Title

On the Shannon Perfect Secrecy Result.

ISBN

978-1-7281-9972-6

Author

Sadjadpour, Hamid R

Publication Date

2020

On the Shannon Perfect Secrecy Result

Hamid R. Sadjadpour
Department of Electrical and Computer Engineering
University of California Santa Cruz
Santa Cruz, CA, USA

***Abstract*—Shannon information theoretic approach to perfect security requires equal number of key bits to the information bits. This fact has been studied for decades and known as "Shannon impossibility result". This paper demonstrates that it is feasible to start with a significantly smaller number of key size than the information bits and generate *final* keys such that perfect secrecy can be achieved. Our claim is that we can obtain the required entropy to achieve perfect secrecy from a combination of random bits and other sources of randomness in order to construct uniformly distributed and unique *final* keys.**

To achieve this goal, we redefine the original problem by Shannon and consider securing multiple groups of packets simultaneously, each group containing T packets of length n bits. We introduce two types of keys called *common* and *temporary* keys that have uniform distribution and are statistically independent of each other and the data. Each *temporary* key has n bits and there are T *common* keys of length n that are used many times. The actual *final* keys that are used to achieve perfect secrecy are constructed by a unique combination of *common* and *temporary* keys and the use of a random pairing approach. Although the size of the *final* keys is equal to the size of the information bits, but the total size of the *common* and *temporary* keys is significantly smaller than the size of information bits. We also introduce an iterative technique to reduce the ratio of key to the data. Our result implies that the ratio of key to the ratio of data can be arbitrarily decreased by increasing the computational complexity of the *final* key construction.

***Index Terms*—Encryption, Perfect Secrecy.**

I. Introduction

Claude Shannon [1] introduced a cipher system in which achieves perfect secrecy against computationally unbounded attackers. Shannon proved that if we use a different key, selected uniformly at random from the set of all keys for each message, then perfect secrecy is achievable. However, the downside of this approach is that the size of the key should be as large as the size of the data. When the size of data increases, the approach is no longer practical. This paper attempts to reduce the key size required to achieve perfect secrecy by redefining the problem statement and removing some of the restrictions that was originally considered in [1].

In this paper, we present a practical solution that achieves perfect secrecy with much smaller key size than the file size. The main idea is to generate the required entropy for achieving perfect secrecy by a combination of two types of keys and randomizing the key bit locations for a number of packets instead of a single packet.

We provide a security solution for a group of files[1] instead of individual files. We achieve our goal by introducing different types of keys. By doing so, we are able to take advantage of multiple parameters (degrees of freedom) and generate unique and uniform *final* keys using fewer number of key bits. Note that our solution generates the same number of *final* key bits as Shannon, however, these keys are generated by combining significantly smaller number of random keys called *common* and *temporary* keys. Our claim is that Shannon perfect secrecy can be achieved using significantly smaller number of random bits than the information bits. The ratio of the key bits proposed in this paper to the original solution by Shannon can asymptotically tend to zero.

The rest of this paper is organized as follows. In section II, the related works are presented. Section III is dedicated to describing the encryption technique and section IV proves that the proposed approach achieves perfect secrecy. Section V demonstrates some of the properties of the proposed approach. The paper is concluded in section VI.

II. Related Works and Current contribution

A. Related Works

The only work in this area was based on the bounded-storage model concept in [2]–[5] and many other works that followed. The ground-breaking work was first introduced by Maurer in [2]. Maurer introduced the creation of a random bit string that is publicly available to both legitimate users and eavesdroppers. However, the storage-bounded eavesdroppers can only access a fraction of the random bit string. Cachin and Maurer [4] introduced a more general bounded-storage assumption by taking advantage of a complicated privacy amplification technique and Renyi entropy considerations. Ding and Rabin [5] presented two protocols that are provably secure against any adversary in this bounded-storage model. All these works assume that the eavesdropper has unbounded computational complexity. However, if the eavesdropper has enough storage capability, these approaches fail. Further, the cipher introduced in [2] is proven to be perfectly secure *with high probability*. Our approach does not require any restriction on the eavesdropper storage size or computational capability and is perfectly secure with probability one for all the stored encrypted data. In our approach, we assume both the user and the eavesdropper have unlimited storage and computational capabilities.

