

# An Information-Theoretic View of Secret Comparison

Ben Hosp

Department of Computer Science  
George Washington University  
Washington, DC 20052, USA  
Email: bhosp@gwu.edu

Poorvi L. Vora

Department of Computer Science  
George Washington University  
Washington, DC 20052, USA  
Email: poorvi@gwu.edu

**Abstract**—An information-theoretic analysis of the standard hash-based secret comparison methods suggests a small change that improves the information-theoretic secrecy of the algorithm while maintaining its computational secrecy.

## I. INTRODUCTION

This paper examines secret comparison (as described in [2], for example) from an information-theoretic point of view. Such an examination of cryptographic protocols is not typically considered useful, because the conditions of information-theoretic secrecy are far more rigid than those for computational secrecy, and are usually only satisfied by inefficient protocols. However, in this case, the analysis does not require information-theoretic secrecy, as the secret comparison protocol cannot be information-theoretically perfect. Instead, the paper examines the protocol with the view of reducing information leakage.

Existing secret-comparison protocols compare secure hash values of the secret. If the secrets do not match, the secure hash assumption implies that no computational information about the secret has been leaked; however, the Shannon information revealed is considerable, and is ignored in protocol design. It is not clear that efficient hashes [4], [3] will always be secure, in fact, past performance indicates otherwise. It appears wise, hence, to reduce the information-theoretic leakage while using a secret comparison protocol that relies on a secure hash assumption.

### A. Problem Definition

Suppose Alice has a secret  $A$  and Bob has a secret  $B$ . Suppose further that the secrets are both elements of the same domain set  $\Delta$  of size  $|\Delta|$ , so it is possible that  $A = B$ . As in [2], Alice and Bob would like to compare  $A$  and  $B$  for equality, but would not like to leak any information to each other in the case that  $A$  and  $B$  are not equal. In other words, Alice would like to do the following things:

- Determine if the value of the boolean expression  $A = B$  is true.
- In the case that  $A = B$  is false, prevent Bob from learning much information about  $A$ ; i.e. minimize information leakage to Bob.
- Prevent Bob from finding out if  $A = B$  but hiding this information from Alice; i.e. prevent Bob from cheating.

Bob has similar goals.

### B. Results

This paper suggests a simple modification to the current most general hash-based protocol to reduce information-leakage while maintaining computational secrecy. The proposed protocol is as follows: Bob and Alice share single bits of hashes of the secret, rather than the entire hash. The protocol ends when either is convinced that  $A = B$  or  $A \neq B$ . It requires a larger number of synchronized interactions, but exchanges no more bits. In fact, the reduction in information leakage comes about from exchanging fewer bits in the case of a false match. The paper shows that bitwise comparison reveals, on average, only a single information-theoretic bit of the secret when  $A \neq B$ , while hash-based comparison techniques reveal as many bits as the length of the hash. At the same time, bitwise comparison reveals no more computational bits of the secret than does hash-based comparison; in fact it often reveals fewer bits.

## II. PRIOR WORK

[2] describes several methods to perform the secret comparison. The simplest one requires Alice and Bob to reveal their secrets to a trusted third party, who will then tell them if they are equal. This third party can either be a person, a computer, or some special-purpose electronic device. In order to avoid revealing their secret even to the third party, they could jointly assign a label (or hash value) to each possible secret in the domain and each (separately) reveal the label of their secret to the third party for comparison (while keeping the list of labels or hash function hidden). Two protocols involving permutations and rotations of  $\Delta$  are described, with a goal of hiding as much information even about the answer to the question "A=B?" from the third party.

