2000

Improvement of reconciliation for quantum key distribution

Keath Chen

Rochester Institute of Technology
Department of Computer Science

Improvement of Reconciliation for Quantum Key Distribution

by
Dr.[1] Keath Chen

A thesis, submitted to
the Faculty of the Department of Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Approved by:

Prof. Stanislaw Radziszowski
RIT

Prof. Edith Hemaspaandra
RIT

Dr. Harald Hempel
Institut für Informatik
Friedrich-Schiller-Universität
Jena, Germany

June, 28, 2000

[1] in Chemical Engineering, University of Rochester

Permission granted

Title of thesis: Improvement of Reconciliation for Quantum Key Distribution

I, Keath Chen, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: Aug., 18, 2000 Signature of Author: _____

Improvement of Reconciliation for Quantum Key Distribution

By Keath Chen

June 28, 2000

Abstract

Quantum Key Distribution is meant to be an ultimate computer security system that will not need upgrade/overhaul from time to time. QKD allows the generation of long key length on demand. QKD, coupled with the unconditional secure "one-time pad" encryption system, will be unbreakable for eavesdropper with infinite resources. Numerous experimental QKD prototypes have demonstrated that QKD is likely to be a reality before quantum computer.

One of the steps in a Quantum Key Distribution protocol is to remove the transmission errors from quantum communication, which typically has a high error rate (one percent or higher). An interactive error control method is utilized in the reconciliation of QKD. This procedure divides transmitted bits into blocks. The size of the block is chosen so that the chance of having multiple error bits in one block is small. By checking parity and doing interactive BINARY search when a parity error is found, error bits can be located and removed. Some error bits escape the detection from the first pass. By repeating this procedure several times, each time randomly dividing bits into blocks, most error bits can be detected and removed. Each parity check means loss of one bit. The goal is to minimize the number of parity checks to locate all (or most) errors and to have a high reliability that the remaining bits have very small residue error rate.

Brassard and Salvail devised a better error control procedure in 1993. By keeping track of block parity information from pass to pass, which they called CASCADE, the number of parity checks per error bit is reduced. They observed no errors left after four passes from 10 simulation runs. However, they offered upper bound of residue error rate that was much higher than observed from simulation. Therefore, there is a need to find the optimal block size and to improve the residue error rate analysis of this procedure.

In this work, improvement in reconciliation procedure is obtained in throughput (i.e. the number of input bits minus the number of parity checks) and in lower residue error rate. Throughput is doubled at high input error rate and less throughput gain at low input error rate. Three most important factors that enhance the system throughput are:

1) Using larger block size to fully utilize the power of CASCADE method.
2) Avoiding any pair of bits to stay in the same block of any later pass, which allows the use of three passes only (instead of four passes in the BS procedure).
3) Some bits are already known to be lost from the current pass, so I avoid carrying these bits to the later pass.

The residue error rate at the end of the second pass (and the third pass) for small initial block size is derived. The residue error rate at the end of the third pass is found much lower than the upper bound given by the previous workers. This residue error rate is found to be a function of input size and the block size.

This error control method has many similar features and difficulties to the recent development of "turbo-code" in the error control-coding field.

1 Introduction

Computer security is becoming increasingly important in the communication industry. The consumer confidence toward the system security is essential for the growth of emerging e-business; financial institutions depend heavily on the security of the computer; hackers constantly challenge the Internet security. Federal government has recently mounted efforts to boost computer security. Encryption schemes are the essential pieces of these efforts. In order to communicate privately, sender and receiver share a short secret key. Encryption schemes use a one-way function to spread the short key over a long plaintext. One-way functions make it very hard for eavesdroppers to calculate the key from plaintext and ciphertext. These schemes rely on the computational complexity such as factoring of large integers or computing discrete logarithms. As computation power, computing theory and algorithms improve, the security of some cryptographic systems will become obsolete, diminish, or need overhaul (e.g., increasing key length or number of rounds); all can be very costly. Recent overhaul from Digital Encryption Standard (DES) to Advance Encryption Standard (AES) is one example [AES].

"One-time pad" is an unconditionally secure encryption scheme that does not rely on assumptions of computational complexity. The key length in a one-time pad has to be the same length as the plaintext, converting the security problem to a key distribution problem.

Except one-time pad, all encryption systems of today will be at risk by the development of quantum computer. Shor's algorithm [S94] of factoring large integers in polynomial time caught much attention as people's fortune may be significantly altered due to computer security. This finding leads to numerous daunting researches toward the long-term goal of a quantum computer. Yet an implementation of a quantum computer may have a disastrous effect on the current security and economic systems.

Quantum encryption may be the best defense against eavesdroppers with supreme technologies and vast resources (i.e. quantum computers). Should quantum computers prove impossible to build, QKD still offer a cryptographic system that will not need system overhaul from decade to decade. (Such as AES or lengthening the key of RSA).

A quantum encryption takes advantage of the fact that eavesdropping activities will lead to disturbance to the transmitted signals that can be detected by the sender and receiver. This branch of activity lead to the development of the quantum encryption protocol and the quantum key distribution (QKD) protocol. The latter already has some small-scale experiment demonstrations, and is the topic of this research study.

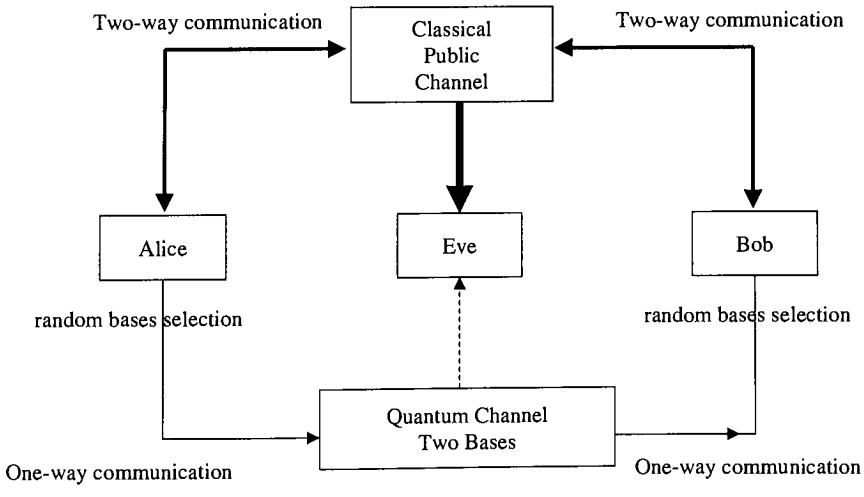
1.1 Quantum Key Distribution

QKD system employs two communication channels (Fig. 1): a one-way quantum channel between the sender (Alice) and the receiver (Bob) and a classical two-way communication channel. The quantum channel is where the security of the system lies. In

the case of BB84 QKD protocol (more about that in section 1.3), two sets of bases are used in the quantum channel. The transmission is meaningful only when Alice and Bob choose the same bases. The key is sent from Alice to Bob in the quantum channel as quantum states. Eve may eavesdrop in the quantum channel. To counter the eavesdropping, Alice randomly chooses quantum bases to encode each key bit as quantum states. Bob measures the quantum signals in either one of the two bases by his independent random choice. Eve has to guess which bases Alice used and her on-line measurement will irreversibly disturb ~50% of the quantum states she chooses to measure. This disturbance introduces ~25% errors to the later measurements by the receiver. Alice and Bob use the classic communication channel to handle various administrative details, including finding out which quantum signals are sent and measured in the same bases, detection of errors caused by eavesdropper and removal of errors from the quantum transmission. The underlying physics that reveals eavesdropping activity, eavesdropping strategies, and counter strategies can be found in numerous literature (see Rieffel and Polak [RP] and Lomonaco [L99], the latter is a good and easy introduction).

The communication between Alice and Bob in the classical channel, encrypted or not, is totally open to Eve, as Eve is allowed to have superior technologies and unlimited resources. Eve cannot be allowed to alter the communication in the classical channel, though. To avoid impersonation by Eve, an unconditionally secure authentication scheme is needed in the classic channel and Alice and Bob need to share a short secret key for authentication purposes. Thus, QKD is actually a key expansion scheme. This is a generic limitation as it is proven impossible to generate an unconditionally secure key without prior shared secret key. An efficient authentication algorithm is provided by Wegman and Carter [WC] [CW].

Figure 1: A one-way quantum channel and a two-way classical channel are needed for QKD.



Wiesner was the first to propose utilizing quantum entanglement for security in 1970. However, his paper on photon storage as identification for digital money was rejected from publication until 1983 [W83]. Bennett and Brassard [BB84] realized the quantum state of photons would be better used for transmission instead of storage. They published the first QKD protocol, the famous BB84 protocol. They subsequently filled in the reconciliation and privacy amplification procedures for this scheme [BBBSS, BBR85, BBR88, and BBCM] and presented an experimental demonstration [BBBSS]. BB84 has attracted most of the followers both in experimental demonstration and in theoretical analysis. Two other protocols were proposed by Ekert [E91] using EPR pairs and by Bennett [B92] using two non-orthogonal quantum states (instead of four states in BB84). A modification of BB84 using six states to increase the security at the cost of throughput was proposed by Bruβ [B98].

Two main methods were used in most of quantum key experiments: photon polarization and photon's phases. The first experiment by Bennett and Brassard was performed at the distance of 32 cm [BB89, BBBSS]. Marand and Townsend [MT] reported transmission over 30 km of commercial optical fiber in the lab. Muller, Zbinden, and Gisin [MZG] reported transmission of 22.7 km over optical cable under Lake Geneva. A single photon transmitted over a 48 km optical fiber network was reported by Hughes et. al. in 1999 [HMP]. Outdoor transmission in nighttime of almost 1 km was reported by Buttler et. al. [BHKLL].

The technical challenges for the QKD are single photon light sources, low attenuation optical fiber, and high efficiency single photon detectors with higher frequency capability. Numerous research efforts are in progress to produce single photon sources (e.g. Brunel et. al. [BLTO] and Kim et. al. [KBKY]). The loss of photon signals over a long distance is another limitation. While current commercially available optical fiber has attenuation about 0.2 dB/km, exotic optical fiber with 0.01 dB/km has been found. The latter will allow photon transmission up to 1000 km for 10% of the photons available at the receiving end. The system throughput and operating range will improve from the above research efforts. Another challenging practical problem for QKD is the development of multi-user network.

I will briefly introduce the aspects of BB84 protocol that are related to this study, recount one system performance analysis by Slutsky et. al. [SSMRF], and then the details of reconciliation procedure in [BBBSS] and the improvement procedure in [BS].

1.2 Quantum States

Before explaining BB84 protocol, I'll give a short introduction to the quantum states. In the classic information theory, a Shannon bit is either 1 or 0, but by no means both at the same time. On the other hand, a quantum bit, or qubit, can be both 1 and 0 at the same time. This is the so-called superposition of states.

Using light polarization as an example, a photon can be in a vertically polarized state $|\updownarrow\rangle$, we assign a label "1" to this state. It can be in a horizontally polarized state $|\leftrightarrow\rangle$, we

assign a label "0" to this state. Or, it can be in a superposition of these states. In this case, we interpret its state as representing both 0 and 1 at the same time. The vertically and horizontally polarized states is a set of basis for QKD, which is called rectilinear basis. After a measurement, a photon state collapses to the measurement state. For example, the vertically polarized sunglasses eliminate glare by letting through only vertically polarized light, while filtering out the horizontally polarized light. The sunglasses are measurement devices in this sense, and the light polarization collapses any photon state to the vertical polarization state after this measurement. A vertically or horizontally polarized photon can pass through a vertically (or horizontally) polarized measurement filter and register a "1" or "0" deterministically in a single-photon-detector.

Another set of basis states is $|\nearrow\rangle$ (polarized at 45°) and $|\nwarrow\rangle$ (polarized at 135°), which is called diagonal basis. Any photon state can be represented by a superposition of the diagonal basis as well as by a superposition of the rectilinear basis. Light polarized in $|\nearrow\rangle$ (or in $|\nwarrow\rangle$) state has equal probability being in 1 or 0 state if measured in the rectilinear basis. Likewise, light polarized in $|\updownarrow\rangle$ (or in $|\leftrightarrow\rangle$) state has equal probability being in 1 or 0 state if measured in the diagonal basis. If the photon signals are prepared by the sender in one basis (e.g. rectilinear) and measured by the receiver in another basis (diagonal), then the result of the measurement will not relate to information from the sender.

Photons can also be circularly polarized: this set of basis $|\circlearrowleft\rangle$ and $|\circlearrowright\rangle$ is called circular basis. Any two of these three sets of bases can be used in BB84 protocol.

1.3 BB84 protocol

The BB84 protocol requires two sets of orthogonal quantum states. Using photon polarization example, Alice needs to have the capability of preparing photons in any of the two orthogonal quantum states and Bob have the capability of measuring photon signals in both bases. There are four stages for this protocol.

Stage 1: communication over quantum channel

Alice prepares two random sequences of bits. One of the random bit sequences decides which quantum basis to use (basis bits, the first row in Fig. 2a). Another binary bit sequence will be used to construct secret key (key bits, the second row in Fig. 2a). The polarization state of each photon Alice sends is determined by the key bit and the basis bit (the third row in Fig. 2a). Bob prepares independently another random string of bits (basis bits, the first row of Bob's action in Fig. 2a) that determine which basis he will use to measure each incoming photon (the measurement result, the second row of Bob's action in Fig. 2a). Because of the properties of quantum states, when Bob uses the same basis as Alice's, Bob measurement result is the same as Alice's key bit (except transmission error). This is shown in boldface basis selection in the first row from Alice and Bob's action. When Bob uses different basis from Alice's, then Bob's measurement results are uncorrelated to Alice's key bits, or ~50% of Bob's result is the same as Alice's key. This is shown in light face basis selection in the first row of Alice and Bob's action.

Stage 2: communication over classical channel

Phase 1: raw key extraction

After the quantum transmission, Alice and Bob compare their basis bits, but not the key bits. They keep the key bits only when the basis bits were the same. This is called raw key (the bit sequence in the third row of Bob's action in Fig. 2a). About 50% of the quantum transmission will be discarded because of the independent random basis selection by Alice and Bob (the white cells in the third row of Bob's action in Fig. 2a).