Network coding schemes have been shown to be very efficient from a security point of view. Cai and Young [6] showed that network coding can be used to achieve perfect secrecy when network encoded files are sent through multiple

[1]In this paper, we use the terms file and packet interchangeably.

paths and only one of the paths are compromised. Bhattad et al. [7] studied the problem of "weakly secure" network coding schemes in which even without perfect secrecy, no meaningful information can be extracted from the network during transfer of data. Subsequent to [7], Kadhe et al. studied the problem of weakly secure storage systems in [8], [9]. Yan et al. also proposed [10], [11] algorithms to achieve weak security and also studied weakly secure data exchange with generalized Reed Solomon codes. There are other efforts to achieve perfect secrecy asymptotically. For example, the authors in [12] introduced the concept of codes for security without any error correction capability by using sparse vectors to achieve asymptotic perfect secrecy. The proposed method [12] significantly outperforms *Advanced Encryption Standard (AES)* in terms of computational complexity. That coding scheme [12] has the unique ability of providing asymptotic perfect secrecy with low decoding complexity. The code for security in [12] still suffered from the problem of achieving perfect secrecy asymptotically which does not make the technique practical.

B. Our Contribution

We define security for a new class of applications. Assume an organization wants to store their sensitive data into the public cloud storage systems such that even the cloud storage provider is unable to retrieve the data now or in the future with the advances in computations. Under this assumption, we no longer have the traditional sender and receiver concept and the key exchange requirement. Instead, the sender stores the encrypted data in the cloud storage systems without providing the keys to the service provider. In this new paradigm, any organization using this technique will secure the secret keys instead of the entire information (data) using significantly smaller size storage. Further, the sender requires the encrypted data to be information-theoretically secure by using perfect secrecy technique also known as one-time pad.

Our approach provides a perfect secrecy solution for all the encrypted data that are stored in the cloud. There is no restriction on the ability of the eavesdropper to use any operation that requires infinitely large computations or memory.

III. Perfect Secrecy Encryption Approach

Let each packet P (or file) has n bits. The i^{th} packet P_i is represented by a vector as $P_i = [p_i^1\ p_i^2\ \dots\ p_i^n]$ which belongs to $\mathbb{F}_{2^n}$. There are T of these files that are encrypted together, i.e., $\mathcal{P} = \{P_1, P_2, \dots, P_T\}$. To achieve perfect secrecy, we use the following steps for encryption of the T files.

1) A set of T files of equal size $\mathcal{P} = \{P_1, P_2, \dots, P_T\}$ is selected to encrypt together. Note that these T files can be segments of a larger file or from multiple files.
2) A random binary matrix $\mathcal{CK}$ of size $T \times n$ is created. Each row of this matrix is unique and has uniform distribution and statistically is independent from other rows and the data. The matrix $\mathcal{CK}$ contains the *common* keys and is defined as

$$\mathcal{CK} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ a_{n+1} & a_{n+2} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{(T-1)\times n+1} & a_{(T-1)\times n+2} & \dots & a_{T\times n} \end{bmatrix}.$$

3) An n bit *temporary* key $\mathcal{TK} = [tk^1 tk^2 \dots tk^n]$ is selected uniformly at random from the set of all n bit strings. This key is statistically independent of the files and the *common* keys. The *temporary* key $\mathcal{TK}$ is only used once for each set of T files.
4) The *final* keys are generated using a combination of *common* and *temporary* keys. The *final* keys for T files (packets) are denoted as

$$\mathcal{K} = \begin{bmatrix} k_1 & k_2 & \dots & k_n \\ k_{n+1} & k_{n+2} & \dots & k_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{(T-1)\times n+1} & k_{(T-1)\times n+2} & \dots & k_{T\times n} \end{bmatrix},$$

$$= \begin{bmatrix} \mathcal{K}_1 \\ \mathcal{K}_2 \\ \vdots \\ \mathcal{K}_T \end{bmatrix}.$$