Next, [2] describes two attempts to involve third parties who do not know they are involved in a secret comparison at all. These involve converting the secret to a phone number, then dialing it and attempting to leave or pick up a message, or converting the secret to a name, and trying to make or cancel an airline reservation under that name. This approach is extended to the electronic world with the idea of Alice creating an account for Bob to attempt to access, with the password set

to her secret. Finally, the use of “containers” to communicate the secret information only to someone who knows which container to open (because in a sense he already knows that information), first with physical objects like cups, playing cards, or envelopes, and then extending the idea to the digital world with digital cryptographic envelopes. These approaches all either involve the trust of some human or electronic third party, or the security of some cryptographic primitive (or, by analogy, the physical security of some group of containers). Many also involve significant trust between Alice and Bob; that is, each must trust that the other will follow the protocol’s rules.

### III. PRELIMINARIES: NOTATION, MODEL AND OBSERVATIONS

#### A. Notation

Let  $P(\alpha)$  be the probability that an event  $\alpha$  occurs, for a given probability distribution  $P$ . Given an event  $\alpha$ , the information learned by observing  $\alpha$  is denoted  $I(\alpha)$ , and is defined as  $I(\alpha) = \log \frac{1}{P(\alpha)}$ . For a random variable  $X$ , its entropy is denoted  $\mathcal{H}(X)$ ; for a real number  $r$  such that  $0 \leq r \leq 1$ , the entropy of a binary random variable with probability distribution  $r$  and  $1 - r$  is denoted as  $\mathcal{H}_b(r)$ . Given a random variable  $X$ , its probability distribution is denoted  $P_X$ . Given two random variables  $X$  and  $Y$ , the mutual information is denoted  $I(X; Y)$ . Note that we can also talk about the mutual information between two events  $\alpha$  and  $\beta$ ; by abuse of notation, it is denoted  $I(\alpha; \beta)$ . As mentioned earlier,  $A$  is Alice’s secret and  $B$  Bob’s. All secrets belong to the domain  $\Delta$ , which is of size  $|\Delta|$ .  $h$  represents a secure hash function. Lower case letters with subscripts represent bits of the hashed random variable, so, for example,  $a_j$  represents the  $j^{\text{th}}$  bits of  $h(A)$ .  $n$  is the number of protocol instances used by Alice and Bob.

#### B. Modeling Assumptions

We view the problem in the situation where no third party, trusted by both Alice and Bob, exists. We will use discrete random variables to model the value of  $A$  (from Bob’s perspective), and specifically the equality or inequality of  $A$  and  $B$ . Let  $EQ$  be a binary random variable:

$$EQ = \begin{cases} eq & A = B \\ -eq & A \neq B \end{cases}$$

Alice (for example) wants to reduce her uncertainty about the value of  $EQ$  as much as possible while reducing Bob’s uncertainty about  $A$  as little as possible. We must assume some probability distribution  $P_A$  for  $A$  from Bob’s point of view, or, equivalently,  $P_B$  for  $B$  from Alice’s point of view (which will imply a probability distribution  $P_{EQ}$  for  $EQ$ , since  $EQ$  depends solely on  $A$  and  $B$ ).

Suppose  $P_A$  is uniform. Then

$$P_A(x) = \frac{1}{\Delta} \forall x \Rightarrow P_{EQ} = \begin{cases} \frac{1}{\Delta} & EQ = eq \\ \frac{|\Delta|-1}{|\Delta|} & else \end{cases}$$

$$\Rightarrow \mathcal{H}(EQ) \simeq 0 \text{ for large } \Delta$$

In other words,  $\mathcal{H}(EQ)$  would already be very small, and Alice (or Bob) would already be almost certain that  $A \neq B$ . Her uncertainty about  $EQ$  would be essentially as low as it can go without actually learning the value of  $B$ . This would make engaging in a secret-comparison protocol inappropriate, because none of the goals of such a protocol could be furthered. Therefore, we examine this problem when  $P_{EQ}(eq)$  is non-trivial.

