Secret-Key Reconciliation by Public Discussion

Gilles Brassard * and Louis Salvail **

Département IRO, Université de Montréal
C.P. 6128, succursale "A",Montréal (Québec), Canada H3C 3J7.
e-mail: {brassard}{salvail}@iro.umontreal.ca.

Abstract. Assuming that Alice and Bob use a secret noisy channel (modelled by a binary symmetric channel) to send a key, reconciliation is the process of correcting errors between Alice's and Bob's version of the key. This is done by public discussion, which leaks some information about the secret key to an eavesdropper. We show how to construct protocols that leak a minimum amount of information. However this construction cannot be implemented efficiently. If Alice and Bob are willing to reveal an arbitrarily small amount of additional information (beyond the minimum) then they can implement polynomial-time protocols. We also present a more efficient protocol, which leaks an amount of information acceptably close to the minimum possible for sufficiently reliable secret channels (those with probability of any symbol being transmitted incorrectly as large as 15%). This work improves on earlier reconciliation approaches [R, BBR, BBBSS].

1 Introduction

Unlike public key cryptosystems, the security of quantum cryptography relies on the properties of the channel connecting Alice and Bob. Physical imperfections in a quantum channel introduce noise into the messages passing through it. The presence of an eavesdropper wiretapping the channel disrupts communication even more. Assuming that Alice and Bob are using a quantum channel or any noisy channel to send a secret key, they would need to reconcile their keys to make them identical. Reconciliation is the process of finding and correcting discrepancies between the secret key sent by Alice and the one received by Bob. This is done by public discussion. In this paper we focus on secret noisy channels modelled by binary symmetric channels.

An eavesdropper can gain information about the secret key both by wiretapping the quantum channel and by listening to the public reconciliation. This information can be eliminated using a privacy amplification protocol [R, BBR] at the cost of reducing the secret key size proportionally to the amount of information potentially known by an eavesdropper. Since using a quantum channel

* Supported in part by NSERC's E. W. R. Steacie Memorial Fellowship and Québec's FCAR.
** Supported by an NSERC scholarship.

is expensive, we would like to minimize the information that a reconciliation protocol divulges.

A quantum public key distribution protocol is described in [BBBSS], which also discusses a way to combine together reconciliation and privacy amplification. The problem of reconciliation has been previously studied in [R, BBR, BBBSS]. Key distribution using independent channels [M] also requires reconciliation.

In section 3 we define the problem and introduce the notion of optimality; in section 4 we show how to construct optimal protocols. In section 5 we discuss efficiency; in sections 6 and 7 we present protocols that can be used in practice.

2 Preliminaries

Let $\{p(x)\}_{x \in X}$ be a probability distribution over a finite set X. The entropy of X, denoted $H(X)$, is defined as

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

(where all logarithms are to the base 2). In particular, $H(X)$ is the expected value of the number of bits required to specify a particular event in X. It is easy to observe that

$$H(X) \leq \log |X|$$

with equality iff $p(x) = 1/|X|$ for each $x \in X$. When X is a Bernoulli trial with parameter p, we denote $H(X)$ by $h(p)$.

Given two sets X and Y and a joint probability distribution $\{p(x, y)\}_{x \in X, y \in Y}$, the conditional entropy $H(X|Y)$ is defined as

$$H(X|Y) = -\sum_{y \in Y} \sum_{x \in X} p(y) p(x|y) \log p(x|y).$$

A binary symmetric channel (BSC) permits transmission of a string of bits, each independently exposed to noise with probability p. Let A be the string sent by Alice and let B be the one received by Bob. If each bit from string A is randomly and independently chosen, it is clear that

$$H(A) = |A|.$$

The conditional entropy of A given B is

$$H(A|B) = H(A \oplus B) = nh(p)$$

where $n = |A| = |B|$. Henceforth we will denote a binary symmetric channel with parameter p by $BSC(p)$.

The quantum channel is an example of a secret binary symmetric channel even if an eavesdropper introduces noise in a non-symmetric way. Before Alice and Bob reconcile their strings, they perform a sample of the transmitted bits to estimate the error rate of the actual quantum communication. If the estimate is acceptably close to the expected error rate of the channel then they publicly and randomly permute their respective string on which reconciliation is then applied. Bob's string can now be assumed to be the result of a transmission over a BSC. If the estimate is not sufficiently close to the expected error rate then they rule out the communication and retry later (consult [BBBSS] for more details).