We randomly pair every two numbers between 1 and nT together. Therefore, there are $\frac{nT}{2}$ pairs that are non-overlapping and the union of them provides all numbers between 1 and nT. The random pairing is uniformly selected from all random pairing possibilities. We assume n is even. The random pairing $\mathcal{RP}$ is applied to *common* key which can be shown as

$$\mathcal{CK}(\mathcal{RP}) = \begin{bmatrix} a_{i_1} & a_{i_2} & \dots & a_{i_n} \\ a_{i_{n+1}} & a_{i_{n+2}} & \dots & a_{i_{2n}} \\ \vdots & \vdots & \vdots & \vdots \\ a_{i_{(T-1)\times n+1}} & a_{i_{(T-1)\times n+2}} & \dots & a_{i_{(T\times n)}} \end{bmatrix}.$$

For example, a_{i_1} and a_{i_2} are the first random pair in the *common* keys.
5) Assume one consecutive pair of $\mathcal{CK}(\mathcal{RP})$ in the same row has elements of j_1 and j_2 where $1 \le j_1, j_2 \le nT$, $j_1 = i_{l\times n+r}$, $j_2 = i_{l\times n+r+1}$, and $0 \le l \le (T-1)$. This implies that j_1 and j_2 are the r and $r+1$ elements of $l+1$ row in $\mathcal{CK}(\mathcal{RP})$. Let $\gamma_{j_1} = a_{j_1} \oplus tk^r$, $\gamma_{j_2} = a_{j_2} \oplus tk^{r+1}$, $\gamma_2 = a_{j_1} \oplus tk^{r+1}$, and $\gamma_1 = a_{j_2} \oplus tk^r$. Note that $1 \le r \le n-1$ since $\mathcal{TK}$ has only n elements. The final key bits are constructed as

$$k_{j_1}, k_{j_2} = \begin{cases} \gamma_{j_1}, \gamma_{j_2} & \text{if } \gamma_2 = 0, \gamma_1 = 0 \\ \gamma_{j_2}, \gamma_{j_1} & \text{if } \gamma_2 = 0, \gamma_1 = 1 \\ \gamma_{j_1}, \neg\gamma_{j_2} & \text{if } \gamma_2 = 1, \gamma_1 = 0 \\ \neg\gamma_{j_2}, \gamma_{j_1} & \text{if } \gamma_2 = 1, \gamma_1 = 1 \end{cases} \quad (1)$$

We define $\neg\gamma_{j_1} = 1 - \gamma_{j_1}$ and k_{j_1} and k_{j_2} are elements of matrix $\mathcal{K}$.

6) The i^{th} encrypted file $\mathcal{F}_i$ is generated as $\mathcal{F}_i = \mathcal{K}_i \oplus P_i$.

Our key construction in (1) is such that both the values of *final* key bits and their locations are random with respect to the *common* keys. Equation (1) is designed in such a way that the total number of ones or zeros for k_{j_1} and k_{j_2} for all 16 possible cases is equal to 8. The *final* keys $\mathcal{K}_i$ have uniform distribution as will be proved later.

The main idea for having total key bits less than information bits while achieving perfect secrecy lies behind the fact that we use the *common* keys for many groups of files, each containing T files. Therefore, as we encrypt more files, the average ratio of the key bits to the information bits decreases significantly. This operation is possible because we use random pairing concept that creates enough entropy to use *common* keys many times. The second reason is because we use a novel iterative approach to construct the *common* keys. The details will be discussed later.