For simplicity, we assume  $P_{EQ}(eq) = \frac{1}{2}$  (other values of  $P_{EQ}(eq)$  are addressed in the Appendix). This is the case for maximum uncertainty for Alice and Bob, that is,  $\mathcal{H}(EQ) = \mathcal{H}_b(\frac{1}{2}) = 1$  bit. We further assume that the unequal secrets are uniformly distributed; in other words, if  $B = \delta_\beta$ ,

$$P_A(\delta_\alpha) = \begin{cases} \frac{1}{2} & \delta_\alpha = \delta_\beta \\ \frac{1}{2} \times \frac{1}{|\Delta|-1} & else \end{cases} \quad (1)$$

We also assume that  $|\Delta|$  is very large, specifically that the number of bits in the representation of each element of  $\Delta$  is very large compared to the number of protocol messages that are exchanged. More formally, we assume that  $n \ll \log_2 |\Delta|$ .

#### C. Some Observations

In the case that  $A \neq B$ , there is some minimum information about  $A$  that Alice must reveal (and conversely, information about  $B$  that Bob must reveal), and this lower bound is greater than zero. When Alice engages in a secret-comparison protocol with Bob, she will reveal an answer: yes or no. In the case of a “no” answer, Bob at least learns that  $A \neq B$ . Before, from his perspective, there were  $|\Delta|$  possible values for  $A$ ; now, there are  $|\Delta| - 1$  possibilities. This is a tight lower bound on the amount of information given up by each of the participants; agents asking each other yes/no questions (and receiving honest answers) cannot give up any less information. If Bob is semi-honest, i.e. performs the protocol honestly, but attempts to discover as much information about Alice’s secret as possible, then engaging in a secret-comparing protocol  $O(n)$  times with him only eliminates  $O(n)$  possible values of Alice’s secret, and the effort required by him to discover the value of  $A$  is comparable to Naive Search.

The above is in contrast to the process of using an inequality to compare two secrets, i.e. asking the question “Is  $A > B$ ?” When Alice and Bob get the answer to that question, they learn which of  $A$  and  $B$  is larger. On average, this will eliminate  $\frac{|\Delta|}{2}$  possibilities for the value of  $A$  (from Bob’s perspective). This means that, in engaging in a secret-inequality protocol with Bob,  $O(n)$  protocol instances reduce the number of possible values of Alice’s secret by a factor of  $\frac{1}{2^n}$ . In this case, a semi-honest Bob’s effort in discovering Alice’s secret is comparable to Binary Search.

## IV. HASH-BASED SECRET-COMPARISON PROTOCOL

### A. Protocol Description

Suppose Alice and Bob decide to compare their secrets as follows: they have a secure hash function  $h$ , so Alice computes

and reveals  $h(A)$  and Bob computes and reveals  $h(B)$ . If  $h(A) = h(B)$ , then they decide  $A = B$  (accept), otherwise they decide  $A \neq B$  (reject). We will suppose for most of the analysis that  $h$  is not a compressor (meaning its range is the same size as  $\Delta$ ). By the definition of a secure hash function, however,  $h$  is not invertible in the computational model under consideration (though it is information-theoretically invertible, that is, there is enough information available to invert it, given unbounded computational resources). This is a description of the general secret comparison protocol.

### B. Analysis

1) *Computational Secrecy*: This protocol is computationally secret if  $h$  is computationally secure. Bob does not receive any information from Alice other than  $h(A)$ , and must invert  $h$  to learn  $A$ . Because  $h$  is computationally secure, the only way to determine  $A$  is to exhaustively search  $\Delta$ . However, once Bob knows  $h(A)$ , his computations are “offline” – that is; without further involving Alice.

2) *Information-theoretic*: This protocol is, clearly, not information-theoretically secure. It is not designed to be so. However, in the case of unequal secrets, it is not clear that it reveals as little information as possible, because Bob can, in the information-theoretic model, uniquely determine  $A$  whether  $A = B$  or not. This is of concern because past performance indicates that popularly-used efficient hash functions will typically be broken in the years to come.