For $0 \leq \lambda \leq \frac{1}{2}$ the tail inequality is

$$\sum_{k=0}^{\lfloor \lambda n \rfloor} \binom{n}{k} \leq 2^{nh(\lambda)}.$$

For X a random variable with finite variance $V(X)$ and expected value $E(X)$ and for $a > 0$ the Chebyshev inequality is

$$prob(|X - E(X)| \geq a) \leq \frac{Var(X)}{a^2}.$$

The Hamming distance $dist(A, B)$ between A and B is the number of places in which A and B differ. The weight $w(A)$ of A is its number of nonzero positions.

3 The Problem

Suppose there is a secret channel between Alice and Bob on which Alice transmits to Bob an n-bit string A such that $H(A) = n$. We model the secret channel with a $BSC(p)$ for some p. Bob then receives an n-bit string B such that $H(A|B) = nh(p)$. Using public discussion Alice and Bob want to share an n-bit secret string S obtained from A and B. This can be done assuming an unjammable public channel without making any other security assumptions. Our goal is to find protocols minimizing the information on S that an eavesdropper with unlimited computing power can get by listening on the public channel.

A reconciliation protocol R^p is defined by Alice's and Bob's algorithms. R^p runs on strings A and B to produce string S by exchanging some information Q on the public channel. This will be denoted by $R^p = [S, Q]$ or by $R^p(A, B) = [S, Q]$ when a specific A and B are considered. If the protocol fails to produce $S \in \{0, 1\}^n$ we will write $S = \perp$. The amount of *leaked information* $I_E(S \mid Q)$ is the expected amount of Shannon information that an eavesdropper E can get on S given Q.

Definition 1. A reconciliation protocol R^p is ε*–robust*, $0 \leq \varepsilon \leq 1$, if

$$(\exists N_0(\varepsilon))(\forall n \geq N_0(\varepsilon)) \sum_{\alpha,\beta \in \{0,1\}^n} prob(A = \alpha, B = \beta) prob(R^p(\alpha, \beta) = [\perp, \cdot]) \leq \varepsilon.$$

3.1 Optimality

It is easy to define the optimality property of reconciliation protocols. Theorem 2 is a direct consequence of an elementary result in information theory, namely the noiseless coding theorem.

Theorem 2. $(\forall p \leq \frac{1}{2})(\forall$*reconciliation protocol* $R^p)$ *if there exists* $0 \leq \varepsilon < 1$ *such that* $R^p = [S, Q]$ *is* ε*–robust then*

$$\lim_{n\to\infty} \frac{I_E(S \mid Q)}{nh(p)} \geq 1$$

where n *is the length of the transmitted string.*

A reconciliation protocol is optimal if the expression in theorem 2 is an equality.

Definition 3. A protocol R^p is *optimal* if

$$(\forall\varepsilon > 0)\,[R^p = [S, Q] \text{ is } \varepsilon\text{–robust}]$$

and

$$\lim_{n\to\infty} \frac{I_E(S \mid Q)}{nh(p)} = 1$$

where the public channel is a BSC(p).

The next section shows how to construct a family of optimal protocols.

4 Optimal Protocols

One way of constructing optimal protocols is to associate a random label of length approximately $nh(p)$ bits to each n-bit string. Alice tells Bob the label of her string A over the public channel. In order to decode string B, Bob computes a string B' of minimal Hamming distance from B, having the same label as A. Protocol 1 implements this process.

Obviously such a protocol is useless in practice since it requires Alice and Bob to choose randomly among 2^{m2^n} functions.

Protocol 1

1. Alice and Bob choose a random function among all functions from $\{0,1\}^n \rightarrow \{0,1\}^m$, where m is a parameter to be determined. The description of this function is exchanged publicly.
2. Alice sends $f(A)$ to Bob on the public channel.
3. Bob decodes his string resulting in string B' such that $dist(B, B')$ is minimal over all strings D such that $f(D) = f(A)$.

The proof of theorem 4 is similar to earlier ones showing the Shannon noisy coding theorem for BSC (see [W]).

Theorem 4. *Protocol 1 is optimal for an adequate choice of parameter m.*

Proof (sketch). Let p be the BSC parameter. Let p_e be the decoding error probability. Let $C = A \oplus B$ and E be the event associated with a decoding error. Clearly $w(C) \approx Bin(n,p)$[3]. A simple counting argument shows that

$$prob(\neg E \mid w(C) \leq r) \geq (1 - 1/2^m)^{\sum_{j=1}^{r} \binom{n}{j}}.$$

For correct decoding to occur it is sufficient that all the strings B' such that $dist(B', B) \leq dist(B, A)$ with $B' \neq A$ be distributed among those $2^m - 1$ elements of the image of f that are not equal to $f(A)$.

