of the IR protocol.

Both these values can be computed if we can predict the probability distribution of the number of error bits accurately. Till now the published results are either experimental [2], [4] or gives a loose estimate/bound [3].

Because the CASCADE protocol is built upon the BINARY IR protocol, see [3, Section 7.1]. One way to advance research in this direction is to first derive an accurate formula for the probability distribution of the BINARY IR protocol. We achieve this in Section IV by an innovative application of the technique of generating functions. The accuracy of our formula is corroborated by matching it with simulation results of the BINARY protocol in Section V. From the probability distribution, we can derive both the information leak of the raw key and the decoding error probability.

In particular, we point out in Section VII that the decoding error probability of BINARY that we derived is already better (smaller) than the upper bound derived by Brassard for CASCADE in [3]. Keeping in mind that the CASCADE protocol is an improved version of BINARY, it should have a better (smaller) decoding error probability. Thus, the bound derived by Brassard may be too loose and can be improved. Our accurate formula for BINARY can be a basis on which to build further research to derive more accurate formula for CASCADE.

II. A Description of the BINARY Information Reconciliation Protocol

In this section, we give a description of the `BINARY` IR Protocol to facilitate the analysis in Section IV later. The `BINARY` IR Protocol divides a raw key of length n into blocks of size k_1. Then Alice sends the parity of each block, defined as the modulo 2 sum of the bits of each block, to Bob.

If the parity of the corresponding block of Bob is different, then there is an error bit in that block and he will inform Alice of it. Alice will send the parity of the first half of the block to Bob, from which he can determine that the error lies in the first half if their parity differs, or the second half if their parity matches. This process is iteratively applied to successive halves of the block until the position of the error bit is narrowed down by a binary search.

However, note that not all errors can be corrected by one pass of the above process. This is because any block with an even number of errors will be bypassed after an exchange of 1 parity bit. And any block with an odd number of errors will only have one error corrected after an exchange of $\log_2(k_1)+1$ parity bits.

Thus `BINARY` implements a multi-pass binary search correction as follows: After the completion of pass i, a random permutation is applied at pass $i+1$ to the raw key to redistribute the positions of the error bits. Then the string is divided into blocks of length $k_{i+1} = 2k_i$ and the binary search error correction is again applied to each block. With respect to an error probability of p, the designer will try to choose the number of passes such that all error bits are corrected at the completion of the protocol.

For simplicity of analysis, we assume both n and k_i are powers of 2.

III. Definitions

We define the following terms (to be used in this paper) as follows:

- n is the length of the key.
- k_i is the block size at pass i.
- p is the probability that a specific bit will be incorrectly transmitted in the quantum channel.
- $\Delta_i(j-y \mid j)$ to be the probability that on the i^{th} pass, $j-y$ errors are corrected conditioned on the number of errors being j.
- $P_i(y)$ is the probability of there being y errors on the i^{th} pass.

IV. An Accurate Analysis of BINARY

Theorem 1. *Let $P_i(y)$ be the probability that there are y errors left in the raw key after pass i of the `BINARY` IR protocol. Then the initial probability distribution before IR is given by:*

$$P_0(y) = \binom{n}{y} p^y (1-p)^{n-y}$$

and for $i \geq 1$, the probability distribution after correction in pass i is given by:

$$P_i(y) = \sum_{j=y}^{y+\frac{n}{k_i}} P_{i-1}(j)\Delta_i(j-y \mid j)$$

Proof. The initial error distribution $P_0(y)$ is based on the binomial distribution $Binomial(n,p)$ with n independent bits where each of them has probability p of having an error.

Now we derive the probability distribution for $P_i(y)$ where $i \geq 1$. We call blocks with an odd number of error bits "odd-parity", and blocks with an even number of error bits "even-parity". Note that at most one error is corrected in each block, and this happens only in blocks with odd parity. Therefore, at most $\frac{n}{k_i}$ errors can be corrected at the i^{th} pass.

For $i \geq 1$ we can consider this as $\frac{n}{k_i}$ different cases, depending on the number of errors that were corrected at this pass. So, the chance that there are j initial errors and that y errors remain after the i^{th} pass is $P_{i-1}(j)\Delta_i(j-y \mid j)$. We sum over all cases and arrive at $P_i(y) = \sum_{j=y}^{y+\frac{n}{k_i}} P_{i-1}(j)\Delta_i(j-y \mid j)$. □

With this, we are left only to consider the calculation of $\Delta_i(j-y \mid j)$.