## V. BITWISE SECRET-COMPARISON PROTOCOL

### A. Protocol Description

We can modify the above protocol so that it has the same properties of cryptographic security but does much better in terms of information-theoretic security. The result is a protocol that allows each of Alice and Bob to reveal one bit of information about their secret at a time, stopping when either becomes certain that the other has or does not have the same secret. (Unfortunately, this has the implication of allowing a malicious participant to engage in a Binary Search of  $\Delta$  rather than a Naive Search – it does not reach the theoretical lower bound of information exchanged in such a comparison.)

The protocol is quite simple and is interactive. It has a risk of false match,  $Pr[FalseMatch] = \frac{1}{2^n}$ , where  $n$  is the number of protocol messages exchanged, and depends on the degree of confidence Alice and Bob agree require in the match  $A = B$ . Obviously, if they need to be completely confident of the match,  $n = \log_2 |\Delta|$ . It proceeds as described below.

Alice randomly generates an integer  $i$  and requests  $b_i$ , the  $i^{th}$  bit of  $h(B)$ . Bob replies with this information and a request for  $a_j$ , where  $j$  is a different random integer and  $a_j$  is the  $j^{th}$  bit of  $h(A)$ ; Alice replies with this information. Alice checks that Bob’s response to her is equal to  $a_i$ , and Bob checks that Alice’s response to him is equal to  $b_j$ . If either notices a discrepancy, that party rejects and the protocol ends. Otherwise, they continue with another iteration of the protocol until they have either rejected or completed  $n$  iterations, where  $\frac{1}{2^n}$  is smaller than the larger of their tolerances for false

positives. One of them is now convinced that  $h(A) = h(B)$  (and therefore  $A = B$ ), and ends the interaction. One may also assume that they negotiate a confidence level beforehand, and hence that both are now convinced of the equality.

### B. Malicious Participants

From the information-theoretic perspective, we do not consider inverting  $h$  to be difficult, because the semi-honest/malicious participants we consider have infinite time and computing power. Therefore, every bit of  $h(A)$  Bob knows allows him to eliminate half of the possible values for  $A$ . If he has tricked Alice into engaging in this protocol in order to verify a guess of the value of  $A$  (or, equivalently, of  $h(A)$ ), and he has made a very lucky guess, he may learn quite a few bits of  $h(A)$ , and if he has a guess that is close enough to  $h(A)$  to satisfy Alice’s tolerance of a false match, she may begin discussing  $A$  with him as if he knows it. Further, he could repeat the protocol in order to learn more bits of  $h(A)$ , with a different guess and perhaps with a different pseudonym. (Maybe Alice will not believe that Bob has two secrets but if she was interested in comparing her secret with Bob’s, she might also be interested in comparing it with Chris’s.) It is therefore necessary for parties engaging in this protocol to insist on low tolerances for false positive results.

### C. Analysis

1) *Information-theoretic Leakage*: Suppose Alice’s secret is  $A = \delta_\alpha$ . We assume (1). Then,

$$\begin{aligned} \mathcal{H}(A) &= \frac{1}{2} \log 2 + \frac{1}{2(|\Delta| - 1)} \log(2(|\Delta| - 1)) \\ &= \frac{1}{2} + \frac{1}{2(|\Delta| - 1)} + \frac{1}{2(|\Delta| - 1)} \log(|\Delta| - 1) \\ &= \frac{|\Delta| + \log(|\Delta| - 1)}{2(|\Delta| - 1)} \end{aligned}$$

and

$$\mathcal{H}(EQ) = \frac{1}{2}$$

Suppose Bob receives the first bit that he requested from Alice. Consider the random variable  $\Gamma_1$ , and let  $\gamma_1$  be the event that this bit matches the bit at that position in the hash of his secret.