IV. Proof of Perfect Secrecy

This section is dedicated to the proof of perfect secrecy for the proposed protocol. We first prove that the T keys generated for one set of files are uniformly distributed and unique. Then we prove that two keys that are generated using the same row of matrix $\mathcal{CK}$ for two different groups of files are also uniformly distributed and unique. Finally, we compute the actual number of times that we can use the same *common* keys $\mathcal{CK}$ for *final* key construction while achieving perfect secrecy.

Theorem 1. *Any two keys generated using the same* temporary *key and different rows of the same* common *key matrix are uniformly distributed, statistically independent, and unique.*

Proof. Let's define two keys as $\mathcal{K}_{i_1} = [k_{(i_1-1)\times n+1}, k_{(i_1-1)\times n+2}, \ldots, k_{i_1\times n}]$ and $\mathcal{K}_{i_2} = [k_{(i_2-1)\times n+1}, k_{(i_2-1)\times n+2}, \ldots, k_{i_2\times n}]$ that belong to the same group of files. The j^{th} bit of these two keys can be constructed using equation (1). Note that the j^{th} bits of $\mathcal{K}_{i_1}$ and $\mathcal{K}_{i_2}$ are using different elements of the matrix $\mathcal{CK}$, and we assume in both cases the first line in equation (1) is used. Other cases can be proved using similar argument.

$$\begin{aligned} k_{j_1} &= a_{j_1} \oplus tk^j \\ k_{j_2} &= a_{j_2} \oplus tk^j \end{aligned} \tag{2}$$

We use Crypto lemma [13] to prove that the j^{th} bit ($1 \le j \le n$) of any two keys $\mathcal{K}_{i_1}$ and $\mathcal{K}_{i_2}$ within one set of T files have uniform distribution and are statistically independent.

Lemma 1. *(Crypto lemma) Let $(C, +)$ be a compact abelian group with group operation $+$, and let $X = M + K$, where M and K are random variables over C and K is independent of M and uniform over C. Then X is independent of M and uniform over C.*

In our particular example, we can easily see that $k_{j_1} \oplus a_{j_1} = k_{j_2} \oplus a_{j_2}$. Therefore, we can write $k_{j_1} \oplus a_{j_1} \oplus a_{j_2} = k_{j_2}$. Let's assume K in Crypto lemma is equal to $a_{j_1} \oplus a_{j_2}$, then it is clear that $a_{j_1} \oplus a_{j_2}$ has uniform distribution. Hence using Crypto lemma and (2), it can be concluded that k_{j_1} and k_{j_2} are independent with uniform distribution. Since this is true for all the bits of these two keys, it can be concluded that any two keys inside one set of T files are statistically independent with uniform distribution and unique. □

In this protocol, we use the *common* keys in matrix $\mathcal{CK}$ for multiple groups of T files. The next theorem proves that these keys that are constructed for multiple sets of files are also statistically independent with uniform distribution. Let's consider the L_1 and L_2 groups of files, each containing T files. These two groups use different *temporary* keys of $\mathcal{TK}_{L_1}$ and $\mathcal{TK}_{L_2}$ to construct $\mathcal{K}_i(L_1)$ and $\mathcal{K}_i(L_2)$ respectively, as described in section III. In order to prove the independence of the keys, we will compare keys that are generated for these two groups of files using the same random pair bits from matrix $\mathcal{CK}$. Further, we also assume that in equation (1), for the ${j_1}^{th}$ element of keys $\mathcal{K}_i(L_1)$ and $\mathcal{K}_i(L_2)$, they are using the same condition in equation (1), i.e., $\gamma_2 = 0, \gamma_1 = 0$. Similar argument can be made for other possible cases.