Figure 2: BB84 quantum communication example --
Alice and Bob agree to use the rectilinear (⊞) and the diagonal (⊠) bases. |↕> represents "1" and |↔> "0" in rectilinear basis. |↖> represents "1" and |↗> "0" in diagonal basis.
a): In the absence of eavesdropping activity: Photon polarization sent by Alice is decided by the message bit and the basis bit. Bob's independent random choice of basis in measurement resulted in loss of 50% of quantum transmission when the bases do not match.

Alice	random bases selection	⊞	⊠	⊠	⊠	⊞	⊠	⊞	⊠	⊞	⊠	⊠	⊞
	random message	1	0	0	1	1	0	0	1	0	1	0	1
	photon polarization sent	↕	↗	↗	↖	↕	↗	↔	↖	↔	↖	↗	↕

Bob	random bases selection	⊠	⊠	⊞	⊠	⊞	⊠	⊞	⊞	⊞	⊞	⊠	⊠
	measurement result	1	0	1	1	1	0	0	0	0	0	0	1
	raw key		0		1	1	0	0		0		0	

b): In the presence of eavesdropping activity: Intercept/resend attack by Eve alters the photon polarization states as Eve has to make random choice of basis before Bob. The presence of Eve can be detected when Alice and Bob compared a substring and found 25% errors in raw key (boldface digits in the last row).

Alice	random bases selection	⊞	⊠	⊠	⊠	⊞	⊠	⊞	⊠	⊞	⊠	⊠	⊞
	random message	1	0	0	1	1	0	0	1	0	1	0	1
	photon polarization sent	↕	↗	↗	↖	↕	↗	↔	↖	↔	↖	↗	↕

Eve	random bases selection	⊠	⊞	⊞	⊠	⊞	⊞	⊠	⊠	⊞	⊞	⊠	⊠
	measurement result	1	0	1	1	1	1	0	1	0	0	0	1
	photon polarization sent	↖	↔	↕	↖	↕	↕	↗	↖	↔	↔	↗	↖

Bob	random bases selection	⊠	⊠	⊞	⊠	⊞	⊠	⊞	⊞	⊞	⊞	⊠	⊠
	measurement result	1	0	1	1	1	1	1	0	0	0	0	1
	raw key		0		1	1	**1**	**1**		0		0	

Phase 2: Error estimation

Over the classic channel, Alice and Bob compare a random subset of the raw key to estimate the error rate p, and then delete this subset from their raw keys to produce their **tentative final key**. If the error rate is within certain limit, they accept it as the physical limitation of their quantum devices, such as transmission errors, detection errors, etc. If the error rate is too high, they assume Eve is listening in on the quantum channel, and they discard everything and start over again.

Phase 3: Reconciliation or error correction

The reconciliation procedure will be described in more detail in section 1.4. Alice and Bob use this interactive parity check procedure to locate error bits. Communication of parity checks over the classical channel leaks information to Eve. Reconciliation leads to information leakage many times of the error rate. The goal is to minimize the number of parity checks while keeping the residue error rate low (high reliability). The **reconciled key** is derived after discarding leakage due to parity checks.

Phase 4: Privacy Amplification

Alice and Bob now have a common reconciled key that is partially known to Eve. Alice and Bob take into account various information leakage due to various form of Eve's attacks and, in addition, add a security margin S. They choose a random hash function based on these parameters to ensure Eve's average information about the secret key is $2^{-S}/\ln 2$ bits. The **final key** is derived after privacy amplification.

The most likely form of attacks is intercept/resend and beam splitting. The intercept/resend attack is a measurement on-line, which will cause disturbance of the quantum channel and is illustrated in Fig. 2b. Alice's action is the same as in Fig. 2a. Eve has to guess the measurement basis before Bob (basis bits, the first row of Eve's action in Fig. 2b). Since a measurement of a quantum state collapses it to the measurement state, Eve's measurement alters ~50% of the photon polarization states (compare the photon states sent by Alice and those by Eve, the last row of Alice and Eve's action). Subsequently, half of Bob's measurement of these altered photon states will be in error, or a 25% error rate in Bob's raw key (the boldfaced digits in the last row of Bob's action in Fig. 2b). This is why Alice and Bob can catch Eve's activity in phase 2 of stage 2. Eve's counter move is to measure a fraction of the quantum transmission so that the overall disturbance will be below the error tolerance in the quantum channel. Eve will then use an error-free quantum channel to resend the quantum states to Bob. There are many other ways Eve can try to minimize disturbance and to maximize information gain. The maximum information leakage due to Eve's attack is an open research topic. Advance analyses (e.g. Fuchs et. al. [FGGNP]) in eavesdropping strategies using future technologies continuously enlarge the bound of Eve's information. Such advance analyses are beyond the scope of this introduction. Because of the intercept/resend strategy, a secure QKD protocol has to assume all errors are the result of eavesdropping, and allows a conservative estimate of Eve's information base on the error rate p estimated

from phase 2 of stage 2. This is dealt with in the privacy amplification step and will be discussed further in section 1.5.

The beam splitting attack works on multi-photon states. The distribution of states from a common laser light source is typically in Poisson distribution. Eve splits the multiple photon states sent by Alice, passes one to Bob, and stores the rest. After Alice and Bob announce the bases used in their communication, Eve obtains 100% information on the photon states that she acquired from beam-splitting. Since this measurement is off-line, Alice and Bob will not be able to detect the presence of beam/splitting attack. The counter measure by Alice and Bob is to use an exotic single photon source or to lower the average number of photons (μ) emitted by a common light source, typically at the range of 0.1 photons per pulse. This limits the useful pulse to ~10%, yet reduces multi-photon pulses to ~1% (μ^2). Again, this is considered further in section 1.5.

1.4 Reconciliation procedure

Because the reconciliation procedure is performed over the classical channel, every parity check that Alice and Bob compare will be known to Eve. For the purpose of breaking the key, knowing the parity of a block is just as informative as knowing one bit deterministically; either way, the number of combinations Eve needs to try to break the key goes down by a factor of two. Alice and Bob need to throw away one bit each time they do a parity check to neutralize this information leakage to Eve. Eve's lost her knowledge of this block as the discarded bit, unknown to Eve, can be either "0" or "1" and the parity for the rest of the block can go either way. The problem is to design the most efficient algorithm to minimize the information leakage (fewest number of parity checks) while leaving as few residue errors as possible (high reliability or low residue bit error rate). I will first describe the reconciliation procedure used in [BBBSS], which I will call BB84 reconciliation procedure, and then discuss the improved CASCADE method from [BS], which I will call BS reconciliation procedure.

1.4.1 BB84 reconciliation procedure

Stage one of BB84 reconciliation

Together, Alice and Bob randomly divide the **tentative final key** into blocks of size k_1. The block size is selected such that few blocks contain more than one error. Therefore, k_1 is a function of error rate p. Alice sends the parity of each block to Bob. Bob compares them to his parities and both discard the last bit from each block. For blocks with parity error, a binary search procedure is used to locate the error. For the rest of this thesis, I will use "BINARY" to represent the binary search routine. BINARY is a standard divide and conquer method: divide the block into two halves, compare the parity of one half, and discard the last bit of the subblock for each parity check; these are done recursively until the block size becomes one[1]. The number of parity checks need to locate an error bit

[1] Notice that sometimes the BINARY search may not find the error bit location because the error bit may be discarded by chance.

is a function of block size; literature typically uses the upper bound for information loss: $\lceil \log(k_1) \rceil$ bit loss per error bit (all logarithms are to the base 2 in this thesis). This is called the first pass. Some error bits remain undetected after the first pass.

Then they repeat the process (the second pass): randomly divide the remaining bits into blocks of length k_2, compare parities and discard the last bit of each block, and do BINARY on blocks with parity error. Then again repeat the process (the third pass) by randomly dividing the remaining bits into blocks of length of k_3, and so on. They will do several passes until the number of errors in all the remaining bits is small enough (probably less than two). The block size selection rule, to have few blocks containing multiple error bits, applies to all passes. As the error rate decreases after each pass, obviously the block size increases after each pass, i.e. $k_{i+1} > k_i$. However, the exact formula on block size was not given in [BBBSS].

Stage two of BB84 reconciliation

Alice and Bob take randomly half of the remaining bits, do a parity check, and discard one bit. If they succeed 20 times without parity errors, they will accept the final string as reconciled key at 2^{-20} error rate (or less), or about one error bit in a million. If anytime they find a parity error, they use BINARY to locate the error and start over from the beginning of the stage two. The BINARY procedure is costly at this stage, so it is important to know when to transition from the first stage to the second stage. Again, this formula is not well defined in [BBBSS]. Notice that the above residue error rate estimation is conservative.

1.4.2 BS reconciliation procedure

Brassard and Salvail made an inspiring improvement which they called CASCADE [BS]. In this procedure, Alice and Bob will not discard a bit immediately after parity checks. Instead, they will mark a bit to be discarded later. This change increases the number of parity checks needed to locate the error bit, but guarantees to locate an error bit. Bob will correct his error bit. Because of such changes, each block will contain an even number of errors (possibly zero) after the error search procedure.

Error search procedure

Alice has a string A and Bob has a string B. Let $A = A_1, \ldots, A_k$ and $B = B_1, \ldots, B_k$ (with $A_i, B_i \in \{0,1\}$). The error search procedure is performed to locate one possible error bit:

1. Alice sends Bob the parity of whole block of k bits.
2. Bob determines whether the parity of his block is different. If so, Bob informs Alice to continue with the BINARY procedure. If not, stop.

If the error search stops at step 2, k bits remain to be in one single block and this block has an even number (possibly zero) of error bits.

BINARY procedure

1. Alice divides A into two subblocks of sizes $\lceil k/2 \rceil$ and $\lfloor k/2 \rfloor$, respectively, and sends the parity of the first half to Bob.
2. Bob determines which subblock contains an odd number of errors. If the size of that subblock is one, stop, this is the error bit and Bob corrects his bit. If the size of that subblock is larger than one, Bob replaces B with that subblock and informs Alice which subblock to continue. Alice also replaces A with that subblock. Both reduce k accordingly. Go back to step 1.

Although Alice only send the parity of the first half to Bob at step one, combining with the parity of the whole block sent earlier, the parity of the second half is also known to Bob (and to Eve). The half without parity error will not be processed further. Therefore, at the end of BINARY procedure, the original block is broken down into $\lceil \log(k) \rceil + 1$ or $\lceil \log(k) \rceil$ subblocks, each subblock has an even number (possibly zero) of error bits.

Alice and Bob share a **tentative final key** which is n bits long and with bit error rate p. On average, pn bits of Bob's string are different from Alice's (error bits). The goal is to remove multiple error bits, it is more efficient to divide A and B into subblocks before applying the error search procedure. Let $A = A_1, \ldots, A_n$ and $B = B_1, \ldots, B_n$ (with $A_i, B_i \in \{0,1\}$) be Alice's and Bob's strings, respectively.

BINARY sweep procedure

Alice and Bob choose a block size k and divide their string into blocks of k bits. They perform error search procedure on every subblock with parity error.

Notice that before the BINARY sweep, there are $\lceil n/k \rceil$ blocks. After the BINARY sweep, the number of blocks is more than $\lceil n/k \rceil$. Many blocks remain k in size, but there are also many smaller blocks. All blocks have an even number of errors after BINARY sweep.

Now, the BS reconciliation procedure is described below:

the first pass of BS reconciliation

Alice and Bob choose the initial block size k_1 and perform the BINARY sweep procedure. Let $K_1 = \{K_{1,1}, \ldots, K_{1,j}, \ldots\}$ be the set of all blocks after the first pass (blocks of sizes k_1 or smaller).

the ith pass of BS reconciliation; $i > 1$

1. Alice and Bob randomly permute (or interleave) all n bits. This interleaver is referred to as random interleaver; interleaver design is one of the main topics of this thesis (see for instance section 2.2.3.3).
2. They choose k_i and perform BINARY sweep procedure.

Let $K_i = \{K_{i,1}, \ldots, K_{i,j}, \ldots\}$ be the set of all blocks after BINARY sweep of the ith pass.

Some blocks in $K_1, \ldots, K_{i-1}$ now show parity error and they put all these blocks in a queue, Q. Then they perform the following CASCADE procedure:

1. If Q is empty, then stop. If not, find the smallest block $K_{m,j}$ in Q ($1 \le m \le i$).
2. Apply BINARY to $K_{m,j}$ and locate an error bit $e1$. The block structure of K_m is updated by replacing $K_{m,j}$ with its subblocks. (The number of elements in K_m, $|K_m|$, increases but $\Sigma_j|K_{m,j}|$ remains the same). Bob corrects his error bit.
3. For $1 \le q \le i$, find $K_{q,j(q)}$ such that $e1 \in K_{q,j(q)}$ ($j(q)$ is simply different block index for each pass). If $K_{q,j(q)} \in Q$,, then delete $K_{q,j(q)}$ from Q; if $K_{q,j(q)} \notin Q$, then add $K_{q,j(q)}$ to Q.
4. Go to step 1.

The guideline for the block size selection at the start of each pass is that few blocks contain more than one error bit. Therefore, $k_1 = f(p)$, the initial block size depends on input error rate. An initial block size selection and block size doubling after each pass ($k_i = 2k_{i-1}$) are recommended in [BS]. With this parameter set, the error rate is reduced at least by half after each pass. For 10000 bits input size and 10 runs, no residue errors were found after four passes.

Given a block size k and input bit error rate p (BER), the probability of having x number of errors in this block follows the binomial distribution:

$$Bin(x,k,p) = \binom{k}{x} p^x (1-p)^{k-x} \tag{1}$$

It can be shown that the probability that a block has an odd number of errors is [W75]:

$$p_{odd} = \sum_{i=1}^{\lceil k/2 \rceil} Bin(2i-1,k,p) = \frac{1-(1-2p)^k}{2} \tag{2}$$

After the BINARY of the first pass, the BER is reduced by the amount of p_{odd}/k_1: ($k = k_1$ here and the denominator k_1 is to convert p_{odd} to bit error rate)

$$p_1 = p - \frac{1-(1-2p)^{k_1}}{2k_1} \tag{3}$$

Because the complicated CASCADE procedure, error rate and information leakage after the second pass or later were not obtained in [BS]. Instead, upper bounds were given. The BS reconciliation procedure suggests how to choose the block size for the first pass. Each subsequent pass will double the block size of the previous pass. In such a way, it is proven in [BS] that

$$p_i \le \frac{p_{i-1}}{2} \tag{4}$$

It will be shown later that this bound for residue error rate is too weak to be useful.