$$P(\gamma_1 | \neg eq) = \frac{\frac{|\Delta|}{2} - 1}{|\Delta| - 1} \simeq \frac{1}{2}$$

if  $\Delta \gg 1$ . Thus:

$$\begin{aligned} P(\gamma_1) &= P(eq)P(\gamma_1|eq) + P(\neg eq)P(\gamma_1|\neg eq) \\ &= \frac{1}{2} \times 1 + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4} \\ P(\neg \gamma_1) &= P(eq)P(\neg \gamma_1|eq) + P(\neg eq)P(\neg \gamma_1|\neg eq) \\ &= \frac{1}{2} \times 0 + \frac{1}{2} \times \frac{1}{2} = \frac{1}{4} \\ \mathcal{H}(\Gamma_1) &= \frac{3}{4} \log \frac{4}{3} + \frac{1}{4} \log 4 = 2 - \frac{3}{4} \log 3 \end{aligned}$$

We are interested in the information communicated by the events  $\gamma_1$  and  $\neg\gamma_1$  about  $EQ$  and  $A$  (by way of  $h(A)$ ). Specifically:

$$\begin{aligned} I(\gamma_1; eq) &= \log \frac{P(\gamma_1, eq)}{P(\gamma_1)P(eq)} = \log \frac{4}{3} \\ I(\gamma_1; \neg eq) &= \log \frac{P(\gamma_1, \neg eq)}{P(\gamma_1)P(\neg eq)} = \log \frac{2}{3} \\ I(\neg\gamma_1; \neg eq) &= \log \frac{P(\neg\gamma_1, \neg eq)}{P(\neg\gamma_1)P(\neg eq)} = 1 \end{aligned}$$

Since  $I(\neg\gamma_1; \neg eq) = \mathcal{H}(EQ)$ ,  $\neg\gamma_1 \Rightarrow \neg eq$ ; only one mismatching bit is required to determine  $A \neq B$ . Also, since  $I(\gamma_1; eq) > 0 > I(\gamma_1; \neg eq)$ ,  $\gamma_1$  suggests that  $EQ = eq$ .

Let  $\Gamma_i$  be the random variable associated with the  $i^{\text{th}}$  bit Bob receives from Alice. Longer series of  $\gamma_i$  events will further reduce the uncertainty about  $EQ$ . For example:

$$\begin{aligned} I(eq; \gamma_2 | \gamma_1) &= \log \frac{P(\gamma_2, eq | \gamma_1)}{P(\gamma_2 | \gamma_1)P(eq | \gamma_1)} = \log \frac{6}{5} \\ I(eq; \gamma_3 | \gamma_2, \gamma_1) &= \log \frac{P(\gamma_3, eq | \gamma_2, \gamma_1)}{P(\gamma_3 | \gamma_2, \gamma_1)P(eq | \gamma_2, \gamma_1)} = \log \frac{10}{9} \end{aligned}$$

And one can easily see the following:

**Lemma 1:**

$$\begin{aligned} &I(eq; \gamma_k | \gamma_{k-1}, \dots, \gamma_1) \\ &= \log \frac{P(\gamma_k, eq | \gamma_{k-1}, \dots, \gamma_1)}{P(\gamma_k | \gamma_{k-1}, \dots, \gamma_1)P(eq | \gamma_{k-1}, \dots, \gamma_1)} = \log \frac{2^k + 2}{2^k + 1} \end{aligned}$$

*Proof:* Straightforward.

We can easily verify that

$$\sum_{i=1}^{\infty} I(eq; \gamma_i | \gamma_{i-1}, \dots, \gamma_1) = \mathcal{H}(EQ) = 1$$

The actual values of  $I(eq; \gamma_k | \gamma_{k-1}, \dots, \gamma_1)$  are slightly larger, since there are actually  $\log |\Delta|$  terms (not  $\infty$ ), but the summation must add up to 1 when  $i$  goes from 1 to  $\log |\Delta|$  (because  $\mathcal{H}(EQ) = 1$ ), so each term must be slightly larger to make up the difference.

