A Probabilistic Analysis of CASCADE

Ruth Ii-Yung Ng
University of Chicago
Chicago, IL 60637
Email: ruthfrancisng@uchicago.edu

***Abstract*—This is a non-technical summary of a paper [1] written for the University of Chicago REU 2013, available at http://math.uchicago.edu/∼may/REU2013/. This paper fully explains the protocols BINARY and CASCADE and shows the comparable efficiency of the latter with another variant that might seem like an optimization to it. In [1] we also present detailed workings of [2] from QCRYPT 2013 and [3], and propose extensions to both of these, with regards to Secret-Key Reconcilation protocols for Quantum-Key Distribution. The key contribution in this is that we were able to clarify the accuracy of the bound in [3], which we present below.**

I. Introduction

Brassard and Savail's paper, [3], is a classic reference paper regarding Secret-Key Reconciliation Protocols for Quantum-Key Distribution. Specifically, the BINARY and CASCADE algorithms involve correcting errors in a shared key over a public channel. So many papers on the topic have done experimental simulations exploring the accuracy of Brassard's approximations for various implementation of CASCADE, that we will refrain from even trying to list them. However, we decided to approach the problem from a more theoretical and mathematical standpoint. Thus, the goal of our research was to begin a theoretical analysis of BINARY and CASCADE, opening this area up to more mathematical research.

We rigorously defined relevant variables, compared the protocol that is currently the "Industry Standard" with a variant that might occur as a seemingly natural optimization, then fully described the steps and approximations made in [3] and [2] with regards to CASCADE and BINARY. Finally, we present a list of extensions and suggestions, evaluating and improvements on the calculations in [3]. We summarize the novel material below.

II. Relevant Definitions and Protocols

Definitions II.1. A **key** is a finite sequence of numbers, or *bits*, where each bit is either 1 or 0. The *key-length* is the length of such a sequence, usually denoted by n.

A key, K, of length n is considered *secure* if, to any party that is not Alice or Bob, K is equally likely to be any key of length n.

We say two keys $K = \{a_1 \dots a_n\}$ and $K' = \{a'_1 \dots a'_n\}$ are equivalent if $a_i = a'_i$ for all $i = 1, 2 \dots n$.

A bit of a key $K = \{a_1, a_2 \dots a_n\}$ is considered **"leaked"** if Eve (a passive attacker) has gained information about K such that she knows that there are only 2^{n-1} possible keys that could be K. However, there exists a bit $a_i \in K$ such that $K' = \{a_1, a_2 \dots a_{i-1}, a_{i+1} \dots a_n\}$ is secure.

In Secret-Key Reconciliation, Alice and Bob wish to obtain the same Secure-Key. Alice shares a key with Bob via an imperfect Key-Distribution Algorithm, such as Quantum Key Distribution. The only way for Bob to correct his key is through a public channel, which is visible to Eve. Thus, we desire to retain the security of a shared key while correcting it.

We say that m bits are leaked if m bits of information individually have been leaked. A bit might take the form of the value of a single bit, or the value of a 1-bit checksum of a series of bits. Notice that if m bits of a key $K = \{a_1, a_2 \dots a_n\}$ are leaked, there is necessarily a key of length $n-m$ that is secure. However, this is not a sufficient condition. In the protocols studied below, it is necessary that we leak bits. Therefore, these protocols also include steps removing required bits to retain key security. Such protocols are made up of pre-established "passes". A *pass* an algorithm that is iteratively performed on a key to correct errors.

A. Parameters in Question

The focus of this paper is to calculate the following:

$P_i(j) =$ Probability that on the i^{th} pass, there are j errors

$L_i =$ Expected number of leaked bits on the i^{th} pass

For a given i, we wish to maximize $P_i(0)$, but not at the cost of too high a value of $\mathbb{E}(\sum L_i)$.

III. Description of CASCADE

For a step-by-step breakdown of the protocols, we would encourage the reader to see our paper in the link posted at [1]. However, the intution of CASCADE is the correct the key by comparing the checksums of blocks of Alice and Bob's keys.

We present a summary of the CASCADE protocol (Industry Standard) that we developed from [3]. We added some details to optimize the algorithm which may have been omitted in [3] for brevity's sake. We begin with the relevant notations.

Notations III.1.

$n =$ Key-length

$i =$ Number of passes completed

$k_i =$ Block size on the i^{th} pass

$p =$ Probability that any one bit is an error

$\sigma_i =$ Arbitrary key permutation function known to all

$\mathcal{B} = \{a_1, \dots, a_n\}$ Bob's entire key

$\mathcal{A} = \{a'_1, \dots, a'_n\}$ Alice's entire key

$B_i^x = \{a_{(x,1)}, a_{(x,2)} \ldots, a_{(x,k_i)}\}$ a subset of Bob's key where B_i^x is the x^{th} block in the i^{th} pass and each $a_{(x,y)}$ is the bit such that $a_{\sigma_i^{-1}((x-1)k_i+y)} = a_{(x,y)}$. In other words, $a_{(x,y)}$ is the y^{th} bit in the x^{th} block of the i^{th} pass, after applying the permutation σ_i.