1.5 System Performance Analysis

I have explained the QKD protocol and the reconciliation procedure. Basically, the reconciliation procedure will affect the QDK throughput and the residue error rate. The QKD system also has an upper limit on the input error rate beyond which no secure key can be derived (BER of 7%~9% was derived as the upper limit in [SSMRF]). A reconciliation procedure with higher throughput will allow a higher upper limit on input BER. The role of the reconciliation procedure in the overall QKD system performance is discussed below.

Privacy, security, throughput, and reliability are some important system performance parameters. Privacy means that Eve has little knowledge about the secret key. Security is that Eve cannot fool Alice and Bob into thinking they have a private key. Throughput is how many bits are derived from QKD during a fixed time period (e.g. one-second). Reliability is how often Alice and Bob will get wrong bit in their shared, final secret key. Privacy and secrecy are by far the most important concerns in any encryption scheme including QKD. These two parameters were, and still are, the main focus in the literature. Another parameter is the system yield. Given a set of hardware capability, the yield may be considered as directly proportional to throughput. In the experimental example demonstrated in [BBBSS], in the duration of 715,000 of photon pulses and 4% error rate, only 754 bits were derived eventually. This is slightly over 0.1% yield. More recent analyses by Slutsky et. al. [SSMRF] showed that 10^4~10^5 per second throughput, based on 50 ns pulse frequency. This yields only a fraction of one percent also.

Reliability of the final shared string is one parameter that is seldom discussed in the literature. The upper bound of output error rate is 10^{-6} in [BBBSS] and about five percent of the input error rate in [BS]. Privacy amplification generates the final secret key from reconciled key by randomly selecting subgroups from the reconciled key and taking the parity of each subgroup as the key bit. Therefore, privacy amplification also amplifies the error rate as the residue error bits are used many times in various subblocks. It seems the impact of privacy amplification on reliability was not discussed in the literature. Depending on the application, such error rate in the key may or may not be acceptable. For some applications such as encryption of plain text using a one-time pad, errors in key may not be critical as there is plenty of redundancy. For other applications, the error rate needs to be tighter (such as keys for further standard encryption, or transmitting compressed data using a one-time pad). For certain applications, the reliability of the final string may need to be essentially error free (such as keys used in military mission).

The maximizing of one particular performance parameter will generally be made at a cost to the other performance parameters. For example, to improve the privacy, one could sacrifice yield and error rate in the privacy amplification step. To improve reliability, one can do more passes in reconciliation and again, lose yield. The optimization of the whole system is not likely to happen soon. In fact, the optimization of a single system

component (between the throughput and reliability in reconciliation procedure from [BS]) is a difficult problem.

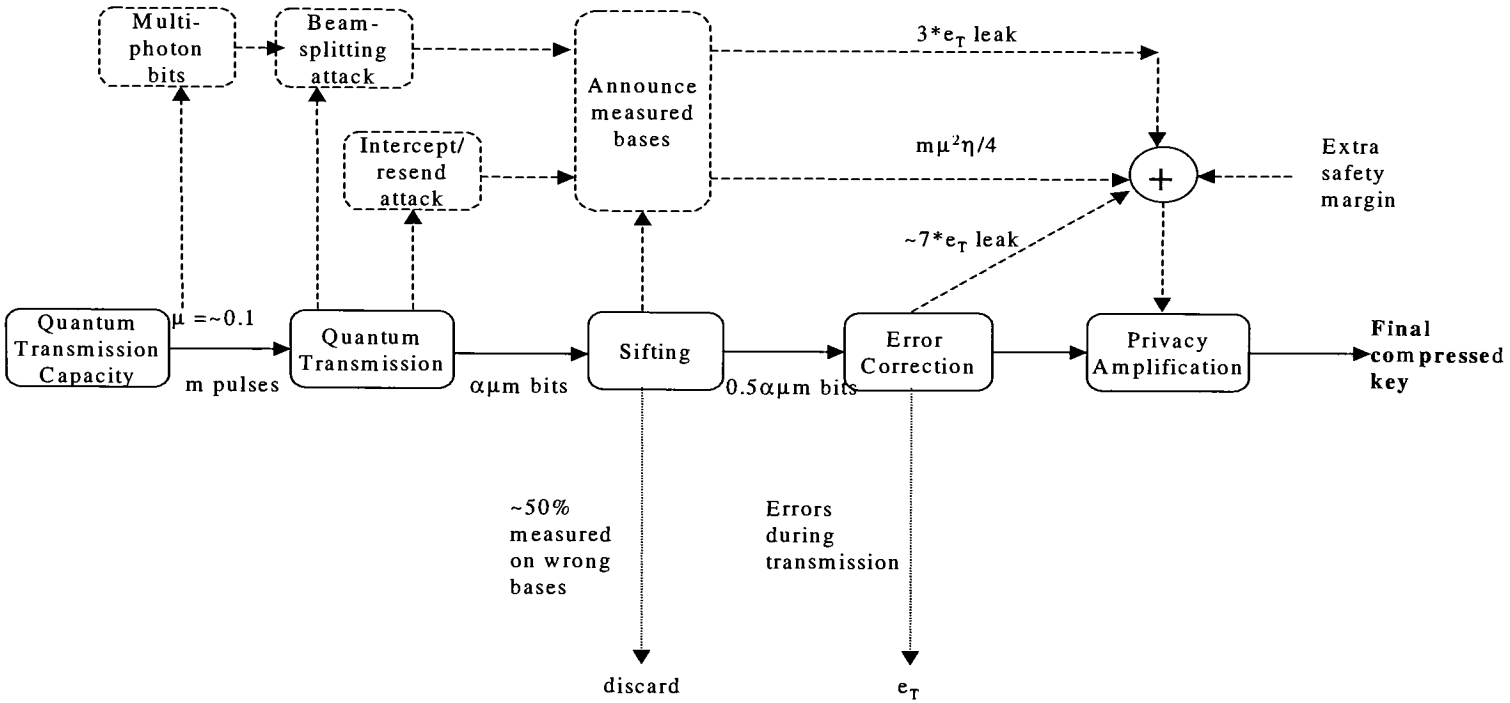
The impact of the reconciliation procedure on the system throughput is illustrated in [SSMRF]. I redraw their analysis in Fig. 3. In this figure, m represents the number of photon pulses Alice sends, α the attenuation in the quantum channel (the fraction of photons that reached Bob's detector), η the efficiency of Bob's measurement device, μ the average number of photons per pulse, e_T the total number of errors in the string. Therefore, the size for the tentative final key as the input of reconciliation procedure is $n = 0.5\alpha\mu m$ and $e_T = pn$. The main path of Fig. 3 illustrates the three most important factors (α, μ, and m) that greatly and directly affect throughput. The upper path of Figure 3 illustrates the effects of eavesdropping on the throughput that are related to the observed error rate p. Many factors affect observed errors: the channel attenuation, the channel error rate, dark count of the measurement device, η and μ. The limit of the system security is estimated to be 9% in [SSMRF], above which no secure QKD can be achieved. One key observation from this figure is that the largest item that leads to information leakage is the reconciliation procedure ($\sim 7 * e_T = 7pn$). However, as will be seen later in Fig. 4, the estimation of information leakage of $7pn$ was not quite accurate. The factor 7 should not be a constant, instead, it should be a function of p. This factor decreases as p goes up. Therefore, the 9% estimation was too conservative. When p is small, the effect of reconciliation on throughput is insignificant. Improvement of reconciliation procedure can help the system throughput only when p is large.

As a side note, one recent paper by Ardehali et. al. [ACL] claimed nearly doubling the throughput of the BB84 protocol. This is achieved by, instead of randomizing the quantum bases in transmission, using bias toward one set of bases over another by both Alice and Bob.

Much hope in throughput increase (by leaps and bounds) comes from hardware improvements: thousand fold increase in pulse rate from pico-second electro-optical devices, single photon light sources, reduction in transmission error rate in optical alignment, or reduction in attenuation by low loss optical-fiber. On the other hand, the optimal performance of one component may not match the optimal performance of another component. One example is the optimal wavelengths are different for optical fiber, light source, and detection efficiency. Improving the reconciliation procedure will have a small effect on the overall throughput (roughly a factor of two at most). However, improving the reconciliation procedure may allow higher upper limit of p for QKD, which, in turn, allows more flexibility in choosing system components for system design engineers.

Unlike its small impact on throughput, improvement in the reconciliation procedure will have a big impact on reliability and is the only place where reliability can be achieved. Reliability is less extensively covered in the literature (some discussions can be found in a series of papers by Mayers and Yao [M, MY, MYa]), in contrast to the well covered subjects of privacy and security. In the experimental examples from [BBBSS], the reliability is 2^{-20} (or $\sim 10^{-6}$), which may not be suitable for those critical applications. One

Figure 3: The capacity loss due to two groups, 1) system capability such as attenuation, low average photon per pulse, measurement on wrong bases, error in transmission (the center solid line), and 2) to prevent eavesdropping activities (the upper portion of the figure).



may be tempted to do more parity checks at the second stage of the BB84 reconciliation procedure to increase reliability . However, one needs to have a good estimate of residue error rate before going into the second stage of the BB84 reconciliation procedure that still requires more passes in the first stage of reconciliation. Besides, the second stage of the BB84 reconciliation procedure may not be the most efficient way of trading throughput for reliability. Although the authors observed no error left in their simulation of 10 runs after four passes in [BS], the bound on output BER is very weak. Therefore, to optimize the reconciliation procedure, one needs to have a better method to define the residue BER, which allows more accurate trade-off between throughput and reliability.

1.6 The need of more reconciliation analysis

In summary, the problems that need to be studied further in the BS reconciliation procedure are:

1) Optimal block size and block size schedule;
2) Lack of accurate estimate of output BER;
3) Lack of accurate estimate of information leak;
4) The window for p (input BER) was extended to 15% (up from 9% from [SSMRF]). Can we do better?

2 Simulation results and analysis

I will compare the performance of my improved reconciliation procedure to the simulation result of [BS], and the data from [BBBSS] in section 2.1. Then, I will show an analysis procedure to calculate the CASCADE performance in section 2.2. Finally, I will discuss how the improvement in reconciliation was achieved in section 2.3.

2.1 Simulation results comparison

The throughput comparison is shown in Fig. 4, and again in Fig. 5. In Fig. 4, the y-axis is the average number of parity checks needed to find one error bit. This is the parameter typically used for comparison in QKD literature and it is more useful from the privacy parameter point of view. In Fig. 5, the y-axis is throughput, which is more useful from the system performance point of view.

My simulation results of the BS reconciliation procedure are almost the same as the simulation data report in [BS][2], and very close to the information leakage bound given in [BS] (Fig. 4). Note that the error bound given in [BS] overestimated the information leakage at high input BERs (this will be addressed in section 2.2.1). BB84 experimental data is also shown in this figure. The advantage of the BS procedure over the BB84 procedure is mainly at low input BER, which confirm the improvement claimed by [BS]. My method, also shown in Fig. 4, outperforms the BS procedure at all input BERs.

[2] All parameters were set at the recommended block size value, block size schedule for the four passes, and the batch size. The results of this simulation are an average of 200 runs or more, while there were only 10 runs in [BS] report.

From the throughput point of view, the BS procedure shows only minor improvement at low input BER (Fig. 5). My procedure shows insignificant advantage over the BS procedure at low input BER. At high BER, the advantage of my improvement is clearly significant. The highest input BER that can be used in my procedure is 30%, at which ~1.8% of bits remained after reconciliation.

Table 1, which is the same data as in Fig. 4 and 5, shows both throughput and reliability data. The reliability data, or the observed residue BER, is the observed total residue error bits divided by n, then averaged over the number of runs. For the BS procedure at high input BER, three passes are sufficient to remove all error bits. Four passes were recommended in [BS] because, sporadically, one or two error-pairs are missed for low input BER runs. This suggests that error-pairs, due to the randomized permutation of bits after the each pass, stick together through the first three passes. I'll call this problem "error-pair sticking." This problem can be avoided, or reduced, by using an interleaver other than randomization. Interleaver design is discussed in section 2.2.3.3.

Because the sporadic nature of the residue errors, the observed output BER is only a rough estimation. Those zero BER shown in Table 1 merely indicate that no errors were observed during hundreds of test runs. Because the output BER is so low, billions of runs are needed to get an accurate estimate of output BER. Therefore, such simulation cannot be useful for defining the reliability of the QKD. The analytical approach in sections 2.2.1 and 2.2.3 is our best hope to derive reliability parameter.

One way to dissect the performance of CASCADE of the BS reconciliation procedure is to trace its performance at the end of the second pass, so that the BER is measurable. This is shown in Table 2 at input BER of 15%. The top of Table 2 shows results a run with 10k bit input size and the bottom with 500k input size. Notice that the average percentages of bits left are about the same at various passes, independent of the input batch size. However, the residue BER is much smaller for the larger input size. Some runs are error free at the end of the second pass while others contain some number of hidden errors. The number of runs containing two, four, six, or eight errors at the end of the second pass is also listed for the 10k input size. The following list summarizes the shortcomings of the BS procedure and surprises:

1) The BS reconciliation procedure recommends four passes. However, my simulation clearly indicates that three passes are needed at most.
2) The residue error rate at the end of second pass (8.6E-5) is much lower than the upper bound reported in [BS] (3.34E-2).
3) Most of the time, the residue errors were in pairs (see middle of Table 2).
4) More intriguing, with larger input batch size, the residue error rate is lowered (1.5E-6 for n = 500k, bottom of Table 2).
5) The BS reconciliation procedure overestimated information leakage at high input BER (Fig. 4).

Figure 4: My simulation of the BS reconciliation procedure matched well with the bound given in [BS]. This also confirms the advantage of the BS procedure over the BB84 procedure [BBBSS]. My procedure had lower information leakage than the BS procedure at all input BER.