Theorem 2. *The keys used for each group of T files are uniformly distributed and statistically independent of the keys generated for another group of files.*

Proof. Let's look at the j^{th} bit of the key of the L_1 and L_2 groups using the same *common* key $\mathcal{CK}$.

$$\begin{aligned} k_{j_1}(L_1) &= a_{j_1} \oplus tk^j(L_1) \\ k_{j_1}(L_2) &= a_{j_1} \oplus tk^j(L_2) \end{aligned} \tag{3}$$

For these two key bits, the *common* key a_{j_1} is the same in construction of the keys but they use two different *temporary* key bits, $tk^j(L_1)$ and $tk^j(L_2)$ that are statistically independent. Using similar argument as Theorem 1 and by observing that $k_{j_1}(L_1) \oplus tk^j(L_1) = k_{j_1}(L_2) \oplus tk^j(L_2)$ and by transferring the temporary keys to one side of equality, $k_{j_1}(L_1) \oplus tk^j(L_1) \oplus tk^j(L_2) = k_{j_1}(L_2)$, one can use the Crypto lemma and (3) to conclude that $k_{j_1}(L_1)$ and $k_{j_1}(L_2)$ are statistically independent with uniform distribution. □

It is important to compute the number of times a single *common* key matrix $\mathcal{CK}$ can be used to generate different final keys $\mathcal{K}$.

Theorem 3. *Let each group contains T files and each file consists of n bits. Further, the* common *keys are used for L groups. The maximum value of L to guarantee perfect secrecy is $\lfloor \frac{T/2 + T \times \log_2(\frac{n \times T}{e})}{T-1} \rfloor$.*

Proof. Let's assume the *common* keys are used L times. Clearly, L is a finite number and at some point, we need to generate new *common* keys in order to assure that the final keys have the entropy requirement to achieve perfect serecy. The total number of information bits is equal to $T \times n \times L$. Since we require uniform distribution of final keys, the entropy for this number of final keys is equal to $T \times n \times L$. In our approach, we have three sources of randomness denoted as common keys $\mathcal{CK}$, temporary keys $\mathcal{TK}$, and random pairing $\mathcal{RP}$. Our goal is to compute the entropy of the final keys

by using these sources. More precisely, we want to compute the joint entropy $H(\mathcal{CK}, \mathcal{TK}, \mathcal{RP})$. These three sources of randomness are independent of each other and therefore, their joint entropy is equal to the sum of individual entropies.

$$H(\mathcal{CK}, \mathcal{TK}, \mathcal{RP}) = H(\mathcal{CK}) + H(\mathcal{TK}) + H(\mathcal{RP}) \quad (4)$$

The first source is the *common* keys that are uniformly distributed and since there are $n \times T$ bits, its entropy is equal to $n \times T$. The second source of randomness comes from *temporary* keys. Each time that we use the *common* keys, we need n bits of *temporary* keys and after using the *common* keys L times, we use a total of $n \times L$ bits, creating entropy of similar value because its distribution is uniform. The final source of randomness comes from random pairing. For the first pair, there are $\binom{n\times T}{2}$ possible choices selected uniformly. The second pair have $\binom{(n\times T)-2}{2}$ possible outcomes. Combining all these cases provides $\frac{(n\times T)!}{2^{\frac{n\times T}{2}}}$ possible outcomes of equal probability. In order to guarantee perfect security, we must have

$$n \times T + n \times L + \log_2(\frac{(n \times T)!}{2^{\frac{n\times T}{2}}}) = T \times n \times L. \quad (5)$$

By utilizing Stirling's approximation for logarithm, i.e., $\log_2(n!) \approx n\log_2(n) - n\log_2(e)$, and solving for L we arrive at

$$L = \lfloor \frac{T(1/2 + \log_2(\frac{n\times T}{e}))}{T-1} \rfloor. \quad (6)$$

In order to guarantee perfect secrecy, we need to use the floor function since the left side of equation (5) should be always greater than the right side of equation (5). □

V. Discussion

A. Complexity-Storage trade-off

A quick look at equation (6) reveals that L grows logarithmically with respect to the values of n and T. For example, for $n = 1000$ and $T = 100$, the value of L is 15. In order to reduce the ratio of the number keys bits to the number of information bits even further, we resort to an iterative approach to increase this ratio. Suppose that value of T is initialized as T_0. Assuming n is fixed, we can generate a total $L_0 \times T_0$ uniform and independent keys of length n where

$$L_0 = \lfloor \frac{T_0(1/2 + \log_2(\frac{n\times T_0}{e})}{T_0 - 1} \rfloor. \quad (7)$$