$A_i^x = \{a'_{(x,1)}, a'_{(x,2)} \ldots, a'_{(x,k_i)}\}$ as in B_i^x, but for $\mathcal{A}$.

1) Alice sacrifices a random subset of bits in her key by sharing them with Bob to estimate p. A random subset of bits would avoid the random and systematic errors of the key-distribution protocol.
2) With the remaining bits, they apply a permutation, σ_1 to their keys to evenly distribute the errors across the key. The key from this point onward is taken to be the original key under the permutation σ_1. The rest of the protocol proceeds in passes, renaming each $\sigma_i(a_i), \sigma_i(a'_i)$ as a_i, a'_i respectively. Starting with $i = 1$, i can be incremented so long as $k_i < n/2$. These keys are denoted $\mathcal{A} = \{a'_1, a'_2 \ldots, a'_n\}$, and $\mathcal{B} = \{a_1, a_2 \ldots, a_n\}$, as described in Notations III.1.
3) Pass i:
   (i) For $i = 1$, we split each key into segments of size k_1 and compare the parities of each of these blocks with the other key's. For keys of different parity, we use a parity-based binary search for the a single error (this is called `CONFIRM` in [3]). For $i \neq 1$, we first create an empty set S which will contain all blocks that are known to contain an error which has not been corrected. For each error $e \in B_x^i$ that we find and correct, remove this block from S. We can use the permutation σ_i^{-1} to retrieve $e = a_j$, the position of e in Bob's original key. Then, notice that $a_j = a_{(x,j')} \in B_x^{i'}$ for some x, j' in each of $i' = 1, 2, \ldots, (i-1)$. Then, for each of the $i-1$ blocks $B_x^{i'}$ identified above, if $B_x^{i'} \notin S$, add $B_x^{i'}$ to S and if $B_x^{i'} \in S$, take it out. Remark: Note that this set S can be seen as the set of all blocks which currently have an odd number of errors. And each time errors are corrected, parities change, and S is changed accordingly. This is called "back-tracking" the error through other passes.
   (ii) Now, for $i \neq 1$, we recursively choose the block of lowest block size in S, we use `CONFIRM` on the block to arrive at an additional error. This error $e \in B_x^{i'}$ is removed from S and also back-tracked through the other passes, specifically, adding or removing an element from S for each of $j = 1, 2, \ldots n$, $j \neq i'$. This is continued until $S = \emptyset$.

IV. Comparison to Our Variant of CASCADE

In [1], we propose a variant that seems like a natural optimization to the protocol in Brassard's paper. We show it instead results in a lower $P_i(0)$ than the "Industry Standard".

The only modification is in step (3) of the protocol, where, instead of immediately correcting an error from each B_x^i of odd-parity we add them to the set S immediately and begin the back-tracking as above until $S = \emptyset$ to complete the pass.

It is not obvious which one of these variants leaks less bits in the process of correcting the whole key. In fact, we constructed examples in [1] where either protocol is more efficient. However, looking at the experimental results for $\sum L_i$ and $P_i(0)$ we see that the two protocols are comparable in efficiency [1], suggesting that implementing the Industry Standard `CASCADE` is preferable since it requires less management of a dynamic list.

One would intuitively think that the variant proposed by us would leak less bits while removing a comparable number of errors because each time an error is corrected in the i^{th} pass, we would pick the smallest block known to contain an error, which will leak less bits in a binary search for an incorrect bit, whereas in the Industry Standard, we begin by correcting all the errors known to be in the i^{th} pass, which have the largest k_i of all the blocks evaluated. However, experimental results prove otherwise and we postulate the following reason.

Notice that when a particular block has an odd number of errors, arranged in a specific way, one particular error will be determined by binary search. However, when an error is back-tracked to a pass, it could be any error in the block. Therefore, perhaps by using a pass as in `BINARY` first, more errors would be generated at the i^{th} pass in order to remove errors at each other pass that would not otherwise be corrected if we back-tracked at every opportunity available.

V. Extensions to Brassard's Paper

In expanding Brassard's derivation of the lower bound on $P_i(0)$, we encountered several areas where the bound could be tightened. We split these into more major contributions as well as some minor contributions. Below are relevant Notations for this sections:

Notations V.1.

$\delta_i(j) =$ $\mathbb{P}$(at the i^{th} pass, $2j$ errors remain in Bob's key)
$E_i =$ $\mathbb{E}$(Errors after the pass i in some updated B_x^1)
$\gamma_i =$ $\mathbb{P}$(2 errors in B_x^1 corrected in pass $i \mid B_x^1$ was erronous)