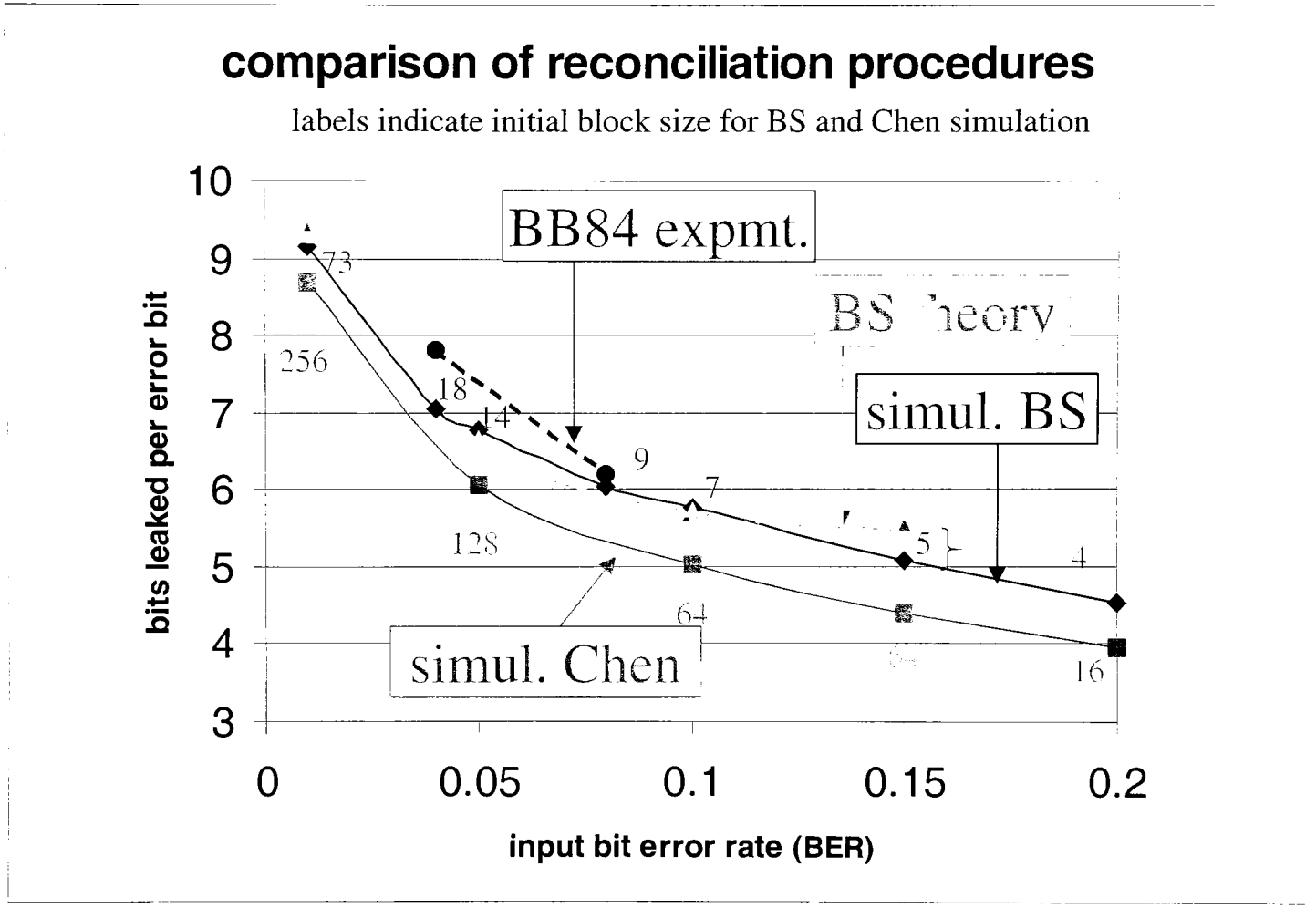


Figure 5: In term of throughput, the BS reconciliation procedure has a small advantage over the BB84 procedure. My procedure is better than the BS procedure except at very low input BER.

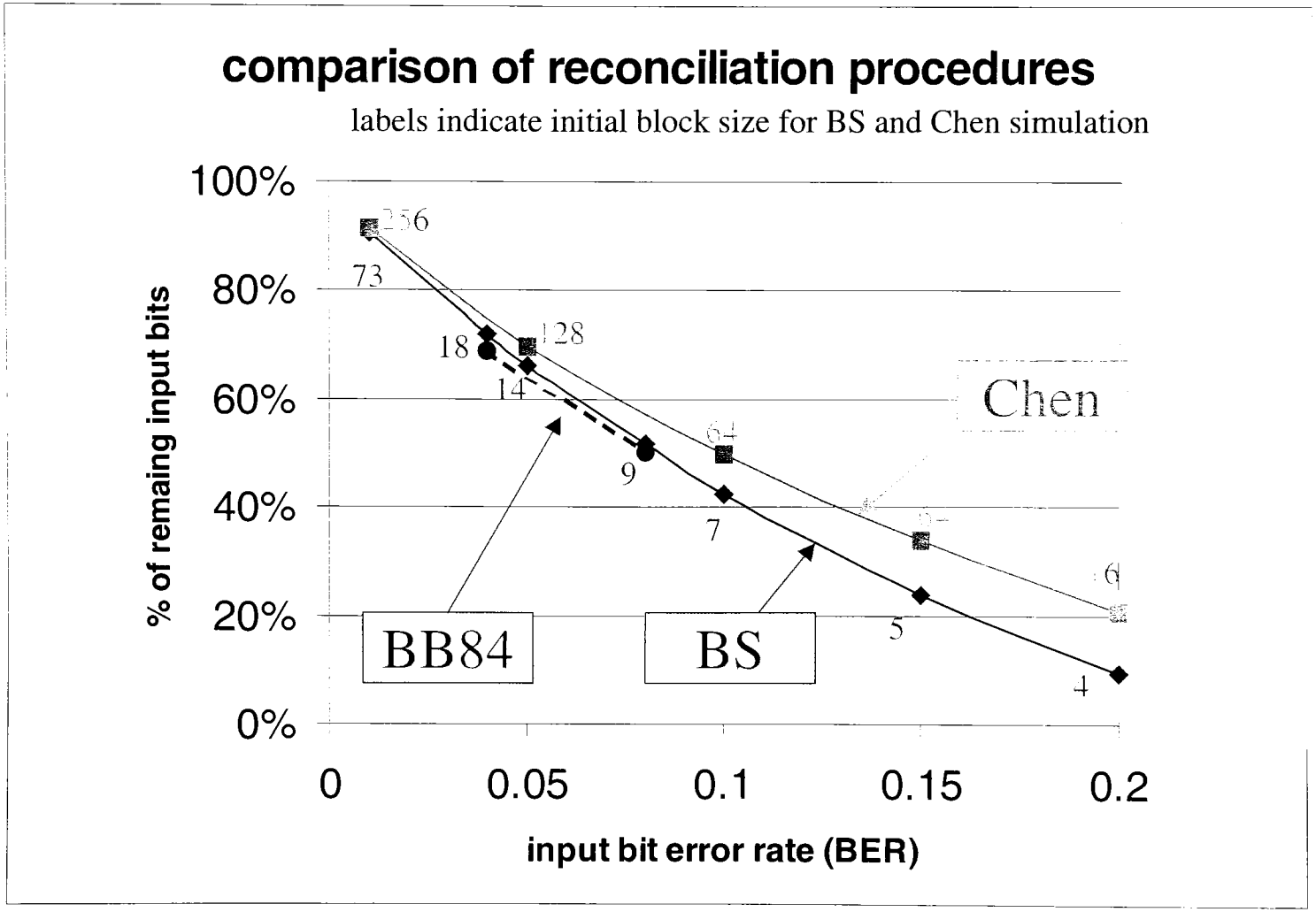


Table 1: Throughput and output BER performance data of the BS procedure as a function of input BER (block size schedule $k_4 = 2k_3 = 4k_2 = 8k_1$ as recommended in [BS]). At low input BER, sporadically, the BS procedure leaves one or two error-pairs undetected at the end of the third pass. At high input BER, the BS procedure over-reconciles (four passes used when three passes are sufficient). The output BER is the number of residue error bits divided by n and then averaged over the number of runs.

				after third pass			after 4th pass	
p	k_1	n	# runs	%bits left	BER	# of error left	bits left	BER
0.01	73	10k	225	91.03%	8.85E-07	1 run with 1 error-pair	90.85%	0
0.04	18	10k	200	72.52%	2.00E-06	1 run with 2 error-pair	71.82%	0
0.05	14	10k	200	66.99%	1.00E-06	1 run with 1 error-pair	66.09%	0
0.08	9	10k	252	53.14%	0	0	51.75%	0
0.10	7	10k	200	44.23%	0	0	42.44%	0
0.15	5	10k	220	26.32%	0	0	23.81%	0
0.20	4	10k	100	12.59%	0	0	9.46%	0

Performance of improved reconciliation procedure, runs with block size schedule $k_3 = 2k_2 = 4k_1$

p	k_1	n	# runs	%bits left	BER	# of error left
0.01	256	100k	100	91.31%	0	0
0.05	128	100k	100	69.66%	0	0
0.10	64	100k	100	49.83%	0	0
0.15	64	50k	100	33.9%	0	0
0.20	16	10k	100	20.6%	0	0
0.25	8	10k	10	10.5%	0	0
0.30	8	10k	10	1.8%	0	0

Table 2: Simulation results after each pass of the BS reconciliation procedure. Notice the lower BER at the end of second pass with larger input size.

observations from 220 runs with 10k bits input and 15% BER.					
pass	0	1	2	3	4
avg. % bits left	100%	60.1%	31.35%	26.3%	23.8%
BER	15%	6.7%	**8.6E-05**	0	0
number of runs with x error bits left after 2nd pass					
x	0	2	4	6	8
# of runs	147	56	13	3	1

observations from 13 runs with 500k bits input and 15% BER.					
pass	0	1	2	3	4
avg. % bits left	100%	60.1%	31.42%	26.4%	23.9%
BER	15%	6.7%	**1.5E-06**	0	0

Table 3: The number of parity checks needed to locate one error bit, $i(k)$, after finding parity error in block of size k (equation (5)). The upper bound is given in the third column.

k	$i(k)$	$\lceil \log(k) \rceil$
1	0.00	0
2	1.00	1
3	1.67	2
4	2.00	2
5	2.40	3
6	2.67	3
7	2.86	3
8	3.00	3
9	3.22	4
10	3.40	4

2.2 Analysis

2.2.1 Tracking CASCADE

The above observations from Table 2 can be explained by tracking the information leakage and residue BER performance of the BS reconciliation procedure for the first two passes. At the first pass, only BINARY sweep is applied. At the second pass, both BINARY sweep and CASCADE are applied. The BINARY sweep procedure has closed form formulas that describe the information leakage and the output BER. I will demonstrate the calculation of the residue BER and the information leakage for the CASCADE procedure with the restriction that the number of errors per block is less than four.

When there are many blocks of size k and with parity error, the average number of parity checks, $i(k)$, needed to locate one error bit is derived as the following recursive formula: $i(1) = 0$ and $i(2) = 1$; and

$$i(k) = \frac{\lceil k/2 \rceil}{k} i(\lceil k/2 \rceil) + \frac{\lfloor k/2 \rfloor}{k} i(\lfloor k/2 \rfloor) + 1 \qquad (5)$$

The last term accounts for the one parity check that determines which half the error bit is located. The first two terms are just the probability of error being in each half times the cost of locating error bit in that half. Table 3 shows some values from this formula and comparison to the crude upper bound used in the literature. Notice that when $k = 5$, the bound used in literature is 25% too big. This is the reason that the bound from [BS] is larger than simulation at $p = 0.15$ (corresponding to $k = 5$), as observed in Fig. 4.

The analysis strategy is to separate the BINARY sweep portion from CASCADE tracking for the second and later passes. The CACADE procedure will be divided into smaller steps and will utilize the information from BINARY sweep.

BINARY sweep in pass one

The output BER, p_1, can be calculated from equation (3). The information leakage I_1 from the first pass is:

$$I_1 = \left\lceil \frac{n}{k_1} \right\rceil * \left(1 + p_{odd} * i(k_1)\right) \qquad (6)$$

where p_{odd} is given by equation (2). The first term in the parentheses is due to the the parity checks of all blocks, and the second term is the successive parity checks due to BINARY for the blocks with parity error.

BINARY sweep in pass two

After the BINARY sweep, the BER p_2, and the information leakage I_2, can be calculated using again the same formulas as in pass one.

$$p_2 = p_1 - \frac{1 - (1 - 2p_1)^{k_2}}{2k_2} \qquad (7)$$

$$p_{odd,2} = \frac{1 - (1 - 2p_1)^{k_2}}{2} \qquad (8)$$

$$I_2 = \left\lceil \frac{n}{k_2} \right\rceil * \left(1 + p_{odd,2} * i(k_2)\right) \qquad (9)$$

CASCADE in pass two: BER

For easy reference, the BINARY sweep in pass one will be referred to as round 1 ($r = 1$) and the BINARY sweep in pass as round 2 ($r = 2$). The CASCADE will be further divided into several smaller steps, each will be referred to as round 3 and larger ($r = 3, 4, \ldots$). During the CASCADE procedure, BINARY is applied to the blocks in the Q in the order of block size and Q contains blocks from both K_1 and K_2. To simplify the calculation, I will assume there are two queues, Q_1 and Q_2, one for each pass. Q_1 is a subset of K_1 and Q_2 a subset of K_2. At the end of the first pass, all blocks in K_1 have even number of error bits and, therefore, Q_1 is empty. At the end of round two, Q_2 is empty but some blocks in K_1 now show parity error because the error bits found and corrected during round two are randomly distributed in K_1. These blocks will be placed in Q_1. BINARY is then applied to all the blocks in Q_1 and this is called round three ($r = 3$). At the end of round three, Q_1 is again empty but Q_2 is not because the errors found during round three are randomly distributed in K_2. BINARY is then applied to all blocks in Q_2 and this is called round four ($r = 4$). This process alternates between Q_1 and Q_2 until both Q_1 and Q_2 are empty.

Notice the odd round number means BINARY is applied to blocks in K_1 and the even round number means BINARY is applied to blocks in K_2. The above simplification seem to have little impact on output BER, but it does over-estimate the information leakage a little bit. At the end of this section I will partially correct this small over-estimate.

To further simplify the calculation, I will consider only $x \leq 3$ by assuming the probability of $x \geq 4$ is negligible. In other words, a block can have at most one hidden error-pair after BINARY. This assumption is pretty safe with the small k_1 recommended in [BS].

Let h_r and e_r represent the number of hidden errors and the number of exposed errors after round r, and $h_r + e_r = h_{r-1}$. For example, $h_1 = p_1 n$, $h_2 = p_2 n$, and $e_2 = h_1 - h_2 = (p_1 - p_2)n$ are already known. Now the problem is to find e_r for each round $r \geq 3$.