When Alice and Bob have exchanged  $m$  bits, they each will have reduced their uncertainty about  $EQ$  by  $\tau = \sum_{i=1}^m I(eq; \gamma_i | \gamma_{i-1}, \dots, \gamma_1)$ , which leads to the following:

**Lemma 2:**

$$\sum_{i=1}^m I(eq; \gamma_i | \gamma_{i-1}, \dots, \gamma_1) = \log \frac{2^{m+1}}{2^m + 1}$$

*Proof:* Lemma 1 provides an expression for  $I(eq; \gamma_i | \gamma_{i-1}, \dots, \gamma_1)$ , and  $\sum_{i=1}^m \log \frac{2^i + 2}{2^i + 1} = \log \prod_{i=1}^m \frac{2^i + 2}{2^i + 1}$  provides the result.

The Appendix contains a similar analysis for the cases when  $P_{EQ}(eq) \neq \frac{1}{2}$ .

2) *Computational Secrecy:* This protocol reveals no more information than is revealed when Alice and Bob simply compare their hashes. If we considered  $h$  sufficient (from a computational perspective) to secure the secrets when all of  $h(A)$  and  $h(B)$  were known, then we should certainly consider the secret computationally secure when not all of  $h(A)$  or  $h(B)$  is known.

#### D. False Positive Results

Notice that the original hash-comparison method involved no risk of a false positive result, because all bits of the hashes were compared. The information-theoretic benefits from using the bitwise comparison protocol instead of direct hash comparison hence come at the cost of incurring a risk of false positive. We could also get an information-theoretic benefit by decreasing  $|h(\Delta)|$ —that is, by making  $h$  a compressor. This would also come at the cost of a risk of false positive results. In fact, any information-theoretic benefit necessarily involves an increased risk of false positive results, because both are direct consequences of an increase in post-protocol  $\mathcal{H}(B)$ ; one in the case of a match, the other in the absence of a match.

1) *Equivalence between Bitwise-comparison and Direct Hash-comparison:* Suppose Alice and Bob have different secrets, and exchange  $n$  bits of the (non-compressor) hash values of their secrets before discovering that  $A \neq B$ . They have reduced their uncertainty about  $EQ$  to 0, and they have each reduced their uncertainty about the other's secret by approximately  $n$  bits. If they had exchanged  $n$  bits without discovering  $A \neq B$  (because one of the bits did not match), then the false positive rate at that point would be  $\frac{1}{2^n}$ . The same reduction in uncertainty and the same false positive rate would result if Alice and Bob simply compared an  $n$ -bit hash of their two secrets. In this sense the Bitwise-comparison protocol and the Direct Hash-comparison protocol are equivalent.

2) *Benefits of Bitwise-comparison:* Despite this equivalence, there is a distinct benefit from the use of the Bitwise-comparison protocol. Suppose Alice and Bob agree that  $\frac{1}{2^n}$  is an acceptable risk of a false positive result. They can achieve this by reducing Alice's uncertainty about  $B$  (and Bob's uncertainty about  $A$ ) by  $n$  bits. That can be accomplished by exchanging  $n$ -bit hashes of  $A$  and  $B$ . That can also be accomplished by exchanging  $n$  random bits of a non-compressing hash of those secrets. It is better to exchange the random bits because they can stop the exchange as soon as they find a mismatch. They get the same fixed risk of a false positive result, but the information divulged about their secrets is only bounded above by  $n$  bits, not fixed at  $n$  bits.

To see more clearly the benefits of this protocol, consider the case in which Alice and Bob each will tolerate no risk of a false positive result, so they insist that  $|h(\Delta)| = |\Delta|$ . When they each simply reveal the hash of their secret "up front", Alice's (for example) uncertainty about  $B$  is reduced to 0; that is, she learns  $\mathcal{H}(B) = \mathcal{H}(h(B)) = O(\log |\Delta|)$  bits about  $B$ .