Theorem 2. *The quantity $\Delta_i(j-y \mid j)$, needed for the computation of Theorem 1, is given by:*

$$\Delta_i(j-y \mid j) = \frac{\binom{\frac{n}{k_i}}{j-y}}{\binom{n}{j}} \times C_{i,j,y}$$

Where $C_{i,j,y}$ = Coefficient of x^j in

$$\left(\frac{(1+x)^{k_i} - (1-x)^{k_i}}{2}\right)^{j-y}\left(\frac{(1+x)^{k_i} + (1-x)^{k_i}}{2}\right)^{\frac{n}{k_i}-(j-y)}$$

Proof. Notice that the event where $j-y$ errors are corrected and y errors remain is equivalent to the event where $j-y$ blocks have an odd-number of error-bits (which we will refer to as being of "odd-parity") and the remaining $\frac{n}{k_i}-(j-y)$ blocks to have 0 or an even number of errors (which we will refer to as being of "even-parity").

Consider, therefore, the following generating function:

$$\left(\binom{k_i}{1}x + \binom{k_i}{3}x^3 + \binom{k_i}{5}x^5 \ldots \binom{k_i}{k_i-1}x^{k_i-1}\right)^{(j-y)} \times$$
$$\left(\binom{k_i}{0} + \binom{k_i}{2}x^2 + \binom{k_i}{4}x^4 \ldots \binom{k_i}{k_i}x^{k_i}\right)^{\frac{n}{k_i}-(j-y)}$$
$$= \left(\frac{(1+x)^{k_i}-(1-x)^{k_i}}{2}\right)^{j-y}\left(\frac{(1+x)^{k_i}+(1-x)^{k_i}}{2}\right)^{\frac{n}{k_i}-(j-y)}$$

The first product term gives the number of ways in which we can have $j-y$ blocks of odd parity, and the second product term gives the number of ways we can have $\frac{n}{k_i}-(j-y)$ blocks of even parity. The resultant polynomial in x therefore has the property that the index of x is the number of errors, and the coefficient is the total number of ways these errors can be distributed such that $j-y$ blocks have odd parity. Also, there are $\binom{\frac{n}{k_i}}{j-y}$ ways to arrange $j-y$ blocks of odd parity amongst the $\frac{n}{k_i}$ blocks. Finally note that there are $\binom{n}{j}$ ways to distribute j errors in a key of length n with no restrictions. Therefore, the probability that $j-y$ errors are solved at pass i is $\binom{\frac{n}{k_i}}{j-y} \cdot C_{i,j,y}$. From this, we have $\Delta_i(j-y \mid j) = \frac{\binom{\frac{n}{k_i}}{j-y}}{\binom{n}{j}} \cdot C_{i,j,y}$.

□

Notice that these equations are sufficient to find $P_i(0)$ given some i. Therefore, we can compute the decoding error probability $1 - P_i(0)$, the chance that on the i^{th} pass not all errors have been corrected. Notice that we can also compute the number of leaked bits. Define the number of leaked bits on the i^{th} pass to be L_i. Then we can proceed as follows:

Theorem 3. *Given $n, i, k_i, P_i(y), \Delta_i(j-y \mid j)$ as before, we have the expected number of leaked bits as follows:*

$$L_i = \sum_{j=0}^{\frac{n}{k_i}} \Big(P_i(j) \sum_{y=0}^{j} \big[\Delta_i(j-y \mid j)((j-y)(\log_2 k_i + 1) + (\tfrac{n}{k_i} - (j-y)))\big]\Big) + \sum_{j=\frac{n}{k_i}+1}^{n} \Big(P_i(j) \sum_{y=j-\frac{n}{k_i}}^{j} \big[\Delta_i(j-y \mid j)((j-y)(\log_2 k_i + 1) + (\tfrac{n}{k_i} - (j-y)))\big]\Big)$$