The decoding error probability is:

$$\begin{aligned} p_e &= prob(w(C) \leq r) prob(E \mid w(C) \leq r) \\ &\quad + prob(w(C) > r) prob(E \mid w(C) > r) \\ &\leq prob(E \mid w(C) \leq r) + prob(w(C) > r). \end{aligned}$$

Let $r = \lfloor np + n\varepsilon_n \rfloor$ and $\varepsilon_n = 1/\log n$. The Chebyshev inequality gives

$$\begin{aligned} prob(w(C) > r) &= prob(w(C) > \lfloor np + n\varepsilon_n \rfloor) \\ &\leq prob(\mid w(C) - np \mid \geq n\varepsilon_n) \\ &\leq \frac{p(1-p)}{n\varepsilon_n^2} \\ &= \frac{(\log n)^2 p(1-p)}{n}. \end{aligned}$$

If $w(C) \leq r$, the decoding probability error is bounded by:

$$\begin{aligned} prob(E \mid w(C) \leq r) &\leq 1 - prob(\neg E \mid w(C) \leq r) \\ &\leq 1 - (1 - 1/2^m)^{\sum_{j=1}^{r} \binom{n}{j}}. \end{aligned}$$

For $m = \lceil \log \delta_n + nh(p + \varepsilon_n) \rceil$ and $\delta_n = \lceil \log n \rceil$ we obtain:

$$prob(E \mid w(C) \leq r) \leq 1 - \left(\frac{1}{e}\right)^{2^{-m} \sum_{j=0}^{r} \binom{n}{j}}.$$

By the tail inequality the decoding probability error is bounded by:

$$prob(E \mid w(C) \leq r) \leq 1 - e^{-2^{nh(p+\varepsilon_n) - m}}.$$

When $n \to \infty$ we have $p_e = 0$, and protocol 1 is ε–robust for all $\varepsilon > 0$. It is easy to see that the resulting amount m of leaked information is asymptotically equal to $nh(p)$. □

To solve the problem that choosing a random function from a huge set is unreasonable, we choose it from a universal$_2$ class of hash functions [CW].

[3] $Bin(n,p)$ is the binomial probability distribution.

Definition 5 ([CW]). Let H be a class of functions from F to G. We say that H is universal$_2$ if for all $x, y \in F$ such that $x \neq y$, the number of functions f in H such that $f(x) = f(y)$ is less or equal than $\#H/\#G$.

Wegman and Carter [CW, WC] also describe classes for which choosing and evaluating functions can be achieved efficiently. The following theorem shows that choosing f among a universal$_2$ class ensures the protocol's optimality.

Theorem 6. *Protocol 1 is optimal for an adequate choice of parameter m if Alice and Bob choose f among an universal$_2$ class of functions.*

Proof. In order to bound the decoding error probability we must bound the probability that:

$$f(C) \neq f(x_1) \wedge f(C) \neq f(x_2) \wedge \ldots \wedge f(C) \neq f(x_l)$$

for $C \in F$, $x_i \in F$ where $x_i \neq C$ for each i. From definition 5, for any $C \in F$ and $x_1 \in F$ the following is true:

$$\#\{f \in H \mid f(C) = f(x_1)\} \leq \frac{\#H}{\#G}.$$

Therefore, the number of functions where $f(C) \neq f(x_1)$ is greater than $\#H(1 - 1/\#G)$. Among the $\#H - \#H/\#G$ remaining functions, there are at most $\#H/\#G$ functions such that $f(C) = f(x_2)$. Applying this argument over the l points in the domain of f gives $\#\{f \in H \mid f(C) \neq f(x_i) \text{ pour } 1 \leq i \leq l\} \geq \#H(1 - l/\#G)$. Let $X(r) = \{x \in \{0,1\}^n \mid w(x) \leq r \text{ et } x \neq C\}$ be the set of strings of weight r or less. Similar to the proof of theorem 4 we have:

$$\begin{aligned} prob(E \mid w(C) \leq r) &= 1 - prob(\neg E \mid w(C) \leq r) \\ &\leq \frac{\#X(r)}{\#G}. \end{aligned}$$

The proof follows by setting $F = \{0,1\}^n$, $G = \{0,1\}^m$ for r, ε_n, δ_n and m set as in theorem 4 (since $\#X(r) \leq 2^{nh(p+\varepsilon_n)}$). □

Thus we have a way to automatically generate optimal reconciliation protocols by specifying a universal$_2$ class in a short and efficient way. The problem with this approach is that there are no known efficient algorithms for Bob to compute the decoded string B'.