A. Major Contributions

1) Firstly, we propose generalizing one of Brassard's major assumptions to fix an error in Brassard's proof. In Brassard's proof, one of the underlying assumptions is assumed only for $i = 1$ (Equation 4.10 in [1]. This means that what should be

$$\sum_{l=j+1}^{\frac{k_i}{2}} \delta_i(l) \leq \frac{1}{4}\delta_i(j), \tag{V.2}$$

was stated by Brassard for only $i = 1$. Brassard's version of this assumption renders the substitution in the calculations to be incorrect for $i \neq 1$. Also, because the equation does not involve a strict equality sign, we cannot create an induction to derive the assumption we require from the one Brassard proposes.
We experimentally explored Brassard's approximation for $i = 1$. The reader is encouraged to look at the full tables

in [1]. However, we were able to generate tables like Table I, where a "1" indicates the assumption is true, "0" where the assumption is false and "X" where the protocol cannot run under those specifications.

j / k_1	0	1	2	3	4	5	6	7	8
8	1	1	1	1	1	X	X	X	X
16	1	1	1	1	1	1	1	1	1
32	0	1	1	1	1	1	1	1	1
64	0	0	0	1	1	1	1	1	1
128	0	0	0	0	0	0	1	1	1
256	0	0	0	0	0	0	0	0	0

TABLE I
TESTING (V.2): $i = 1, p = 0.05$

Through this analysis we generalized that the assumption held for large values of j, small values of p and small values of k_i. However, there are still specifications under which the assumption does not hold. We did not find a way to efficiently simultate this for $i \neq 1$ so future work could expand upon this.

2) Secondly, Brassard states $E_i \leq \frac{E_{i-1}}{2}$ as an assumption of his result (Equation 4.11)in [1]. In [3], the equation is cited as a fact, and used without justification. However, notice that $E_i \leq \frac{E_{i-1}}{2}$ is not universally true for all i, p and n. For example, consider a block where every bit is an error, meaning that $p = 1.0$. Then, no errors would get corrected in any pass and $E_i = E_{i-1}$ for all passes. In this case, the equation $E_i \leq \frac{E_{i-1}}{2}$ does not hold. We cannot determine under what circumstances Brassard quoted this equation, but it is clearly not true in the general case.

B. Minor Contributions

There are a number of other areas for minor changes to tighten the bound.

1) The property

$$e^{-ab} = \lim_{b\to\infty} (1-a)^b.$$

is used in Brassard's proof as an approximation (Just below equation 4.6 in [1]). For smaller values of n, this approximation is inaccurate as was shown in [4].

2) Later in his calculations Brassard makes an approximation (4.6 in [1]). He approximates the probability of all $\frac{nE_{i-1}}{k_1} - 1$ errors being outside B_x^i to be,

$$\left(1 - \left(1 - \frac{k_i}{n}\right)^{\frac{nE_{i-1}}{k_1}}\right)^2,$$

instead of,

$$\left(1 - \left(1 - \frac{k_i}{n}\right)^{\frac{nE_{i-1}}{k_1}-1}\right)^2.$$

Which would be the more accurate formula since we are considering the case where there are $\frac{nE_{i-1}}{k_1} - 1$ errors. We are unsure how much of an effect this has on the value of γ_i, however, in [3], Brassard does not even present this as an approximation.

3) Finally, we were unable to theoretically analyze Brassard's other assumption in his main theorem (Theorem 4.8 in [1]). However, we did write code that experimentally explored the validity of this assumption

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

.

In similar fashion to I, we generated the following table based on experimental runs of CASCADE.

p / k_1	0.01	0.03	0.05	0.07	0.09	0.11
8	1	1	1	1	1	0
16	1	1	0	0	0	0
32	1	0	0	0	0	0
64	1	0	0	0	0	0
128	0	0	0	0	0	0

TABLE II
TESTING (V.3)

In general, we can say that the assumption is true for small values of p and small values of k_i.

VI. Conclusion

In conclusion, our paper's [1] contribution is that it clearly defines the probabilistic challenge that is proposed by Brassard regarding BINARY and CASCADE. It presents significant extensions to Brassard's work which can potentially improve the bound we have on $P_i(0)$ and $\mathbb{E}(\sum L_i)$. The full paper also gives a complete rundown of Brassard's proof and the proof in [2], the most recent theoretical bounds on BINARY and CASCADE at present. This will allow readers interested in extensions (of which there are many listed in the full paper) to rigorously understand the starting point which we are all at with regards to this problem.

Acknowledgments

We would like to thank Prof, J. Peter May, Benjamin Fehrman and Jessica Lin for organizing the REU, as well as Rediet Abebe, this project's mentor. We also wish to thank Antonio (Tuca) Auffinger and Yan Zhang for helping to analyze the protocol. I would also like to thank my scholarship agency, DSO National Laboratories, for their support.

References

[1] Ng, Ruth *A Probabilistic Analysis of BINARY and CASCADE University of Chicago REU* http://math.uchicago.edu/∼may/REU2013/

[2] Seet, Sean; Ng, Ruth; Khoo, Khoongming *An Accurate Analysis of the BINARY Information Reconciliation Protocol by Generating Functions. QCRYPT 2013*

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

[4] Yamazaki, Koichi; Nair, Ranjith; Yuen, Horace P. *Problems of the CASCADE Protocol and Renyi Entropy Reduction in Classical and Quantum Key Generation* http://arxiv.org/pdf/quant-ph/0703012.pdf