Proof. The total number of bits leaked on the i^{th} pass of BINARY given that there are $(j-y)$ odd parity blocks is $((j-y)(\log_2 k_i + 1) + (\frac{n}{k_i} - (j-y))$. This is because each odd parity block causes a total of $\log_2 k_i + 1$ parity bits to be shared, and each even parity block causes 1 parity bit to be shared. Therefore, the expected value of the number of leaked bits given j is $\sum_y \Big(\Delta_i(j-y \mid j)((j-y)(\log_2 k_i + 1) + (\frac{n}{k_i} - (j-y)))\Big)$. Then, in order to get an estimate of the expected value of the number of leaked bits, we multiply the probability of there being j errors $P_i(j)$ and the expected value of the number of leaked bits given that there are j errors and sum over all j: $L_i = \sum_j \Big(P_i(j) \sum_y \big[\Delta_i(j-y \mid j)((j-y)(\log_2 k_i + 1) + (\frac{n}{k_i} - (j-y)))\big]\Big)$. However, due to the constraint that $j - y \leq \frac{n}{k_i}$, we must use two summands.

□

Therefore on the i^{th} pass the total number of leaked bits is given by $\sum_{a=1}^{i} L_a$

V. Comparison of our Computation with Simulation of BINARY

To corroborate our results, we ran 10000 trials on `BINARY` with $k_1 = 16$, $n = 256$, $p = 0.03$ and matched this against the probability distribution $P_i(j)$ computed from Theorems 1 and 2. The computation is done on the mathematical software MAPLE, while the simulation of `BINARY` is programmed in C. Due to space restrictions, here we display a truncated table of comparison in Table I.

We repeated the comparison for $k_1 = 16$, $n = 2048$, $p = 0.03$ in Table II.

From both tables, we see that our derivation of the probability distribution of `BINARY` by Theorems 1 and 2 is corroborated by actual simulation results.

A point to note for the second simulation for $n = 2048$, the polynomials in Theorem 2 got too large and we need to truncate them to speed up the computation. We empirically determine the points at which truncation yields an accurate approximation. Therefore, for example, the exact polynomial in the calculation of $\Delta_2(4 \mid j)$ for $n = 2048$, $k_1 = 16 \Longrightarrow k_2 = 2 \times k_1 = 32$, $n/k_2 = 128$, is given by:

$$\left(\frac{(1+x)^{32}-(1-x)^{32}}{2}\right)^4 \left(\frac{(1+x)^{32}+(1-x)^{32}}{2}\right)^{124} = \left(\binom{32}{1}x + \binom{32}{3}x^3 + \binom{32}{5}x^5 + \cdots + \binom{32}{31}x^{31}\right)^4 \times \left(\binom{32}{0} + \binom{32}{2}x^2 + \binom{32}{4}x^4 + \cdots + \binom{32}{32}x^{32}\right)^{124}$$

But instead, we considered the product of 128 truncated polynomials:

$$\left(\binom{32}{1}x + \binom{32}{3}x^3 + \binom{32}{5}x^5 + \cdots + \binom{32}{9}x^9\right)^4 \times \left(\binom{32}{0} + \binom{32}{2}x^2 + \binom{32}{4}x^4 + \cdots + \binom{32}{10}x^{10}\right)^{124}$$

This can be done because $\Delta_2(4 \mid j)$ is given by the coefficient of x^j, which is obviously covered by the truncated expression for $j \leq 10$. Even for larger j, e.g. $\Delta_2(4 \mid 15)$ or $\Delta_2(4 \mid 20)$, the "bulk" of the coefficient of x^{15} or x^{20} is contributed by summing the cross-product terms of the 128 truncated polynomials.

Also, we can compare our theoretical expected value of the number of leaked bits L_i from Theorem 3 to the average number of leaked bits in each pass from 10000 trials of `BINARY` simulation.

We see that the theoretical calculation is very close to the experimentally determined value of L_i.