5 Efficiency

Finding a class of functions for which protocol 1 is optimal and such that Alice and Bob can reconcile efficiently is comparable to finding efficient decodable error correcting codes. This is due to similarities between these two problems when a non-interactive protocol such as protocol 1 is being considered. The non-interactive scheme is relevant for some applications such as quantum oblivious transfer [BBCS]. We will see that using H_3 (defined below, for more details consult [CW]) yields a decoding time complexity equivalent to that of solving the general problem of decoding linear codes.

Definition 7. An R^p reconciliation protocol is:

1. *efficient* if there is a polynomial $t(n)$ such that $\overline{T}^{R^p}(n) \leq t(n)$ for n sufficiently large, where n is the length of the strings transmitted over the secret channel;
2. *ideal* if it is both optimal and efficient.

where $\overline{T}^{R^p}(n)$ represent the expected running time of R^p given an n-bit long input string.

Theorem 8 gives an (unlikely) hypothesis that is necessary and sufficient for protocol 1 to be ideal when used along with class H_3. Recall that H_3 is the set of functions $f : \{0,1\}^n \rightarrow \{0,1\}^m$ whose i^{th} bit $f_i(x)$ of $f(x)$ is such that $f_i(x) = \bigoplus_{j=1}^{n} \lambda_{i,j} x_j$ for $\lambda_{i,j} \in \{0,1\}$. When m and n are fixed, choosing f consists of choosing $\lambda_{i,j} \in_R \{0,1\}$ for all $i \in [1..m]$ and $j \in [1..n]$.

Theorem 8. *Protocol 1 used with the universal$_2$ class of functions H_3 is ideal if and only if* $\mathbf{NP} \subseteq \mathbf{BPP}$.

Proof. If C is a class of decision problems, let C^* denote the class of problems that are polynomially equivalent to some problem in C. Let X be the problem of executing step 3 of protocol 1 and let X^{H_3} be the same problem when H_3 is used. [BMT] shows that determining least-weight solution in a system of linear equations in $GF(2)$ is $\mathbf{NP}$–hard. This problem is equivalent to X^{H_3}. Moreover we can easily show that $X \in (\sum_2^p)^*$. We want to show that $X^{H_3} \in \mathbf{BPP}^* \Leftrightarrow \mathbf{NP} \subseteq \mathbf{BPP}$. The left to right direction of this statement is obviously true since X^{H_3} is $\mathbf{NP}$–hard. To prove the other direction we use a result of [Z] showing that $\mathbf{NP} \subseteq \mathbf{BPP} \Leftrightarrow \mathbf{PH} \subseteq \mathbf{BPP}$ combined with the fact that $X \in (\sum_2^p)^*$. □

6 Almost-Ideal Protocols

To be useful in practice a reconciliation protocol need not be optimal. Before execution of the protocol, Alice and Bob can agree on an arbitrarily small amount of additional information relative to the theoretical bound, which they are willing to reveal during execution. If the resulting protocol is also efficient then we have a protocol that might be useful in practice. For example, if the bits exchanged over the secret channel are costly, then it might be better to spend more time computing during reconciliation, thus saving some more of these costly bits.

We will formalize the latter property, and construct in section 6.2 a family of protocols that satisfy it.

6.1 Definition

An almost-ideal protocol has an error probability approaching 0 as the length of the strings increases, but its amount of leaked information is allowed to be slightly greater than the theoretical bound. Alice and Bob indicate it by choosing a parameter ζ. They must be able to set ζ such that the corresponding amount of leaked information is as close to the theoretical bound as they wish. Once the parameter ζ is set, the expected running time $\overline{T}^{R^p_\zeta}$ of protocol R^p_ζ must be bounded by a polynomial.

Definition 9. A reconciliation protocol R^p_ζ is *almost-ideal* if for all $\zeta > 0$ we have

1. $(\forall \varepsilon > 0) \left[R^p_\zeta = [S, Q] \text{ is } \varepsilon\text{–robust} \right]$
2. $\lim_{n \to \infty} \dfrac{I_E(S \mid Q)}{nh(p)} \leq 1 + \zeta$ and
3. $(\exists \text{ polynomial } t)(\exists N_0(\zeta))(\forall n \geq N_0(\zeta)) \left[\overline{T}^{R^p_\zeta} \leq t(n) \right]$

for n the length of the strings transmitted over the BSC(p).