For each round r we need information from the previous two rounds (r-1 and r-2) to calculate e_r. Both round r and round r-2 are dealing with block structure K_1 if r is odd, or K_2 if r is even. At the end of round r-2, there are h_{r-2} hidden error bits (set T). By the above assumption of $x \leq 3$, these h_{r-2} bits are distributed in $h_{r-2}/2$ blocks in K_1 (or in K_2) each having one hidden error-pair. During round r-1, set T is divided into two subsets, e_{r-1} bits in set E are exposed and h_{r-1} bits in set H remain hidden ($T = E \cup H$). The bits in E is randomly distributed in these $h_{r-2}/2$ hidden error-pairs. Therefore, the probability of one of these $h_{r-2}/2$ hidden error-pairs having y number of error bits in E is according to a binomial distribution $Bin(y, 2, e_{r-1}/h_{r-2})$, where $0 \leq y \leq 2$. In round r, those blocks in K_1 (or K_2) containing a hidden error-pair which has one error bit in E ($y = 1$) are now showing parity error and BINARY will be applied to these blocks to detect another error bit. Therefore, the number of hidden error bits and the BER after each round r are:

$$e_r = Bin(1, 2, e_{r-1}/h_{r-2}) * \frac{h_{r-2}}{2} \tag{10}$$

$$h_r = h_{r-1} - Bin(1, 2, 1 - h_{r-1}/h_{r-2}) * \frac{h_{r-2}}{2} \tag{11}$$

$$\hat{p}_r = \frac{h_r}{n} \tag{12}$$

These equations completely map out how the residue error rate diminishes at each round of CASCADE. The CASCADE will proceed, on the average, until $e_r < 1$, and at the end of the second pass, $p_2' = \hat{p}_r$. Notice that e_r is proportional to n; therefore, the larger input size, the more rounds of CASCADE will be performed.

As an example, I will calculate the BER and information leakage through two passes of the BS procedure at $p = 0.15$ and $n = 10$k. The BINARY sweep portion of this calculation is shown in Table 4 and the tracking of $\hat{p}_r$ during CASCADE is shown in Table 5. Notice that $e_r < 1$ at $r = 10$, and CASCADE stops at this point. Therefore, $p_2' = \hat{p}_{10} \sim$ 3.3E-5 for $n = 10$k. If $n = 500$k, $e_r < 1$ until $r = 14$ (the column of e_r times 50), and $p_2' = \hat{p}_{14} \sim$ 1.1E-6 in this case. This observation nicely explains why the output BER was found smaller at larger input batch size in Table 2.

The result of the above CASCADE tracking example, $p_2' \sim$ 3.3E-5, is much closer to the experimental observation in Table 2 (8.6E-5) than the error bound given in [BS] (3.3E-2). The error bound given in [BS] is too weak to be useful. The small difference between the above CASCADE tracking example and experimental observation in Table 2 is explained in the following paragraphs.

Notice that allowing bits in E to be randomly distributed in T implies that a hidden error-pair in one pass cannot form the same error-pair for the other pass; otherwise, some of the $h_{r-2}/2$ hidden error-pairs will exclude bits from E. In other words, this derivation forbids the "error-pair sticking", which is one important factor in interleaver design, as will be discussed in section 2.2.3.3.

Using a random interleaver in constructing blocks for each pass, some hidden error-pairs from the first pass have a small but definite probability to be in the same block of the second pass (and then in the same block of the third pass). I'll refer to this as "error-pair sticking" in this thesis. The error-pair sticking is not allowed in the way the CASCADE tracking in Table 5 is calculated. The error-pair sticking problem is more serious when the input size is small and the initial block size is large, as expected from the use of a random interleaver. This is why the error-pair sticking is observed at low input BER (larger initial block size) in Table 1 and why most hidden errors were in pairs at the end of the second pass, as reported in Table 2.

Table 4: The output BER and information leakage from the first two rounds of the BS reconciliation procedure. The input size n = 10k bits and p = 0.15.

round	block size k	# of input error bits	p_i eqn(3) or eqn(7)	# of errors left = p_i*n	# of errors exposed = n_e	total # of parity checks I_i	# of bits left
1	5	1500	6.68%	668	832	3996.8	6003.2
2	10	668	2.87%	287	381	2295.4	3707.8

Table 5: Tracking of error rate during CASCADE for the first two passes.

input parameters:
input size n = 10k bits, p = 15%, k_1 = 5, k_2 = 10

round	h_r	e_r	$\hat{p}_r$
0	1500	0	15%
1	668.1	831.9	6.68%
2	287.2	380.9	2.87%
3	123.3	163.9	1.23%
4	52.9	70.4	0.53%
5	22.7	30.2	2.3E-03
6	9.8	13.0	9.8E-04
7	4.19	5.57	4.2E-04
8	1.80	2.39	1.8E-04
9	0.77	1.03	7.7E-05
10	0.33	**0.44**	**3.3E-05**
11	0.14	0.19	1.4E-05
12	0.06	0.08	6.1E-06
13	0.03	0.03	2.6E-06
14	0.01	0.01	**1.1E-06**
15	0.005	0.01	4.8E-07
16	0.002	0.00	2.1E-07

An efficient implementation of reconciliation procedure needs to avoid error-pair sticking with a better interleaver design. In section 2.2.2, I will estimate the contribution from this error-pair sticking using the parameters of the above example, just to support this claim. Another type of error that will escape the detection from CASCADE is the "error-cluster" formation, discussed in section 2.2.3.

Notice that in the BS reconciliation procedure, the block size for the third pass is twice the block size of the second pass. Now, it is clearly confirmed that the output BER at the end of second pass is much lower, the use of $k_3 = 2k_2$ does not make sense. (Recall the guideline for choosing block size in [BS] is to have few blocks with more than one error). There are two ways to utilize this knowledge. One way is to use much larger k_3 so long as it fits the guideline of block size selection. However, this will improve the throughput only slightly as most of the information leakage occurs at the first two passes. The more effective way to increase the throughput is to increase the initial block size. I have implemented the larger block size in my experiments reported in Fig. 4 and 5.

CASCADE in pass two: Information leakage

To simplify the example, again I will ignore multiple error-pairs in one block (i.e. blocks with $x \geq 4$ are not considered). Therefore, only blocks with $x = 2$ and $x = 3$ errors (before BINARY sweep) need to be considered here. As mentioned earlier in section 1.4.2, blocks with $x = 3$ are divided into subblocks of various sizes (an example is shown in Fig. 8) and one of these subblocks contain one hidden error-pair after the first pass. Blocks with $x = 2$ remain intact and also contain one hidden error-pair.

During CASCADE, all these blocks with one hidden error-pair are processed by BINARY once and only once. The detection of the first bit of the error-pair comes from the BINARY of the blocks of the other pass; the cost is counted there. The detection of the second bit of the error-pair is as follows. For blocks with $x = 2$, the cost of BINARY during CASCADE is $i(k_1\text{-}1)$. The minus one is because one position in this block is already known to be an error bit. For blocks with $x = 3$, the cost of BINARY is smaller than $i(\lceil k_1/2 \rceil\text{-}1)$.

The numbers of error-pairs being exposed by CASCADE are:

$$\varepsilon_1 = \sum_{odd,\, r>1} e_r \quad \text{and} \quad \varepsilon_2 = \sum_{even,\, r>2} e_r$$

for K_1 and K_2, respectively. Some of these blocks originally have two or three errors ($x = 2$ or 3). Let f_j be the fraction of blocks with $x = 2$ for the jth pass. That is,

$$f_j = Bin(2, k_j, p_{j\text{-}1})/[Bin(2, k_j, p_{j\text{-}1}) + Bin(3, k_j, p_{j\text{-}1})] \qquad (13)$$

where $p_0 \equiv p$. Then, the total number of parity checks during the CASCADE for the first two passes is:

$$I_{\&2} = \sum_{i=1}^{2} \varepsilon_i * \left[i(k_i - 1) * f_i + i\left(\lceil k_i/2 \rceil - 1\right) * \left(1 - f_i\right) \right] \qquad (14)$$

For the example given in Table 4, $I_{\&2}$ is found to be 630. The number of bits left is $n\text{-}I_1\text{-}I_2\text{-}I_{\&2} = 3078$. This value is slightly smaller than 3135 observed in Table 2.

The over-estimation in $I_{\&2}$ is now partially adjusted below. In the above calculation, I assume the BINARY is applied to all blocks in Q_i and then alternately to all blocks in $Q_{i\pm1}$. During round r of CASCADE, there are e_r blocks showing parity error (one of its error bit is in set E). Similarly, there are d_r blocks that have two error bits in set E:

$$d_r = Bin(2,2,e_{r-1}/h_{r-2}) * \frac{h_{r-2}}{2} \qquad (15)$$

For each odd round r, information leakage is over-estimated. Instead of search all blocks in Q_2 during previous round r-1, if we search each block one by one according to BS procedure, the locations of these d_r blocks will be known. At that point, it costs less to search the second error bits in these d_r blocks than to search the same error bits in Q_2. The cost of the former is about $i(k_1 - 1)$ and the later $i(k_2)$. The amount of over-estimation due to the above assumption is about

$$\sum_{odd,\, r>3} d_r * \{\, i(k_2) - i(k_1 - 1)\}$$

which is 34 bits in our example. This will bring the total bits left after the second pass from 3078 to 3112, even closer to the 3135 observed in Table 2.

The above summation skips round three because BS procedure starts CASCADE at round three (and the benefit of information leakage reduction starts at round four). By modifying BS procedure and start CASCADE at round two, there is an additional reduction in information leakage by $d_3 * \{i(k_2) - i(k_1 - 1)\}$ which is 152 bits in our example. This sounds a much bigger benefit than the 34 bits from the application of CASCADE to all other rounds. However, there is a twist to this modification. This throughput improvement is realized for larger n and small k_1; for the example given in Table 8 later, adding this modification to my method increases the throughput from 29.2% to 31.8% at $k_1 = 5$; but this modification also decreases the throughput from 33.4% to 32% at $k_1 = 16$. One side effect of this modification is that it will leave more error bits undetected at the end of the second pass and, therefore, more cost of detecting these errors at the third pass. For example, let $[a1,a2]$ and $[b1,b2]$ be two hidden error-pairs in K_1 after the first pass, and $[a1,b1,c1]$ are in $K_{2,a1}$ and $[a2,b2]$ are in $K_{2,a2}$ at the beginning of the second pass. If CASCADE starts at round three as specified in the BS procedure, BINARY will be applied to block $K_{2,a1}$ first and there is only 1/3 chance that $[a1,b1]$ remain undetected, resulting a hidden error-cluster (see section 2.2.3 and Fig. 6 for definition of error-cluster). On the other hand, by starting CASCADE at the round two, usually BINARY will not be applied to block $K_{2,a1}$ (as its block size is one of the largest in Q) and $c1$ will be detected first (from application of BINARY to a block in K_1). Consequently, the hidden error-cluster is almost surely to occur. A smart reconciliation implementation may decide at which round the CASCADE should start depending on n and k_1. This is one of the future improvement opportunities.

BINARY sweep of pass three

Again, after the BINARY sweep, the BER p_3, and the information leakage I_3, can be calculated using again the same formulas as in pass one.

$$p_3 = p_2' - \frac{1-(1-2p_2')^{k_3}}{2k_3} \tag{16}$$

$$p_{odd,3} = \frac{1-(1-2p_2')^{k_3}}{2} \tag{17}$$

$$I_3 = \left\lceil \frac{n}{k_3} \right\rceil * (1 + p_{odd,3} * i(k_3)) \tag{18}$$

After CASCADE, the BER is reduced to p_3', and the information leakage is I_3'. The CASCADE is even more complex as the error discovery process occurs among three passes. The tracking is similar in spirit to pass two. I will not do this portion of analysis in this thesis.

2.2.2 Error-pair sticking

The error-pair sticking problem ought to be removed by a better choice of interleaver for the input bits before each pass. This section is only to confirm that this problem reflects discrepancy observed in section 2.2.1.

Following the assumption that the probability of multiple error-pairs in a block is negligible. At the start of the second pass, the error rate is p_1. For each of the n/k_2 blocks of the second pass, the number of errors x in the block is binomial distributed with $p_x = Bin(x, k_2, p_1)$. Error-pair sticking is possible when $x \geq 2$. I'll restrict the calculation to only one error-pair sticking possibility even when $x > 4$. When a block has two errors ($x = 2$), the first error bit is freely chosen from any of the h_1 (or $p_1 n$) error bits, but the second error bit has limited choices for error-pair sticking to occur. There are h_1-1 choices of error bits but only one of them result in error-pair sticking. Therefore, the probability of error-pair sticking is $1/(h_1\text{-}1)$ for $x = 2$.

The probability of forming error-pair sticking for $x = 3$ can be divided into two parts. The first part is the probability that the first two error bits chosen are forming error-pair; that probability is given above. The second part is when the first two error bits are not an error-pair but the third error bit makes an error-pair with either one of the first two. The first error bit is freely chosen from any of the h_1 error bits, the second error bit has h_1-2 out of h_1-1 possible choices, and the third error bit has two choices out of the rest h_1-2 error bits for error-pair sticking to occur. Therefore, the probability for the second part is $2/(h_1\text{-}1)$. In general, the probability of one error-pair sticks to a block of the second pass with x number of errors is:

$$p_s = \sum_{i=1}^{x-1} \frac{i}{h_1 - 1} \tag{19}$$

For any block of the second pass with an odd number of errors, BINARY is applied to locate one of the errors before CASCADE. BINARY may find one bit from the error-pair and break it up. The probability that one of the error-pair bit is found during this BINARY is $2/x$ and the probability of this error-pair remain after BINARY is $p_r = (x\text{-}2)/x$. Therefore, the total BER attributed to the error-pair sticking for the first two passes is:

$$P_{\&2} = 2 * \left(\frac{n}{k_2}\right) / n * \sum_{x=2}^{k_2} p_x * p_s * p_r = \frac{2}{k_2} * \sum_{x=2}^{k_2} p_r * p_s * \sum_{i=1}^{x-1} Bin(x, k_2, p_1) \tag{20}$$

In our example, the output BER due to error-pair sticking is shown in Table 6. $P_{\&2}$ is found to be 4.67E-5, which is roughly the difference between the simulation (BER = 8.6E-5) and the CASCADE tracking (3.3E-5).