	Pass $i=1$	Pass $i=2$	Pass $i=3$	Pass $i=4$
$P_i(0)$	0.25533	0.68058	0.86172	0.91689
$P_i(2)$	0.35897	0.23196	0.10989	0.06616
$P_i(4)$	0.24146	0.06757	0.02286	0.01372
$P_i(6)$	0.10347	0.01608	0.00457	0.00269
$P_i(8)$	0.03174	0.00319	0.00081	0.00047
$P_i(0)$	0.24960	0.68040	0.86090	0.91700
$P_i(2)$	0.36690	0.23110	0.10870	0.06530
$P_i(4)$	0.23750	0.06630	0.02430	0.01390
$P_i(6)$	0.10310	0.01910	0.00500	0.00310
$P_i(8)$	0.03440	0.00230	0.00080	0.00070

TABLE I
BINARY Calculation (top) and Simulation (bottom): $k_1 = 16$, $n = 256$, $p = 0.03$

	Pass $i=1$	Pass $i=2$	Pass $i=3$	Pass $i=4$
$P_i(0)$	0.00002	0.08808	0.63320	0.92559
$P_i(2)$	0.00020	0.17874	0.23883	0.06221
$P_i(4)$	0.00114	0.21261	0.08548	0.00974
$P_i(6)$	0.00421	0.19004	0.02891	0.00192
$P_i(8)$	0.01166	0.14029	0.00937	0.00042
$P_i(0)$	0.00000	0.08700	0.63740	0.92750
$P_i(2)$	0.00010	0.17690	0.23460	0.05950
$P_i(4)$	0.00110	0.21610	0.08540	0.01030
$P_i(6)$	0.00360	0.19370	0.02840	0.00230
$P_i(8)$	0.01070	0.13500	0.01020	0.00020

TABLE II
BINARY Simulation (top) and Calculation (bottom): $k_1 = 16$, $n = 2048$, $p = 0.03$

	Experimental	Theory
L_1	36.1064	36.1109
L_2	16.9910	16.9480
L_3	7.0888	7.0955
L_4	3.0052	2.9708

TABLE III
THE EXPECTED AMOUNT OF LEAKED INFORMATION IN THE i^{th} PASS, $k_1 = 16, n = 256, p = 0.03$

	Experimental	Theory
L_1	289.2384	288.8842
L_2	141.6640	141.3399
L_3	59.8124	59.8092
L_4	14.4960	14.5381

TABLE IV
THE EXPECTED AMOUNT OF LEAKED INFORMATION IN THE i^{th} PASS, $k_1 = 16, n = 2048, p = 0.03$

VI. A DESCRIPTION OF THE CASCADE IR PROTOCOL

In this section, we give a description of the `CASCADE` IR protocol and an illustration of its difference from `BINARY` to facilitate further analysis in Section VII. The `CASCADE` Protocol is similar to `BINARY` except that in `CASCADE` there is an additional step known as back-tracking. Whenever errors are discovered in the i^{th} pass, $i > 1$, it is clear that it must have escaped notice in prior passes due to being one of an even number of errors in a block. In `CASCADE`, back-tracking describes the process of correcting known errors in previous passes in order to get blocks of odd parity on which the Binary Search can be performed. This process allows us to identify more errors in the same number of passes than `BINARY`. The exact mechanism of back-tracking used in `CASCADE` can be found in Section 7.1 of Brassard [3].

We compare the implementation of `BINARY` and `CASCADE` in the diagram below. The relative position of initial error bits e_1 to e_{10} within blocks are represented as they are permutated through three passes.

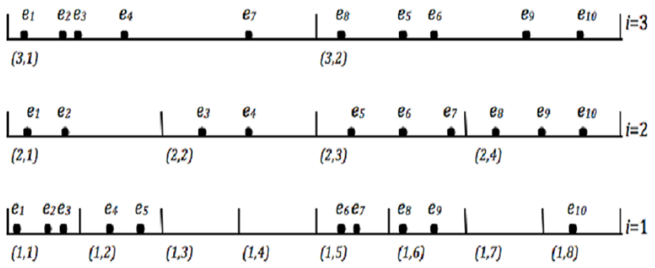
e1 e2 e3 e4 e7 e8 e5 e6 e9 e10 i=3
(3,1) (3,2)
e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 i=2
(2,1) (2,2) (2,3) (2,4)
e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 i=1
(1,1) (1,2) (1,3) (1,4) (1,5) (1,6) (1,7) (1,8)

Fig. 1. Example for Comparison of Protocols

Under `BINARY` and `CASCADE`, the error corrections and passes will be executed as follows (the tuple following an error indicates that error was attended to within the block indicated by the tuple):