By defining $T_1 = L_0 \times T_0$, we can create a new *common* key of size $T_1 \times n$. This is achieved by combining L_0 common keys of size $T_0 \times n$ to construct a new common key of size $T_1 \times n$. Then, we compute L_1 similar to (7) by simply replacing T_0 in that equation with T_1. This step will result in $T_2 = L_1 \times T_1$ uniform and independent keys of length n. Continuing this procedure for m steps, it is easy to see that $T_m = (\Pi_{i=0}^{m-1} L_i) \times T_0$. Note that in each step, we need to generate L_i uniform temporary keys of length n. For example, if we initialize the systems with values $n = 1000$, $T = 100$ and after four iterations, the ratio of key bits to information bits is 3.6×10^{-7} while in the first iteration, this ratio was only 0.067.

Let's define the ratio of keys to information bits as R. In the initialization step, we are using $T_0 \times n$ *common* key bits and $L_0 \times n$ *temporary* key bits. After the first iteration, we only use $L_1 \times n$ *temporary* keys to expand the *common* key size that we created in the previous stage. This will continue m iterations. Therefore, we use a total of $(T_0 + \sum_{i=0}^{m} L_i) \times n$ key bits. We will explain later that we use an innovative technique to perform random pairing such that there is no need to require additional key bits. The total number of information bits that can be encrypted is equal to $L_m \times T_m \times n$.Therefore, the ratio of key bits to information bits is given by

$$R = \frac{T_0 + \sum_{i=0}^{m} L_i}{(\Pi_{i=0}^{m} L_i) \times T_0}. \quad (8)$$

By careful review of this iterative approach, it appears that the ratio of key bits to information bits R approaches zero by increasing the number of iterations, or equivalently by increasing the computational complexity of encoding and decoding. The result demonstrates that the original approach by Shannon was pessimistic because it relied only on randomizing the key bits. However, with faster computers that can be deployed, longer key bits can be replaced with computational complexity through random pairing approach that will be described next.

B. Random Pairing

Majority of the entropy creation in the new approach comes from random pairing concept. The random pairing approach requires significant secret key contribution that can eventually wipe out all the gains achieved in the previous section. We use an innovative approach to implement the random pairing without requiring any additional key bits.

Let's define the set $S_1 = \{1, 2, \ldots, L_0\}$ and $\mathcal{CK}_i$ as the i^{th} *common* key for $1 \le i \le L_0$. By careful review and rewriting of equation (5), it is clear that the contribution of the random pairing approach at the initialization ($T = T_0$) is approximately $T_0 \times n \times (L-1)$ bits of information[2].

$$n \times L + \log_2(\frac{(n \times T_0)!}{2^{\frac{n\times T_0}{2}}}) = T_0 \times n \times (L-1) \quad (9)$$

For $\mathcal{CK}_i$ and $i \in S_1$, we use the concatenation of all $\mathcal{CK}$'s in the set $S_1 - \{i\}$ for generating the required $T_0 \times n \times (L-1)$ bits to perform random pairing of the elements in $\mathcal{CK}_i$. Note that these $\mathcal{CK}$'s are statistically independent and hence, the condition for equation (5) and (9) holds. Using this technique, we will now have the following set of *common* keys at the end of first iteration, i.e., $\mathcal{CK}_1^{S_1-\{1\}}, \mathcal{CK}_2^{S_1-\{2\}}, \ldots, \mathcal{CK}_{L_0}^{S_1-\{L_0\}}$, where the subscript refers to the *common* keys that is used for construction of the new set of *common* keys and the superscript represents the sequence that is used for random pairing. The size of each $\mathcal{CK}_i$ is T_0 by n while the size of $\mathcal{CK}_i^{S_1-\{i\}}$ is $(L_0 \times T_0)$ by n. $\mathcal{CK}_i^{S_1-\{i\}}$ is constructed by combining the L_0 different and independent *temporary* keys $\mathcal{TK}$s that are created using $\mathcal{CK}_i$ while the random pairing is achieved with the help of all $\mathcal{CK}_j$ with $j \in S_1 - \{i\}$. Also, all