2.2.3 Error-cluster formation

The error-pair sticking problem due to random interleaver is explained in the last section. A better interleaver design may minimize or eliminate error-pair sticking problem. Even with error-pair sticking possibility eliminated, there are still some errors that remain undetected after two passes. This can happen if two (or more) hidden error-pairs of the first pass happen to swap partners and form different hidden error-pairs at the second pass. Together, these error bits escape the error detection from the CASCADE of the first two passes. I'll call these groups of error bits as hidden "error-clusters".

2.2.3.1 cluster formation rule

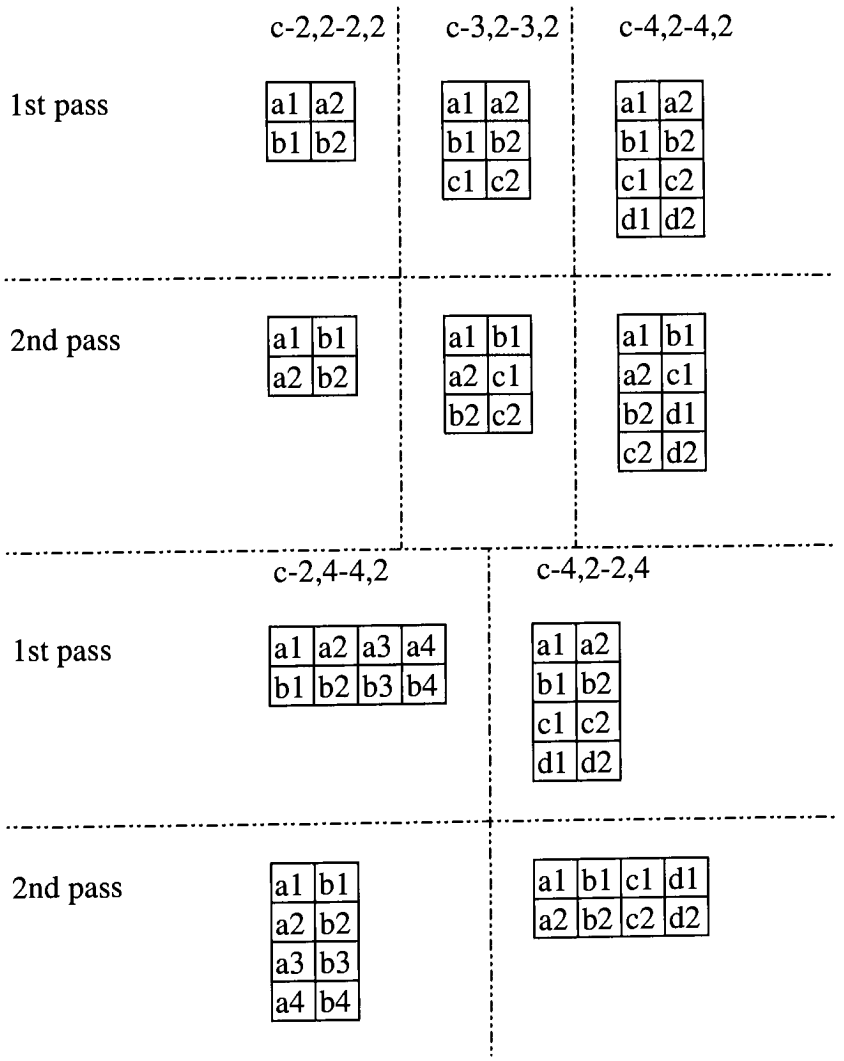
The number of hidden errors in a block in K_1 is any even number up to the block size k_1; likewise, a block in K_2 up to the block size k_2. Additionally, a better interleaver may achieve the following rule: that no two bits from the second pass can be in the same block of the first pass (equation (22) below). With these two rules, one can build the types of "legal" clusters that are allowed. I will show a few simple examples of cluster types which are legal and which are not.

The clusters will be named as c–α,β–χ,δ, where α is the number of blocks from the first pass that's involved in the cluster, β the number of hidden errors in each block of the first pass, χ is the number of blocks from the second pass that's involved in the cluster, δ the number of hidden errors in each block of the second pass. Some "legal" hidden error-clusters after two passes are illustrated in Fig. 6. In this figure, each hidden error bit is labeled and each row represents the hidden errors in a block of the first or the second pass. Notice that each block contains an even number of errors and no pair of error bits are repeated at the other pass. For example, the c-2,2-2,2 cluster is formed when the two error-pairs ([*a1*,*a2*], [*b1*,*b2*]) of the first pass swap partners and forms another two error-pairs ([*a1*,*b1*], [*a2*,*b2*]) of the second pass.

Table 6: Residue BER due to error-pair sticking. For n = 10k, input BER = 15%, $k_1 = 5$ and $k_2 = 10$.

x	p_x	p_s	p_r	$2p_xp_sp_r/k_2$
2	0.115	0.0015	1	3.46E-05
3	0.022	0.0045	1/3	6.62E-06
4	2.8E-03	0.0090	1	4.98E-06
5	2.4E-04	0.0150	3/5	4.28E-07
BER due to error-pair				4.67E-05

Figure 6: The type of error-clusters that can be formed from the first two passes. Each row represents the hidden error bits in a block. Notice that none of the hidden error-pairs of the second pass is also an error-pair from the first pass.



For counter examples, c-2,2-1,4, c-1,4-2,2, and c-2,4-2,4 are forbidden. However, c-2,2;1,4-4,2 is a legal hidden error-clusters (i.e. two hidden error-pairs and one four-hidden-errors block from the first pass).

2.2.3.2 cluster formation after two passes (small initial block size)

After eliminating error-pair sticking, the error cluster is the only way that error bits are left undetected after the CASCADE procedure. A formula for the probability of error-cluster formation is the ultimate definition of the reliability of QKD. However, the various types of legal clusters are too numerous. To simplify this problem, again, I will limit to the small initial block size k_1. In our example, there are 2000 blocks from the first pass. Out of 2000 blocks, there are $2000*[Bin(2,5,0.15) + Bin(3,5,0.15)] \sim 325$ blocks with two hidden errors and there are $2000*[Bin(4,5,0.15) + Bin(4,5,0.15)] \sim 4.5$ blocks with four hidden errors. Therefore, ignoring the probability of having four hidden errors in a block in our example introduces only a small error.

Recall that the number of hidden errors for the first two rounds are h_1 ($= p_1 n$) and h_2 ($= p_2 n$) respectively. Therefore, a hidden error bit from the first pass will have a probability p_2/p_1 to remain undetected after the second round. Now the probability of cluster formation is simplified to the problem of how to permute these $p_2 n$ bits to obtain a specific error-cluster pattern.

Cluster-2,2-2,2

The probability of forming cluster-2,2-2,2 can be viewed as two hidden error-pairs [*a1*,*a2*] and [*b1*,*b2*] from the first pass swap partners and form two hidden error-pairs [*a1*,*b1*] and [*a2*,*b2*] of the second pass (Fig. 6). As there are $p_2 n/2$ hidden error-pairs after the second round, there are $p_2 n/2$ ways to choose the first error-pair and named this pair as [*a1*,*b1*]. The probability that both *a2* and *b2* (hidden error-pair partners from the first pass) escape the detection from the BINARY of the second pass is $(p_2/p_1)^2$. Now we need the probability of *a2* and *b2* forming a hidden error-pair. Let's choose the location of *a2* first (the order of choosing *a2* or *b2* first does not change the probability). There is no constraint as *a2*'s can be in any of the $p_2 n/2$-1 error-pairs in pass two. The probability of *b2* being in the same error-pair with *a2* is $1/(p_2 n\text{-}3)$. But we also double count in the first choice of [*a1*,*b1*] pair (that is, we could have chosen [*a2*,*b2*] instead). We also need to convert the probability of cluster formation to the probability to BER by a factor $4/n$. So the BER due to the formation of c-2,2-2,2 is:

$$P(\text{c-2,2-2,2}) = p_2 n/2*(p_2/p_1)^2*(1/(p_2 n\text{-}3))*(4/n)*(1/2)$$

Cluster-3,2-3,2

Similar to the calculation of cluster-2,2-2,2, starting from choosing an error-pair ([*a1*,*b1*]) from $p_2 n/2$ error-pairs. Again, the probability that both *a2* and *b2* remain undetected after BINARY of the second pass is $(p_2/p_1)^2$. Now, *a2* and *b2* have to be in separate error-pairs of the second pass (otherwise it would produce Cluster-2,2-2,2). Let the error-pair partner

of the second pass for *a2* be *c1*. The probability that *c2*, the error-pair partners of *c1* from the first pass, remains undetected after BINARY of the second pass is (p_2/p_1). The probability of *c2* forming an error-pair of the second pass with *b2* is $1/(p_2n\text{-}5)$. But we also triple count in the first choice of [*a1*,*b1*] pair (that is, we could have chosen [*a2*,*c1*] *or* [*b2*,*c2*] instead). We also need to convert the probability to BER by a factor $6/n$. So the BER due to the formation of cluster-6 is:

$$P(\text{c-3,2-3,2}) = p_2n/2*(p_2/p_1)^3*(1/(p_2n\text{-}5))*(6/n)*(1/3)$$

Cluster-4,2-4,2

Starting by choosing [*a1*,*b1*] pair from $p_2n/2$ error-pairs, with the probability $(p_2/p_1)^2$ that *a2* and *b2* remain undetected after BINARY of the second pass. *a2* and *b2* have to be in separate hidden error-pairs of the second pass and let their error-pair partners be *c1* and *d1* (i.e. their error-pair partners cannot be an error-pair from the first pass, otherwise, c-3,2-3,2 is formed). The probability that both *c2* and *d2* remain undetected after BINARY of the second pass is $(p_2/p_1)^2$. And the probability that *c2* is in the same error-pair of the second pass with *d2* is $1/(p_2n\text{-}7)$. But we also quadruple count in the first choice of [*a1*,*b1*] pair. We also need to convert the probability to BER by a factor $8/n$. So the BER due to the formation of cluster-8 is:

$$P(\text{c-4,2-4,2}) = p_2n/2*(p_2/p_1)^4*(1/(p_2n\text{-}7))*(8/n)*(1/4)$$

The formula may be generalized to (with the above assumptions):

$$p_2' = p_2 * \sum_{y=2}^{\lfloor k_2/2 \rfloor} \left(\frac{p_2}{p_1}\right)^y \frac{1}{p_2n-2y+1} \qquad (21)$$

For our example, $p = 0.15$, $k_1 = 5$, $k_2 = 10$, $n = 10000$; $p_1 = 0.0668$, $p_2 = 0.0287$, $p_1n = 668$ and $p_2n = 287$, the probabilities of various type of clusters are shown in the third column of Table 7, corresponding to $p_2' = 3.18\text{E-}5$. This value compares well with the value derived from CASCADE tracking (3.3E-5, Table 5). I also ran simulation and obtained $p_2' < 4.4\text{E-}5$ using a better, but not perfect, interleaver. This interleaver is by doing randomization twice. The blocks of the second pass are constructed using the random interleaver like in the BS reconciliation procedure; after that, all blocks that contain pairs of bits from the same blocks of the first pass are found and these blocks are mixed and randomized again. Therefore, this interleaver reduces, but does not eliminate, the probability of error-pair sticking. Notice also that p_2' is approximately inversely proportional to the input size n (the last term in equation (21)) which was also confirmed by simulation.

In summary, the calculation of cluster formation probability demonstrates the importance of interleaver design. The close agreement among the probability of error-cluster formation, the BER from CASCADE tracking, and the BER from simulation demonstrate the analytical approaches to derive the reliability of QKD.

2.2.3.3 interleaver design and the minimum distance of a pair of bits

In this work, I did not find or implement a "perfect" interleaver. A series of questions about a perfect interleaver are still open. (How perfect is "perfect"? Is it actually possible? How to construct it?) In this section, I'll venture for a definition of a "perfect" interleaver.

The task is to construct K_i at the beginning of the ith pass. That is, to divide the n bits into $\lceil n/k_i \rceil$ blocks most efficiently using information of K_1, ..., K_{i-1}, where set $K_y = \{K_{y,1}, \ldots, K_{y,j}, \ldots\}$ is all blocks of the yth pass ($1 \le y < i$) after i-1 passes. Some $K_{y,j}$'s are smaller than k_y because BINARY breaks up some blocks into smaller blocks.

The first level of meaning for a perfect interleaver is that this interleaver will eliminate the possibility of an error-pair from any earlier pass to be placed at the same block of the current pass. For two passes, it means that any pair of bits $[x,y]$ in any block t of the second pass ($K_{2,t}$) cannot belong to any block s of the first pass ($K_{1,s}$). This is represented by the following equation:

$$\forall s,t \left(\forall [x,y] \in K_{2,t} \Rightarrow [x,y] \notin K_{1,s}\right) \tag{22}$$

Or, equivalently

$$\forall s,t \left(\forall [x,y] \in K_{1,s} \Rightarrow [x,y] \notin K_{2,t}\right) \tag{23}$$

For the third pass, it means that any pair of bits $[x,y]$ in any block u of the third pass cannot belong to any block t of the second pass nor belong to any block s of the first pass. This is represented by the following equation.

$$\forall s,t,u \left(\forall [x,y] \in K_{3,u} \Rightarrow [x,y] \notin K_{2,t} \cap [x,y] \notin K_{1,s}\right) \tag{24}$$

Or, equivalently

$$\forall s,t,u \left([x,y] \in K_{2,t} \cup [x,y] \in K_{1,s} \Rightarrow \forall [x,y] \notin K_{3,u}\right) \tag{25}$$

Table 7: Probability of error-cluster formation after two or three passes for the example case. (From equations (21) and (27)).

	clusters after 2 passes		clusters after 3 passes	
y	name	prob.	name	prob.
2	c-2,2-2,2	1.87E-05	c-2,2-2,2-2,2	6.75E-11
3	c-3,2-3,2	8.09E-06	c-3,2-3,2-3,2	2.22E-13
4	c-4,2-4,2	3.50E-06	c-4,2-4,2-4,2	~2E-15
5	c-5,2-5,2	1.52E-06	c-5,2-5,2-5,2	~2E-17
	total =	3.18E-05	total =	6.77E-11

With an interleaver satisfying properties (22) and (24), some types of still possible hidden error-clusters after three passes are shown in Fig. 7. Notice that the error bits' swapping from the first two passes is the same as clusters shown in Fig. 6, despite that the hidden error-pairs for the second pass are labeled differently. The hidden error-pairs of the third pass are obtained by swapping error-pairs again. For c-2,2-2,2-2,2, there is only one legal error-cluster of the third pass. For c-3,2-3,2-3,2, there are four legal clusters. For c-4,2-4,2-4,2, there are many legal clusters (not all of them shown here). The upper-left cluster is the base cluster for each error-cluster type. Other clusters are variations from the base cluster (the error-bits swapping shown in boldface).