`BINARY`

1) Pass 1
2) $e_1, (1,1)$
3) $e_{10}, (1,8)$
4) Pass 2
5) $e_2, (2,1)$
6) $e_5, (2,3)$
7) Pass 3
8) $e_7, (3,1)$
9) $e_9, (3,2)$

`CASCADE`

1) Pass 1
2) $e_1, (1,1)$
3) $e_{10}, (1,8)$
4) Pass 2
5) $e_2, (2,1)$
6) $e_3, (1,1)$
7) $e_4, (2,2)$
8) $e_5, (1,2)$
9) Pass 3
10) $e_7, (3,1)$
11) $e_6, (1,5)$
12) $e_9, (2,4)$
13) $e_8, (1,6)$

We see in this example that over three passes, only 6 out of 10 error bits are corrected in `BINARY` while all 10 error bits are corrected in `CASCADE`.

VII. COMPARISON OF DECODING ERROR PROBABILITY WITH BRASSARD'S BOUND [3]

A. Brassard's Upper Bound on Decoding Error Probability

In Brassard's analysis [3] of `CASCADE` with an error probability of p, he focuses on the number of errors in block v of pass 1 (this block is called K_v^1 in [3]). After error correction in pass i, he would backtrack the error bits through to the first pass to investigate the number of errors remaining in the original block K_v^1. He denotes the probability that there are $2j$ errors left in K_v^1 after pass i to be $\delta_i(j)$.

Proposition 1. *([3, Section 7.2]) In the `CASCADE` IR protocol with an error probability of p, the following inequality holds for $\delta_i(j)$:*

$$\delta_i(j) \le \frac{\delta_{i-1}(j)}{2} \le \frac{\delta_1(j)}{2^{i-1}} \text{ for } i, j > 0,$$

if the following two conditions hold:

$$k_1 p - \frac{(1-(1-2p)^{k_1})}{2} \le -\frac{\ln(1/2)}{2},$$

and

$$\sum_{r=j+1}^{k_1/2} \delta_1(r) \le \frac{1}{4}\delta_1(j) \text{ for all } j.$$

Using Proposition 1, we can compute the decoding error probability from $\delta_i(0)$ as follows.

From [3, Section 7.2], the probability that there are $2j$ errors in the block after pass 1 is given by:

$$\delta_1(j) = \binom{k_1}{2j} p^{2j}(1-p)^{k_1-2j} + \binom{k_1}{2j+1} p^{2j+1}(1-p)^{k_1-(2j+1)}$$

Then for $j > 0$, we compute $\delta_i(j) \le \frac{\delta_1(j)}{2^{i-1}}$ for all $j > 0$. From these, we compute:

$$\delta_i(0) = 1 - \sum_{j=1}^{k_1/2} \delta_i(j) \ge 1 - \sum_{j=1}^{k_1/2} \frac{\delta_1(j)}{2^{i-1}}.$$

Since $\delta_i(0)$ is the chance of no errors in block K_v^1. Assuming all the blocks are independent, the chance that there are no errors in all n/k_1 blocks is given by:

$$P_i(0) = \delta_i(0)^{n/k_1} \geq \left(1 - \sum_{j=1}^{k_1/2} \frac{\delta_1(j)}{2^{i-1}}\right)^{n/k_1}.$$

Note that the above expression subtracted from 1 would give an upper bound for the decoding error probability, given by $1 - P_i(0)$.

B. Comparison to Brassard

In this section, we compute $P_i(0)$ in BINARY, based on Theorems 1 and 2, and compare it to Brassard's lower bound for $P_i(0)$ in CASCADE, based on Proposition 1 and the subsequent discussion. It can be verified that Brassard's bound can be applied because the two conditions of Proposition 1 are satisfied for $k1 = 16$ and $p = 0.03$. I.e.

$$16 \times 0.03 - \frac{(1-(1-2\times 0.03)^{16})}{2} \leq -\frac{\ln(1/2)}{2},$$

and

$$\sum_{r=j+1}^{8} \delta_1(r) \leq \frac{1}{4}\delta_1(j) \text{ for all } j.$$

The results are summarized below:

Value	Our Calculation for BINARY	Brassard's Lower Bound for CASCADE
$P_1(0)$	0.25533	0.25533
$P_2(0)$	0.68058	0.51271
$P_3(0)$	0.86172	0.71854
$P_4(0)$	0.91688	0.84839

TABLE V
COMPARISON, $k_1 = 16$, $n = 256$, $p = 0.03$

Value	Our Calculation for BINARY	Brassard's Lower Bound for CASCADE
$P_1(0)$	0.00002	0.00002
$P_2(0)$	0.08808	0.00476
$P_3(0)$	0.63320	0.07106
$P_4(0)$	0.92559	0.26839
$P_5(0)$	0.98420	0.51894
$P_6(0)$	0.99492	0.72068
$P_7(0)$	0.99722	0.84902

TABLE VI
COMPARISON, $k_1 = 16$, $n = 2048$, $p = 0.03$

We see that our calculation of $P_i(0)$ in BINARY is much larger than Brassard's lower bound for $P_i(0)$ in CASCADE. Equivalently, our calculation gives a much smaller decoding error probability $1 - P_i(0)$ for BINARY than the upper bound deduced from Brassard for CASCADE.

However, from the description of CASCADE in Section VI and the subsequent example to illustrate its superior error correction capability over BINARY, it is obvious that CASCADE can correct many more errors for the same number of passes. Thus, CASCADE should have a much higher $P_i(0)$ or equivalently, a much smaller decoding error probability $1 - P_i(0)$, than BINARY.

Thus, we conclude that Brassard's bound may be too loose. One possible reason for this discrepancy is that our analysis focuses on the global distribution of error bits, while Brassard's analysis focuses on the error distribution within a block. Thus when we extrapolate Brassard's result to all blocks, the bound will not be as tight. Thus, our methods here may yield a useful analysis of the global distribution of error bits in the CASCADE IR protocol.

VIII. CONCLUSION

In our paper, we derived an accurate formula for the probability distribution of the number of errors in any one pass of BINARY, from which we can extract important information like the decoding error probability and the expected amount of leaked information. We also confirmed the validity of our results by comparing them to experimental data.

Moreover, we better Brassard's bound on CASCADE, which suggests we can build on our existing research to get a more accurate analysis of CASCADE. This would enable us to make our calculations more informative in practical decisions of how many passes should be carried out in a QKD system.

ACKNOWLEDGMENTS

We would like to thank Chu-Wee Lim of DSO National Laboratories for suggesting the use of generating functions and provided C implementations of BINARY for us to verify our results.

REFERENCES

[1] Bennet, C.H.; Bessette, F.; Brassard, G.; Salvail, L., Smolin J. *Experimental Quantum Cryptography*. Journal of Cryptology, Vol. 5, No. 1 pp. 3-28, 1992.

[2] Bellot, Patrick; Dang, Minh-Dung *BB84 Implementation and Computer Reality*. *RIVF 2009* RIVF, pp. 1-8, 2009.

[3] Brassard, Gilles; Salvail, Louis *Secret-Key Reconciliation by Public Discussion*. *EUROCRYPT '93* Workshop on the theory and application of cryptographic techniques on Advances in cryptology, pp. 410-423, Springer-Verlag, 1994

[4] Calver, Timothy *An Empirical Analysis of the Cascade Secret Key Reconciliation Protocol for Quantum Key Distribution*, Thesis submitted to Airforce Institute of Technology, available from http://www.dtic.mil/dtic/tr/fulltext/u2/a549804.pdf.

[5] Capraro, I.; Occhipinti, T. *Implementation of a Real Time High Level Protocol Software for Quantum Key Distribution*, *ICSPC 2007*, IEEE International Conference on Signal Processing and Communications, pp. 704-707, IEEE Press, 2007.

[6] Renato Renner, *Security of Quantum Key Distribution*, PhD Thesis submitted to Swiss Federal Institute of Technology Zurich, available from http://arxiv.org/pdf/quant-ph/0512258v2.pdf.

[7] Valerio Scarani, Renato Renner *Security Bounds for Quantum Cryptography with Finite Resources*, Proceedings of TQC2008, Lecture Notes in Computer Science 5106 (Springer Verlag, Berlin), pp. 83-95 (2008).