[2]Note that the first term in the left side of equation (9) does not contribute significantly to the entropy and can be safely ignored.

the new *common* keys are statistically dependent and cannot be combined for the next iteration. In order to continue this process to increase the size of the *common* keys in the second iteration, we need to continue this step as follows. We use more sets of *common* keys to generate the next group of *common* keys as $\mathcal{CK}_{L_0+1}^{S_2-\{L_0+1\}}, \mathcal{CK}_{L_0+2}^{S_2-\{L_0+2\}}, \ldots, \mathcal{CK}_{2L_0}^{S_2-\{2L_0\}}$. Construction of this set of *common* keys is similar to the previous one except that the *common* keys that are used for this group belong to the set $S_2 = \{L_0+1, L_0+2, \ldots, 2L_0\}$. For the second iteration, we need to construct a total of L_1 sets of *common* keys, each one containing L_0 elements as shown below. Each row of this set is constructed similar to the technique explained above and the *common* keys in each row are statistically dependent.

$$\begin{array}{rr} \mathcal{CK}_1^{S_1-\{1\}}, \ldots, & \mathcal{CK}_{L_0}^{S_1-\{L_0\}}, \\ \mathcal{CK}_{L_0+1}^{S_2-\{L_0+1\}}, \ldots, & \mathcal{CK}_{2L_0}^{S_2-\{2L_0\}}, \\ \vdots, \ldots, & \vdots \\ \mathcal{CK}_{(L_1-1)L_0+1}^{S_{L_1}-\{(L_1-1)L_0+1\}}, \ldots, & \mathcal{CK}_{L_1L_0}^{S_{L_1}-\{L_1L_0\}}. \end{array} \quad (10)$$

Now, if we define the set of indexes $S_1^1 = \{1, L_0+1, \ldots, (L_1-1)L_0+1\}$[3], we can create the next set of *common* keys of size $(L_1 \times L_0 \times T_0)$ by n from this set of *common* keys, $\mathcal{CK}_1^{S_1-\{1\}}, \mathcal{CK}_{L_0+1}^{S_2-\{L_0+1\}}, \ldots, \mathcal{CK}_{(L_1-1)L_0+1}^{S_{L_1}-\{(L_1-1)L_0+1\}}$. Note that all these *common* keys are statistically independent and therefore, the technique that we described earlier can be used. The first *common* key is designed as $\mathcal{CK}_1^{S_1 \cup S_1^1 - \{1\}}$. As we can see again, this key is constructed from $\mathcal{CK}_1^{S_1-\{1\}}$ along with L_1 *temporary* keys $\mathcal{TK}$. The random pairing selection is chosen by using $(L_1-1) \times T_1 \times n$ bits from the set $S_1^1 - \{1\}$. This set of common keys can be denoted as $\mathcal{CK}_1^{S_1 \cup S_1^1 - \{1\}}, \mathcal{CK}_{L_0+1}^{S_2 \cup S_1^1 - \{L_0+1\}}, \ldots,$ $\mathcal{CK}_{(L_1-1)L_0+1}^{S_{L_1} \cup S_1^1 - \{(L_1-1)L_0+1\}}$. In this new notation, the superscript $S_1 \cup S_1^1 - \{1\}$ for the *common* key $\mathcal{CK}_1^{S_1 \cup S_1^1 - \{1\}}$ means that for the first iteration, we used the set $S_1 - \{1\}$ to perform random pairing and for the second iteration, we used the set $S_1^1 - \{1\}$ for random pairing.