On the other hand, suppose they instead reveal the bits of their hash one at a time as described above, but with zero

tolerance for false positives; that is, they will continue to exchange bits until they find a mismatch (true negative) or run out of bits to exchange because they have found that all their bits match (true positive). Consider the number of exchanges they can each expect to reveal about their secrets when their secrets are not equal. Approximately half the time, there will be a mismatch on the first bit exchanged. If there is not, there will be a mismatch approximately half the time on the second bit exchanged, and so on. This gives the following lemma:

**Lemma 3:** *The average amount of information revealed about the secret when  $A \neq B$  is one bit.*

*Proof:* The average amount of information revealed is:

$$\mathcal{H}(A; \gamma_1, \gamma_2, \dots | \neg eq) = \lim_{k \rightarrow \infty} \sum_{i=1}^k \frac{1}{2^i} = 1$$

The upper bound on the information Alice learns is indeed still  $\mathcal{H}(B)$ , but the expected amount of information revealed is smaller than in the hash-based protocol, which leads to the main result:

**Theorem 1:** *The average information revealed by Bitwise Comparison is  $\frac{1}{n}^{th}$  that of Hash-Based Comparison when  $A \neq B$ , and both protocols may be used to provide a zero probability of a false match.*

*Proof:* Straightforward, from Lemmas 2 and 3. For  $P_{EQ}(eq) \neq \frac{1}{2}$ , see Appendix.

## VI. CONCLUSIONS

We have demonstrated a simple modification to current secure-hash-based secret comparison protocols. Our method reveals one bit of Shannon information in the case when the two secrets are equal, while the hash-based protocol reveals as many bits of Shannon information as the length of the hash. At the same time, our protocol reveals no more computational information than the hash based one. The cost of the information-theoretic improvement is an increase in communication complexity.

## VII. APPENDIX A: DIFFERENT PROBABILITY DISTRIBUTIONS FOR $EQ$

### A. Information per Protocol Bit

The analysis in the main text involved the assumption that  $P(eq) = \frac{1}{2}$ . When  $P(eq)$  is some other non-trivial  $\rho$ , the following applies:

$$I(eq; \gamma_k | \gamma_{k-1}, \dots, \gamma_1)$$

$$\begin{aligned} &= \log \frac{P(\gamma_k, eq | \gamma_{k-1}, \dots, \gamma_1)}{P(\gamma_k | \gamma_{k-1}, \dots, \gamma_1) P(eq | \gamma_{k-1}, \dots, \gamma_1)} \\ &= \log \frac{1}{P(\gamma_k | \gamma_{k-1}, \dots, \gamma_1)} = \log \frac{1}{\frac{1+p^k}{1+p^{k-1}}} \\ &= \log \frac{1+p^{k-1}}{1+p^k} \end{aligned}$$

$$I(eq; \gamma_k, \gamma_{k-1}, \dots, \gamma_1) = \sum_{i=1}^k \log \frac{1+p^{i-1}}{1+p^i}$$

$$\begin{aligned} \lim_{k \rightarrow \infty} I(eq; \gamma_k, \gamma_{k-1}, \dots, \gamma_1) &= \sum_{i=1}^{\infty} \log \frac{1+p^{i-1}}{1+p^i} \\ &= \log \prod_{i=1}^{\infty} \frac{1+p^{i-1}}{1+p^i} \\ &= \log \lim_{k \rightarrow \infty} \frac{2}{1+p^k} \\ &= \log 2 = 1 \end{aligned}$$

### B. False Positive Rate

In the case where  $P(eq)$  is some non-trivial  $\rho$ , we should notice that the average information revealed when  $A \neq B$  is unaffected.

## REFERENCES

- [1] J. Cohen Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *CRYPTO '86*, pages 251–260, Berlin, 1986. Springer-Verlag. Lecture Notes in Computer Science Volume 263.
- [2] R. Fagin, M. Naor, P. Winkler. Comparing information without leaking it. *CACM*, Vol 39, No. 5, May 1996, pp. 77–85.
- [3] Secure Hash Standard. *FIPS Pub. 180-1*. 1993
- [4] Ron Rivest. RFC 1321 - The MD5 Message Digest Algorithm. *Internet RFC*, 1992