The distance of a pair of bits is defined as one if they are in the same block and as infinity if they are not in the same block. Consider only one pass, any pair of bits has either distance of one or distance of infinity. Consider more than two passes, any pair of bits is directly related (in the same block in any of the passes), indirectly related (share a partner or partner of partner and so on), or unrelated (distance of infinity). The last case is a symptom of bad interleaver design so I will not consider that any further. If a pair of bits have a common partner then the distance is defined as two, as two blocks are needed to find their relationship. The number of blocks needed to establish the relationship of a pair of bits is defined as the distance of these two bits. Some of distance examples can be found in Fig. 7. The construction of a "perfect" interleaver takes into account of the pair-wise distance information. The minimum distance of a block of the ith pass is the minimum distance of any pair of bits within this block, utilizing the block structure information from all previous passes ($K_1, \ldots, K_{i-1}$).

For example, refer to Fig. 7, all the error-pairs of the second pass have distance of infinity as the distance of one is excluded by (22). The distances of error-pairs of the third pass are derived from the block structure of the first two passes. The distance of an error-pair *a1-b2* from the base cluster of c-2,2-2,2-2,2 is two because *a1-a2* are in the same block of pass one and *a2-b2* are in the same block of pass two. The shorthand notation, such as *a1-a2-b2,* is shown below the minimum distance for each cluster in Fig. 7, indicates how the minimum distance is derived.

A stronger requirement for a perfect interleaver is, in addition to the above, that it also eliminates the possibility of error-pairs swapping partner the second time. This rule will eliminate the formation of cluster with $D = 2$ (like c-2,2-2,2-2,2 in Fig. 7). This means that any pair of bits $[x,y]$ in any block i of the third pass can not share the same partner at the previous two passes. This is represented by the following property:

$$\forall s,t,u \left([x,y] \in K_{2,t} \cap [y,z] \in K_{1,s} \Rightarrow [x,z] \notin K_{3,u} \right) \tag{26}$$

The concept of distance is not new. The interleaver design for turbo-code (see discussion on page 37) uses similar concept of weight distribution [DD].

2.2.3.4 cluster formation after three passes (small initial block size)

The simplest way for two error-pairs to remain undetected after three passes is to swap partners for the second time at the third pass, as shown in Fig. **7**. The probability of cluster formation after two passes is already derived in Table 7. I'll extend the calculation of probability to the cluster formation after three passes. In section 2.2.3.2, the probability of cluster formation is calculated from the probability of hidden error bits surviving the round two. In this section, I start from some error-cluster patterns after pass two and calculate the probability of these error-pairs swap partners again.

Cluster-2,2-2,2-2,2

Consider the simplest error-cluster of two error-pairs, c-2,2-2,2-2,2 shown in Fig. **7**. According to property (22), error bits *a1* and *a2* have to be positioned in two separate blocks of the third pass. There is no other restriction for the positions of *a1* and *a2*. Error bit *b2* has to be in the same block with *a1* and the error bit *b1* has to be in the same block with *a2* (according to property (24)) to make c-2,2-2,2-2,2. The probability of positioning *b2* in the same block with *a1* is $(k_3\text{-}1)/(n\text{-}k_3\text{-}1)$; there are $n\text{-}k_3\text{-}1$ positions that *b2* can be in (one row occupies by [*a2*,*b1*], or k_3 positions, is not allowed), and only $k_3\text{-}1$ of these will produce the hidden error-pair. Likewise, the probability of positioning *b1* in the same block with *a2* is also $(k_3\text{-}1)/(n\text{-}k_3\text{-}1)$. Therefore,

$$P(\text{c-2,2-2,2-2,2}) = P(\text{c-2,2-2,2})*[(k_3\text{-}1)/(n\text{-}k_3\text{-}1)]^2$$

Cluster-3,2-3,2-3,2

The probability of forming one of the three error-pairs is $(k_3\text{-}1)/(n\text{-}2k_3\text{-}1)$, except now there are two rows that the second error bit is forbidden. The probability of swapping partners among these three error-pairs and forming one of the c-3,2-3,2-3,2 in Fig. **7** is:

$$P(\text{c-3,2-3,2})*[(k_3\text{-}1)/(n\text{-}2k_3\text{-}1)]^3$$

There are four clusters for c-3,2-3,2-3,2 as shown in Fig. **7**. Let me define a function ζ as function of y, where y = 2,3,4.... $\zeta(2) = 1$ and $\zeta(3) = 4$. For $y = 4$ or larger, $\zeta(y)$ has some large values. Therefore,

$$P(\text{c-3,2-3,2-3,2}) = \zeta(3)*P(\text{c-3,2-3,2}))*[(k_3\text{-}1)/(n\text{-}2k_3\text{-}1)]^3$$

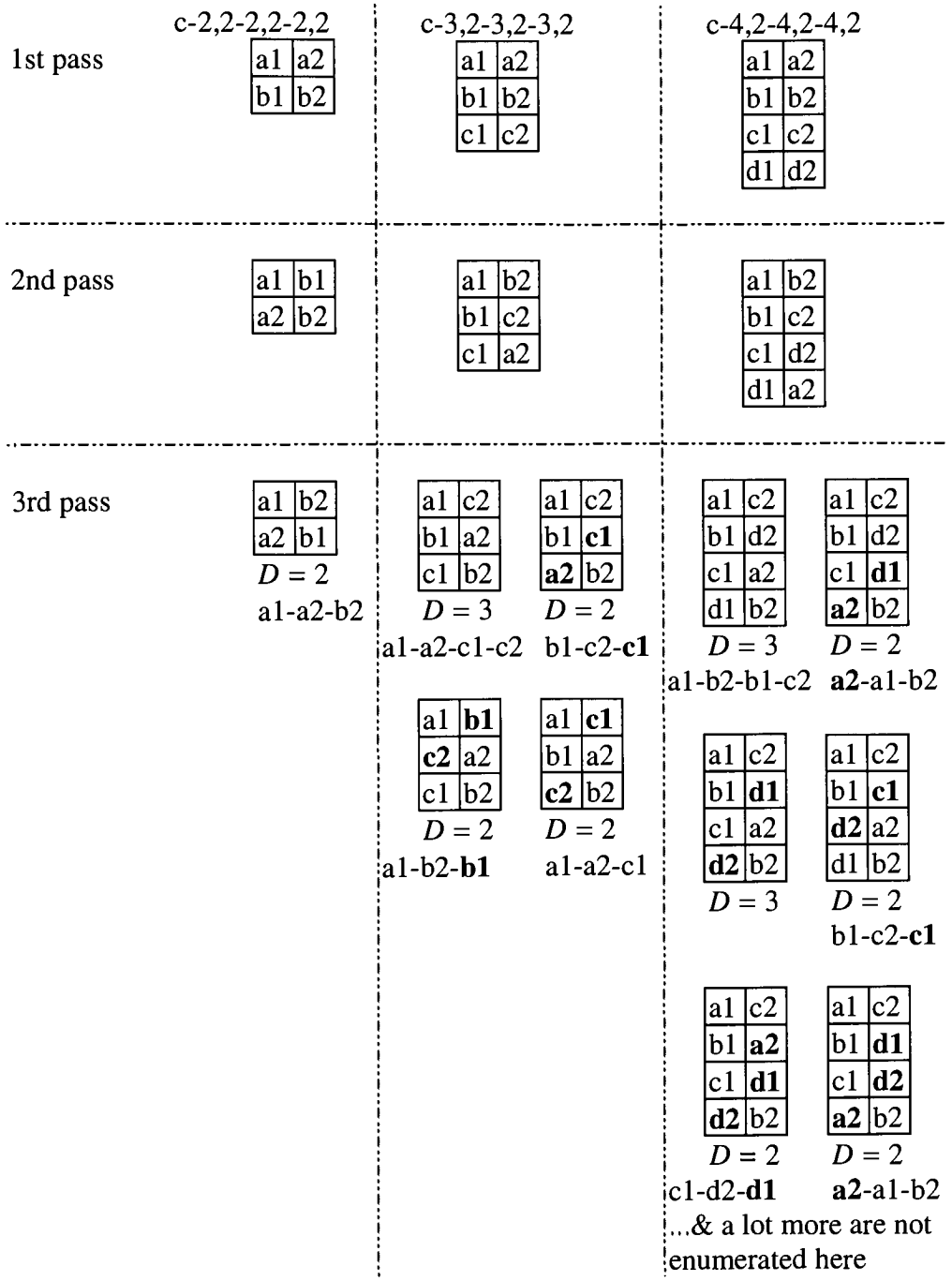
Cluster-4,2-4,2-4,2

Similarly:

$$P(\text{c-4,2-4,2-4,2}) = \zeta(4)*P(\text{c-4,2-4,2})*[(k_3\text{-}1)/(n\text{-}3k_3\text{-}1)]^4$$

Using equation (21), the generalized formula can be written as:

Figure 7: Some error-clusters, using an interleaver defined by condition (22) and (24), that can be formed after three passes. Each row represents the hidden error-pairs in a block in each pass. Each cluster from the third pass is labeled with the minimum distance of the cluster (see definition in text) and how the minimum weight is obtained. Condition (22) and (24) eliminate the possibility of cluster of the third pass with minimum distance $D = 1$.



$$p_3' = p_2 * \sum_{y=2}^{\lfloor k_2/2 \rfloor} \left(\frac{p_2}{p_1}\right)^y \frac{\zeta(y)}{(p_2 n - 2y + 1)} * \frac{(k_3 - 1)^y}{(n - (y-1)k_3 - 1)^y} \quad (27)$$

The probabilities of various clusters after three passes are shown in the last column of Table 7 and the output BER (p_3') after three passes will be about 6.75E-11. Notice that the probability of forming c-3,2-3,2-3,2 is much lower than the probability of forming c-2,2-2,2-2,2, in contrast to the smaller difference between c-3,2-3,2 and c-2,2-2,2. By using an interleaver that obeys property (26), p_3' will be (2.22E-13)/4 = 5.55E-14. However, I am not sure an interleaver obeying equation (22), (24), and (26) is achievable. Some thoughts on whether an interleaver design is achievable or not are included in Appendix.

The value for p_3' (6.75E-11) is drastically lower than the BER given by [BS] (1.7E-2) or by [BBBSS] (10^{-6})). This value is so low that using simulation to obtain BER estimate is not a good idea. Unfortunately, there is no independent verification of the accuracy of equation (27).

For better throughput, my analysis calls for a larger initial block size. The residue BER calculation has to take into account multiple error-pairs in a block, which needs future work to extend the above derivations.

2.3 Improvement of Reconciliation Procedure

To summarize the above observations, there are several ways to improve the BS reconciliation procedure in terms of less information leakage and lower output BER.

Improvement ideas:
1) Larger initial block size to increase throughput.
2) Better choice of interleaver, instead of randomization, to avoid or reduce error-pair sticking. This also eliminates the need of the fourth pass.
3) Some bits are already leaked or known to Eve in an earlier pass. There is no need to carry these bits to the later pass.

Most of the improvement in throughput comes from the selection of the initial block size (this work is done at the same block size schedule as in [BS], i.e., doubling block size after each pass). The upper half of Table 8 shows that throughput increases as k_1 increases (at the end of the third pass). The throughput tends to level off at large block size. Similar trend is observed for my improved procedure (the bottom half of Table 8).

By removing the leaked bits from the later passes, a small gain in throughput is realized. This can be seen by comparing the top and bottom half of Table 8 at the same initial block size. At $k_1 = 5$, there is 3% gain in throughput; at $16 \le k_1 \le 64$, the gain is 1%.

Because of error-pair sticking problem, the BS procedure shows significant residue BER at $k_1 = 32$ and even more at $k_1 = 64$ (upper half of Table 8). By using a better interleaver,

the error-pair sticking problem is eliminated at $k_1 = 32$ (bottom half of Table 8). However, the residue error pattern becomes a cluster of some kind at $k_1 = 64$. By using larger block size, such residue error-cluster disappeared from the few runs of simulation (it is removed or reduced). The error-cluster problem may be due to the imperfect interleaver used in this work. More elaborate interleaver design may help eliminate such problem.

The implementation of interleaver is tricky. It is easy to fall into the trap of laying bits in a three dimensional array of $k_1*k_2*k_3$ and taking the parity of the rows as the first pass, taking the parity of the columns as the second pass, and taking the parity of the third (depth) dimension as third pass. This is indeed easy to implement and faster in computation. However, this effectively reduces the input size of the problem from n to size of the array $k_1*k_2*k_3$. (n is much larger than $k_1*k_2*k_3$). As probabilities of forming various types of clusters are inversely proportional to input size, such implementation may actually increase the residue BER.

I also varied the initial block size from 5 to 32 in order to see whether the block size being power of two may perform better. The result seems to be monotonically increasing. Therefore, it is inconclusive whether the block size of power of two would do any good. I also tried the equal block size for all three passes and found the effect is also small, again inconclusive. Employing equal block size takes more CPU time. Therefore, I kept the block size schedule of doubling block size after each pass.

Table 8: Increasing initial block size of the BS procedure will improve the throughput significantly. The error-pair sticking is evident. By taking out the randomization procedure and removing leaked bits to the next pass increases throughput slightly. Yet, cluster formation is now a problem. Larger block size can be used when there is a larger batch of input bits.

BS procedure with varying block size, after the third pass, $p = 0.15$.

k1	n	# runs	% bits left	BER	# of error left
5	10k	220	26.3%	0	0
16	10k	176	32.5%	0	0
32	10k	155	32.8%	2.5E-06	2 runs with 1 error-pair
64	10k	85	33.0%	1.2E-05	5 runs with 1 error-pair

Improved reconciliation: w/o randomization and reduced carry-over.

k1	n	# runs	% bits left	BER	# of error left
5	10k	100	29.2%	0	0
16	10k	100	33.4%	0	0
32	10k	100	33.7%	0	0
64	10k	100	33.6%	2.8E-05	7 runs with cluster-4
64	10k	100	33.5%	1.4E-05	1 cluster-4 1 cluster-10
64	50k	100	33.9%	0	0

3 Discussion

There are four major areas that could further improve my current implementation or analysis: (given a set of input parameters (n, p, k_1, k_2, and k_3)).