6.2 Protocol Shell

Protocol **Shell** uses interaction on the public channel to correct efficiently the secret strings by dividing them into blocks of fixed length. If the blocks have been corrected using a subprotocol with a decoding error probability δ_k (for blocks of length k), then **Shell** will correct them by producing an additional amount of leaked information proportional to δ_k and $\frac{1}{k}$.

Shell is constructed from a few simple interactive primitives described in the following sections.

`BINARY`. When strings A and B have an odd number of errors, then Alice and Bob can perform an interactive binary search to find an error by exchanging fewer than $\lceil \log n \rceil$ bits over the public channel in the following manner:

1. Alice sends Bob the parity of the first half of the string.
2. Bob determines whether an odd number of errors occurred in the first half or in the second by testing the parity of the first half and comparing it to the parity sent by Alice.
3. This process is repeatedly applied to the half determined in step 2. An error will be found eventually.

The reconciliation protocol described in [BBBSS] uses `BINARY` as the main primitive.

`CONFIRM`. If the strings of Alice and Bob are different, `CONFIRM` tells them with probability $\frac{1}{2}$; if they are identical, `CONFIRM` says so with probability 1.

1. Alice and Bob choose a random subset of corresponding bits from their strings.
2. Alice tells Bob the parity of her subset.
3. Bob checks that his subset has the same parity.

They can apply this process k times to convince themselves that their strings are identical. This test will fail with probability 2^{-k}.

BICONF^s. Combining BINARY and CONFIRM gives us another primitive that can correct several errors. BICONF^s runs CONFIRM s times. Each time CONFIRM shows a subset for which Alice's and Bob's string have different parities they run BINARY on this subset and thus correct an error. Let $\Delta^s(l|e)$ be the probability that BICONF^s corrects l errors if there are e errors. We have:

$$\Delta^s(l \mid e) = \begin{cases} \binom{s}{l} 2^{-s} & \text{if } l \neq e \\ \sum_{j=e}^{s} \binom{j-1}{e-1} 2^{-j} & \text{if } l = e. \end{cases} \tag{1}$$

Shell. Protocol **Shell** runs a basic protocol R^k_{bas} on blocks of size k. First Alice and Bob divide their strings into k-bit long "primary" blocks. In pass 1 they apply R^k_{bas} followed by BICONF^1 on each of these blocks. If BICONF^1 detects an error, all bits from Bob's corresponding block are set to equal Alice's bits. In pass s ($s > 1$), Alice and Bob join pairs of adjacent blocks from the preceding pass to form the blocks for the current pass. On each of these new blocks they execute BICONF^1 s times. When an error is found, the primary block containing the erroneous bit is replaced by Alice's corresponding primary block. The process is repeated until there is only one block at the current pass (at pass $s = \lceil \log \frac{n}{k} \rceil$).

6.3 An Almost Ideal Protocol

Suppose that R^k_{bas} has an error probability $\delta_k \leq 1/2$ and leaked an amount I^k_{bas} of information such that $\frac{I^k_{bas}}{kh(p)} = \tau$. Executing BICONF^1 s times in pass s (i.e. on blocks of size $2^{s-1}k$) is equivalent to the execution of BICONF^s with respect to the distribution of the number of erroneous primary blocks that will be corrected. Therefore, if there are e erroneous primary blocks in a current block during pass s, the probability of correcting l of these is $\Delta^s(l|e)$ as defined by equation 1. Let $p'_s(e)$ be the probability of having e erroneous primary blocks in a current block after completion of pass s, and let $p_s(e)$ be the same probability before execution of pass s. It follows that

$$p'_s(e) = \sum_{j=e}^{e+s} p_s(j)\Delta^s(j-e \mid j)$$

with

$$p_s(e) = \sum_{j=0}^{e} p'_{s-1}(j)p'_{s-1}(e-j).$$

Moreover

$$p'_1(0) = 1 - \frac{\delta_k}{2}, p'_1(1) = \frac{\delta_k}{2}.$$

We can prove the following theorem by induction on s and e (see [Sa] for the proof):

Theorem 10. $(\forall \delta_k \leq \frac{1}{2})(\forall e > 0)(\forall s \geq 1)\left[p'_s(e) \leq \frac{1}{2^{s+e-1}}\right]$.

The failure probability is less than $\frac{k}{n}$ and tends to 0 as n increases. Hence the condition 1 from the definition of an almost ideal protocol is met. A bound on the amount of information $I_S(n)$ leaked by **Shell** can be obtained using theorem 10:

$$I_S(n) \leq nh(p)\tau + \delta_k n + \frac{4n(\lceil \log k \rceil + 2)}{k}$$

and we have that

$$\lim_{n\to\infty} \frac{I_S(n)}{nh(p)} \leq \tau + \frac{\delta_k}{h(p)} + \frac{4(2 + \lceil \log k \rceil)}{kh(p)}.$$

If protocol 1 is used as a basic protocol then there is a k for which $\frac{\delta_k}{h(p)} + \frac{4(2+\log k)}{kh(p)} \leq \varepsilon$ for any $\varepsilon > 0$. In addition the amount I^k_{bas} of leaked information can be chosen such that $\frac{I^k_{bas}}{kh(p)} \leq \tau$ for all $\tau > 1$. It is clear that **Shell** works in polynomial time for any fixed k.

Corollary 11. *If we use protocol 1 as a basic protocol then* **Shell** *is almost-ideal.*

The size of the blocks grows too fast as the amount of leaked information approaches the theoretical bound for **Shell** to be used together with protocol 1 in an efficient implementation. However it is possible to use **Shell** with other types of basic protocol, such as a systematic error-correcting code. For example, Alice and Bob can agree on a systematic $(N, k')-$code [4] of distance d. Afterwards they can compute the bound on the amount of leaked information produced by **Shell** on the primary blocks of size $k = tk'(t > 0)$, with $\tau = \frac{N-k'}{k'h(p)}$ and $\delta_k = 1-(1-\varepsilon)^t$, where ε is the probability that a k'-bit-long block holds more than $\lfloor \frac{d}{2} \rfloor$ errors. However **Shell** is no longer almost-ideal with this type of basic protocol.

7 A Practical Protocol

In section 7.1 we present protocol **Cascade**, which can easily be implemented. **Cascade** leaks an amount of information close to the theoretical bound on a $BSC(p)$ when p is as big as 15%. In section 7.2 a rough analysis shows how to choose protocol parameters for which the error probability decreases exponentially fast as a function of the number of passes. We also give a table comparing the amount of information leaked empirically by **Cascade** (given these parameters) to the theoretical bound.

7.1 Cascade Description

Cascade proceeds in several passes. The number of passes is determined by Alice and Bob before execution. This choice is related to parameter p. Let $A = A_1, \ldots, A_n$ and $B = B_1, \ldots, B_n$ (with $B_i, A_i \in \{0, 1\}$) be Alice's and Bob's strings respectively.

[4] An $(N, k')-$code has $N-$bit codewords to encode $2^{k'}$ messages.

In pass 1, Alice and Bob choose k_1 and divide their string into blocks of k_1 bits. The bits whose position is in $K_v^1 = \{l \mid (v-1)k_1 < l \leq vk_1\}$ form block v in pass 1. Alice sends the parities of all her blocks to Bob. Using `BINARY` Bob corrects an error in each block whose parity differs from that of Alice's corresponding block. At this point all of Bob's blocks have an even number of errors (possibly zero). This part of the protocol is taken from [BBBSS]. However, in that paper the leaked information about the secret string is eliminated during execution by removing one bit of each subset for which the parity is known. In our protocol all the bits are kept. Saving this information from pass to pass allows us to correct more errors.

At each pass $i > 1$, Alice and Bob choose k_i and a random function $f_i : [1..n] \to [1..\lceil \frac{n}{k_i} \rceil]$. The bits whose position is in $K_j^i = \{l \mid f_i(l) = j\}$ form block j in pass i. Alice sends Bob

$$a_j = \bigoplus_{l \in K_j^i} A_l$$

for each $1 \leq j \leq \lceil \frac{n}{k_i} \rceil$. Bob computes his b_j's in the same way and compares them with the a_j's. For each $b_j \neq a_j$ Alice and Bob execute `BINARY` on the block defined by K_j^i. Bob will find $l \in K_j^i$ such that $B_l \neq A_l$ and correct it. All the blocks K_v^u for $1 \leq u < i$ such that $l \in K_v^u$ will then have an odd number of errors. Let $\mathcal{K}$ be the set of these blocks. Alice and Bob can now choose the smallest blocks in $\mathcal{K}$ and use `BINARY` to find another error. Let l' be the position of this error in strings A and B. After correcting $B_{l'}$, Bob can determine set $\mathcal{B}$ formed by the blocks containing $B_{l'}$ from each pass from 1 to pass i. He can also determine the set $\mathcal{K}'$ of blocks with an odd number of errors by computing