1) Calculate the probability of hidden error-cluster formation after three passes by including the possibility of having more than four errors in a block. This is the ultimate analysis of reliability of the reconciliation procedure.
2) Answer the issues about the "perfect" interleaver. Whether it is possible? If so, how? If not, what is the lowest level of error-pair sticking?
3) Find the maximum initial block size given an aim output BER.
4) Find the optimal block size schedule has not been thoroughly examined.

One conclusion from this work is that the number of simulation runs needed to establish a good output BER estimate is impossibly huge. This is because the output BER is so low, a reliable estimate would require billions of runs. In addition, the output BER depends on the input batch size, which make the experimental space huge. Two approaches I outlined previously in calculating the probability of cluster formation and the method of CASCADE tracking, the latter I stopped at the second pass. These calculations so far are restricted to a small initial block size. Such calculation needs to be expanded and consider the possibility of more than four errors in a block. For the throughput issue of CASCADE tracking method, there is also a need to find a formula for number of parity checks needed to detect multiple error bits in a block. The optimal initial block size and block size schedule could be achievable after this formula is obtained.

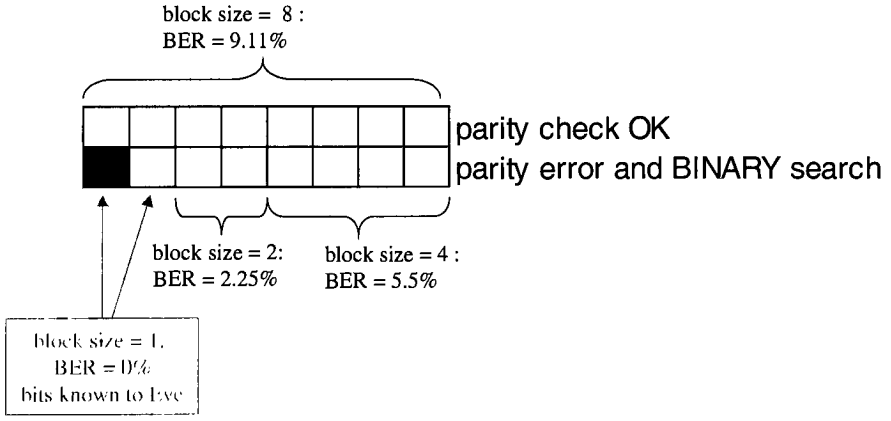
There are some possibilities that may further improve the reconciliation throughput. Figure 8 illustrates a basic analysis of parity checks and the BINARY routine on blocks (size $k_1 = 8$ and $p = 0.15$). A block without parity error (between Alice and Bob) will not go through BINARY and output BER $p_1 = 9.11\%$ (from equation (3)). A block with parity error will go through BINARY routine that will divide this block into blocks of smaller size. In this example ($k_1 = 8$), a block is divided into a block size of four, a block size of two, and two blocks of size one. After BINARY, the BER for the bits in block size of four is 5.5%, for block size of two is 2.25%, and for block size of one is 0% (totally reliable). There is one bit marked (gray cells in Fig. 8) in each block that will be discarded at the end of reconciliation. In the BS procedure, all these bits are carried to the second pass and all are treated equally. Mixing bits with different BER is a lossy process. There is definitely no point in carrying the bits from block size of one to the next pass, as they are already known to eve. This idea is already implemented in this work. One way to further increase throughput is to place bits with high BER in smaller blocks and bits with low BER in larger blocks.

Another way to improve throughput is to treat the marked bit differently. In this view, the white cells in Fig. 8 will be collected and interleaved to blocks of the second pass; BINARY and CASCADE will be done on these cells first. Depending on the number of errors found at this stage, the block size for the gray cells (marked bits) is then decided.

For example, during the BINARY routine of the white cells some errors bits are found to be coming from block size of two. Clearly the marked bit for that block is also an error bit. There is no need to carry that bit into the second pass of gray cells.

Finally, it is interesting to note that utilizing multi-dimensional parity checks is not new. The "product code", or two-dimensional parity check, was proposed by P. Elias in 1954 [E]. However, the revolutionary development of "turbo-code" started at the same time of BS procedure was developed (in 1993). Turbo-code is basically a convolution encoder that encodes parity checks twice, once without bits interleaving and once with bits interleaving. At the decoder end, an iterative decoding process significantly reduces the residue error rate. This iterative decoding processing in the turbo-code is similar to CASCADE in reconciliation procedure. The interleaver design may influence the output BER significantly. Currently, random permutation is typically employed in turbo-code mostly because the "optimal" interleaver is difficult to find except for a few simple cases [DP], and difference between random interleaver and the "optimal" interleaver tend to be small. The search for the two "optimal" interleavers for a turbo-code that encodes parities thrice is an even more difficult problem. Therefore, the analyses of "optimal" interleaver for turbo-code and for reconciliation procedure share similar difficulties.

Figure 8: The error rate of a bit after a parity check differs, depending on the block size for that bit.



4 Conclusions

Improvement of reconciliation procedure for QKD is established. The improvement in throughput is significant at high input BER. The major part of this improvement is due to more accurate residue error rate estimation, which leads to the selection of a larger initial block size. The minor contributions were obtained from using better interleaving strategy (instead randomization) and avoiding carrying to the next pass bits that are already lost. Only three passes are needed when a better interleaver is used.

The reliability of the key generated from QKD is also enhanced. More accurate residue error rate estimation is found. The current analysis shows drastically much lower residue error rate than the level given in [BS] or in [BBBSS]. The residue BER is lower with larger input batch size and smaller initial block size. As smaller initial block size also means lower throughput, the optimal initial block size should be chosen from the co-optimization of the throughput and the residue BER. However, in most situations, the throughput gain levels off at certain block size. Therefore, conservative choice of initial block size will ensure that the residue BER is low.

Further improvement of current work is needed in the search an optimal interleaver design, the initial block size selection rule, to analyze the probability of cluster formation when initial block size is large, and to figure out the number of parity checks needed to detect multiple error bits within a block. A few other ideas to further increase the throughput are also addressed. At last, similarity between CASCADE and the iterative decoding of "turbo code" in error control coding is noted.

5 Acknowledgement

I would like to thank Stanisław P. Radziszowski and Edith Hemaspaandra for encouragement in this study. And thanks to Harard Hempel for being reader of this work.

6 Appendix

The design of interleaver remains to be a challenge. An easier question is whether there is a possibility to achieve that when given a set of input parameters (n, k_1, k_2, and k_3). While I cannot answer that question directly, the following formula may provide a bit of insight.

Formula (28) is the probability that a randomly selected block happens to be a "perfect interleaver". For the second pass, one needs to select k_2 bits. For a random interleaver, the number of choices is $n*(n-1)*(n-2)*(n-3)$ ...$*(n-k_2+1)$. For the perfect interleaver, the number of choice is more limited: $n*(n-k_1)*(n-2k_1)*(n-3k_1)*$.....$*(n-(k_2-1)*k_1)$. Therefore, the probability of accidentally getting a perfect interleaver at the second pass is:

$$\frac{\prod_{i=0}^{k_2-1}(n-i*k_1)}{\prod_{i=0}^{k_2-1}(n-i)} \tag{28}$$

Similarly, the probability of accidentally getting an interleaver at the third pass that satisfies property (24) is:

$$\frac{\prod_{i=0}^{k_3-1}(n-i*(k_1+k_2-1))}{\prod_{i=0}^{k_3-1}(n-i)} \tag{29}$$

7 References

(In the order of the initial of surnames of all authors)

[ACL] M. Ardehali, H. F. Chau, and H.K Lo, "Efficient Quantum Key Distribution", quant-ph/9803007, http://xxx.lanl.gov/find/quant-ph.

[AES] For AES development, see http://csrc.nist.gov/encryption/aes/.

[B92] C. H. Bennett, "Quantum Cryptography Using Any Two Nonorthogonal States", Phys. Rev. Lett. **68**, 3121, 1992.

[B98] D. Bruβ, "Optimal eavesdropping in quantum cryptography with six states", Technical Report, quant-ph/9805019, http://xxx.lanl.gov/find/quant-ph.

[BB84] C.H. Bennett and G. Brassard, "Quantum Cryptography: Public Key Distribution and Coin Tossing", in Proceedings of IEEE international Conference on Computers, Systems and Signal processing, Bangalore, India (IEEE, New York), 175, 1984.

[BB89] C.H. Bennett and G. Brassard, "The dawn of a new era for quantum cryptography. The experimental prototype is working!" SIGACT News, **20**, vol. 4, 78-82, 1989.

[BBBSS] C.H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental Quantum Cryptography", J. of Cryptography, **5**, no. 1, 3-28, 1992 (Eurocrypt '90).

[BBCM] C. H. Bennett, G. Brassard, C. Crepeau, U. M. Maurer, "Generalized Privacy Amplification", IEEE trans. Info. Theory, **41**, no. 6, 1915-1923, 1995.

[BBR85] C. H. Bennett, G. Brassard, J. M. Robert, "How to reduce your enemy's information" Advances in Cryptology, Crypto '85 Proceedings. 468-376, 1985.

[BBR88] C. H. Bennett, G. Brassard, J. M. Robert, "Privacy amplification by public discussion", SIAM Journal on Computing, **17**, 210-229, 1988.

[BHKLL] W.T. Buttler, R. J. Hughes, P. G. Kwiat, S. K. Lamoreaux, G.G. Luther, G.L. Morgan, J.E. Nordholt, C. G. Peterson, and C.M. Simmons, "Practical free-space quantum key distribution over 1 km", Technical report, quant-ph/9805071, 1998.

[BLTO] C. Brunel, B. Lounis, P. Tamarat, and M. Orrit, "Triggered Source of Single Photons Bases on Controlled Single Molecule Fluorescence", Physical Review Letters, **83**, no. 14, 2722-2725, 1999.

[BS] G. Brassard and L. Salvail, "Secret-Key Reconciliation by Public Discussion", Advances in Cryptology, Eurocrypt., 410, 1993.

[CL] H. F. Chau, and H.K Lo, "Unconditional Security of Quantum Key Distribution Over Arbitrarily Long Distances", quant-ph/9803006, http://xxx.lanl.gov/find/quant-ph.

[CW] J. L. Carter, and M. N. Wegman, "Universal classes of hash functions", Journal of Computer & System Science, **2**, 143, 1979.

[DD] S. Dolinar and D. Divsalar, "Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations", JPL TDA progress Report, **42-122**, 56-65, 1995.

[DP] D. Divsalar and F. Pollara, "Multiple Turbo Codes for Deep-Space Communications", JPL TDA progress Report, **42-121**, 66-77, 1995.

[E] P. Elias, "Error-Free Coding," IRE Transactions on Information Theory, PFIT-4, 29-37, Sept, 1954.

[E91] A. K. Ekert, Quantum cryptography based on Bell's theorem. Physical Review Letters, **67**, no. 6, 661-663, 1991.

[FGGNP] C. A. Fuchs, N. Gisin, R. B. Griffiths, C. S. Niu, and A. Peres, "Optimal eavesdropping in quantum cryptography. I. Information bound and optimal strategy", Physical Review A, **56**, no. 2, 1163-1172, 1997.

[HMP] R. J. Hughes, G. L. Morgan, and C. G. Peterson, " Practical Quantum Key Distribution over 48-km Optical Fiber Network", Technical report, quant-ph/9904038, 1999.

[KBKY] J. Kim, O. Benson, H. Kan, and Y. Yamamoto "A Single-Photon Turnstile Device", Nature, **397**, 500-503, 1999.

[L99] S. J. Lomonaco, Jr. " A Talk on Quantum Cryptography or How Alice Outwits Eve", Proceeding of the Coding theory, Cryptology, Number Theory Conference at US Navy Academy in Annapolis, Maryland, Springer-Verlag, http://www.cs.umbc.edu/~lomonaco/Publications.html.

[M] D. C. Mayers, "Quantum Key Distribution and String Oblivious Transfer in Noisy Channels", Technical report, quant-ph/9606003, 1996.

[MT] C. Marand and P. D. Townsend, "Quantum Key distribution over the distance as long as 30 km", Optics Letters, **20**, 1695-1697, 1995.

[MY] D. C. Mayers and C. C. Yao, "Unconditional security in quantum cryptography", Technical report, quant-ph/9802025, 1998.

[MYa] D. C. Mayers and C. C. Yao, "Quantum cryptography with imperfect apparatus", Technical report, quant-ph/9809039, 1998.

[MZG] A. Muller, H. Zbinden, N. Gisin, "Underwater quantum coding", Nature, **378**, 449-449, 1995.

[RGGGZ] G. Ribordy, J Gautier, N. Gisin, O. Guinnard, H. Zbinden, "Automated plug and play Quantum Key Distribution", Electronics Letters, **34**, no. 22, 2116-2117, 1998.

[RP] E. Rieffel and W. Polak, "An Introduction to Quantum Computing for Non-Physicists", Technical report, quant-ph/9809016, 1998.

[S94] P. W. Shor. "Algorithms for quantum computation: discrete log and factoring", In Proceedings of 35th IEEE FOCS, 124-134, 1994.

[SRSF] B. Slutsky, R. Rao, , P.C. Sun, and Y. Fainman, "Security of quantum cryptography against individual attacks", Phys. Rev. A, **57**, no. 4, 2383-2398, 1998, http://kfir.ucsd.edu/Pubs/abstracts.shtml.

[SSMRF] B. Slutsky, P.C. Sun, Y. Mazurenko, R. Rao, and Y. Fainman, "Effect of channel imperfection on the secrecy capacity of a quantum cryptographic system", J. Mod. Opt., **44**, no. 5, 953-961, 1997, http://kfir.ucsd.edu/Pubs/abstracts.shtml.

[W75] A.D. Wyner, "The Wire-Tap Channel", The Bell System Technical Journal, **54**, no. 8, 1355-1387, 1975.

[W83] S. J. Wiesner, "Conjugate coding", SIGACT News, **15**, 78-88, 1983.

[WC] M. N. Wegman and J.L. Carter, "New has function and their use in authentication and set equality", Journal of Computer & System Sciences, **22**, 265, 1981.