$$\mathcal{K}' = \mathcal{B} \nabla \mathcal{K}.^5$$

If $\mathcal{K}' \neq \emptyset$ then Bob finds another pair of errors in the same way. This process is repeated until there are no more blocks with an odd number of errors, at which point pass i ends, and each block in passes 1 through i has an even number of errors (perhaps zero).

7.2 Using Cascade

In this section a simple analysis using only one of **Cascade**'s properties shows its usefulness in practice. This analysis yields a particular choice of block size such that the probability that a block K_v^1 has one or more errors decreases exponentially with respect to the number of passes.

The property we use is that in the passes following pass 1, correcting an error in K_v^1 implies that a second one from the same block K_v^1 will be corrected.

For parameters $k_1, \ldots, k_\omega$ chosen in a manner that depends on p, we will try to determine $\delta_i(j)$, the probability that after the pass $i \geq 1$, $2j$ errors remain in K_v^1. $\delta_1(j)$ is easily determined for $X \approx Bin(k_1, p)$:

$$\delta_1(j) = prob(X = 2j) + prob(X = 2j + 1)$$

[5] $\mathcal{B} \nabla \mathcal{K} = (\mathcal{B} \cup \mathcal{K}) \setminus (\mathcal{B} \cap \mathcal{K})$

Let E_i be the expected number of errors in K_v^1 after completion of pass i. For pass 1 we have:

$$E_1 = 2\sum_{j=1}^{\lfloor \frac{k_1}{2} \rfloor} j\delta_1(j) = k_1 p - \frac{(1-(1-2p)^{k_1})}{2}.$$

If the functions f_i with $i > 1$ are randomly chosen from $\{f \mid f : [1, \ldots, n] \to [1, \ldots, \frac{n}{k_i}]$ then for $n \to \infty$, we can determine a bound on the probability γ_i of correcting at least 2 errors at pass $i > 1$ in a block K_v^1 still containing errors after completion of pass $i-1$. Since errors are corrected two by two in passes $i > 1$, we have

$$\begin{aligned}\gamma_i &\geq 1 - \left(1 - (1 - \tfrac{k_i}{n})^{\frac{nE_{i-1}}{k_1}}\right)^2 \\ &\approx 1 - \left(1 - e^{-\frac{k_i E_{i-1}}{k_1}}\right)^2.\end{aligned}$$

We can bound $\delta_i(j)$ using γ_i for $i > 1$

$$\delta_i(j) \leq \left(\sum_{l=j+1}^{\lfloor \frac{k_1}{2} \rfloor} \delta_{i-1}(l)\right) + \delta_{i-1}(j)(1-\gamma_i).$$

Suppose that k_1 is chosen such that

$$\sum_{l=j+1}^{\lfloor \frac{k_1}{2} \rfloor} \delta_1(l) \leq \tfrac{1}{4}\delta_1(j) \tag{2}$$

and let $k_i = 2k_{i-1}$ for $i > 1$. We have

$$\delta_i(j) \leq \tfrac{1}{4}\delta_{i-1}(j) + (1 - e^{-2^{i-1}E_{i-1}})^2 \delta_{i-1}(j).$$

If in addition the choice of k_1 is such that

$$E_1 \leq -\frac{\ln \frac{1}{2}}{2} \tag{3}$$

it follows that

$$\gamma_i \geq 1 - (1 - e^{-2E_1})^2 \geq \tfrac{3}{4}.$$

When $k_i = 2k_{i-1}, i > 1$ and k_1 satisfies 2 and 3, we have $\delta_i(j) \leq \frac{\delta_{i-1}(j)}{2} \leq \frac{\delta_1(j)}{2^{i-1}}$ since $E_i \leq \frac{E_{i-1}}{2}$.

We can bound the amount of information $I(\omega)$ per block of length k_1 (per block K_v^1) leaked after ω passes, with parameters k_i set as above (ω must not depend on n for the argument to apply), as follows:

$$I(\omega) \leq 2 + \frac{1-(1-2p)^{k_1}}{2}\lceil \log k_1 \rceil + 2\sum_{l=2}^{\omega}\sum_{j=1}^{\lfloor \frac{k_1}{2} \rfloor}\frac{j\delta_1(j)}{2^{l-1}}\lceil \log k_1 \rceil .$$

To completely eliminate the errors, we choose ω large enough that we can use **Shell** on blocks formed by concatenating a large number of blocks K_v^1. Thus by corollary 11 the total amount of leaked information will approach that of **Cascade**.

Table 1 gives the values of k_1(the largest one satisfying 2 and 3) for $p \in \{0.15, 0.10, 0.05, 0.01\}$ and the values of $I(4)$ are computed. In addition the average amount of leaked information $\widehat{I}(4)$ for 10 empirical tests (with $n = 10,000$) under the same conditions is reported. For each of these tests all errors were corrected after pass 4.

Table 1. **Cascade** benchmark

p	k_1	$\widehat{I}(4)$	$kh(p)$	$I(4)$
0.01	73	6.47	5.89	6.81
0.05	14	4.60	4.01	4.64
0.10	7	3.81	3.28	3.99
0.15	5	3.80	3.05	4.12

8 Conclusions

The reconciliation problem is a variant of the noisy coding problem. The extension of the noisy coding theorem [Sh] due to Elias [E] shows that there exist optimal linear codes. Thus, it is not surprising that there exist optimal reconciliation protocols. One must use the systematic version of an optimal linear code to obtain an optimal reconciliation protocol. While these results from information theory are non-constructive, all our results are constructive.

Theorem 8 gives an (unlikely) hypothesis for which non-interactive ideal reconciliation schemes exist. If we consider other classes of hash functions, it is possible to obtain ideal protocols based on weaker hypotheses. High performance non-interactive reconciliation protocols would be useful for efficient implementation of quantum oblivious transfer [BBCS].

From a practical point of view, **Cascade** is an efficient protocol that leaks less information than the best error-correcting-codes-based reconciliation protocols. It is an improvement on the protoocol used in [BBBSS] in a true quantum setting. It would be of interest to have a detailed analysis of **Cascade**'s performance that would tell how to choose the parameters so as to minimize the amount of leaked information while maintaining a low failure probability.

9 Acknowledgements

The authors wish to thank Pierre McKenzie, Hiroki Shizuya and Claude Crépeau for helpful discussions. Special thanks to François Bessette and Daniel Simon for their suggestions and for their great help in redaction.

References

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

[BBR] C. H. Bennett, G. Brassard, J.–M. Robert, *Privacy Amplification by Public Discussion*, SIAM Journal on Computing, Vol. 17, No. 2, 1988, pp. 210–229.

[BBCS] C. H. Bennett, G. Brassard, C. Crépeau, M.–H. Skubiszewska, *Practical Quantum Oblivious Transfer*, In proceedings of Crypto '91, Lecture Notes in Computer Science, vol 576, Springer Verlag, Berlin, 1992, pp. 351–366.

[BMT] E. R. Berlekamp, R. J. McEliece, H. C. A. van Tilborg, *On the Inherent Intractability of Certain Coding Problems*, IEEE Transaction on Information Theory, Vol. IT-24, No. 3, 1978, pp. 384–386.

[CW] J. L. Carter, M. N. Wegman, *Universal Classes of Hash Functions*, Journal of Computer and System Sciences, Vol. 18, 1979, pp. 143–154.

[E] P. Elias, *Coding for Noisy Channels*, IRE Convention Record, 1957, pp. 46–47.

[M] U. M. Maurer, *Perfect Cryptographic Security from Partially Independent Channels*, In proceedings of 23*rd* Symposium on Theory of Computing, 1991, pp. 561–571.

[Sh] C. E. Shannon, *A Mathematical Theory of Communication (Part I)*, Bell System Technical Journal, Vol. 27, 1948, pp. 379–423.

[Sa] L. Salvail, *Le Problème de Réconciliation en Cryptographie*, Master thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1991.

[R] J.–M. Robert, *Detection et Correction d'Erreurs en Cryptographie*, Master thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1985.

[WC] M. N. Wegman, J. L. Carter, *New Hash Functions and Their Use in Authentication and Set Equality*, Journal of Computer and System Sciences, Vol. 22, 1981, pp. 265–279.

[W] D. Welsh, *Codes and Cryptography*, Oxford Science Publications, 1989.

[Z] S. Zachos, *Probabilistic Quantifiers Games*, Journal of Computer and System Sciences, Vol. 36, 1988, pp. 433–451.