We can repeat the same procedure for all the columns of equation (10) to create new sets of *common* keys. For example for the second column, define the set $S_2^1 = \{2, L_0+2, \ldots, (L_1-1)L_0+2\}$ and by using the *common* keys $\mathcal{CK}_2^{S_1-\{2\}}, \mathcal{CK}_{L_0+2}^{S_2-\{L_0+2\}}, \ldots, \mathcal{CK}_{(L_1-1)L_0+2}^{S_{L_1}-\{(L_1-1)L_0+2\}}$ that are statistically independent, we generate the next set of common keys as $\mathcal{CK}_2^{S_1 \cup S_2^1 - \{2\}}, \mathcal{CK}_{L_0+2}^{S_2 \cup S_2^1 - \{L_0+2\}}, \ldots,$ $\mathcal{CK}_{(L_1-1)L_0+2}^{S_{L_1} \cup S_2^1 - \{(L_1-1)L_0+2\}}$.

In this approach, at the beginning we start with L_0 *common* keys of size T_0 by n. In the first iteration, we generate more *common* keys and we will have a total of L_0 *common* keys of size $(L_0 \times T_0)$ by n. In the m^{th} iteration, we have generated $\Pi_{i=0}^{m} L_i$ *common* keys, each one having a size of $(\Pi_{i=0}^{m-1} L_i \times T_0)$ by n bits. Next figure describes the common key construction procedure for the first iteration starting from L_0 *common* keys.

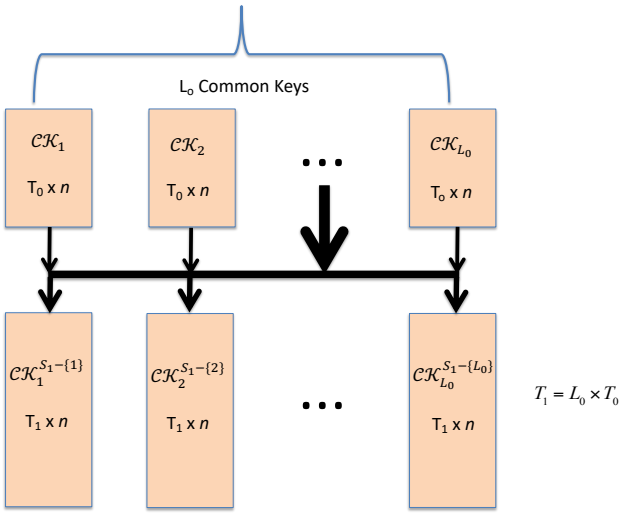

Fig. 1. First iteration of key construction. Each *common* key in the first iteration is constructed from one of the *common* keys from initial stage and L_0 *temporary* keys plus $L_0 - 1$ remaining *common* keys that are used for random pairing.

Remark 1. Our random pairing approach suggests that it is feasible to achieve the required entropy in equation (4) by taking advantage of the high computational complexity of available computers. Note that this result implies that in order to achieve perfect secrecy, it is not necessary to have significant number of key bits. Other sources of randomness can be used in order to achieve perfect secrecy while reducing the secret key requirements.

Remark 2. In this approach, random pairing table is a deterministic table and it is easy to show that it is not even required to build this table. This table is not part of secret key and it is known by eavesdropper. However the selection of the random pairing that is based on a combination of the *common* keys is secret. The details of the implementation is beyond the scope of this paper.

Figure 2 demonstrates the evolution of common key after each iteration[4]. At the initial stage, all the bits of the *common* key should be generated by a random bit generator. However after the initialization, we only need the *temporary* keys in order to increase the size of the number of rows for the *common* key. For example in the first iteration, the size of the *common* key is originally $T_1 \times n = T_0 \times L_0 \times n$ but at the end of the first iteration, this size increases to $T_0 \times L_0 \times L_1 \times n$ which is L_1 times larger at the end of this iteration. Note that in each iteration, we only need new *temporary* key bits and the random pairing $\mathcal{RP}$ operation does not require any new key

[3]These are all elements of the first column of equation (10).

[4]For simplicity, all the subscript and superscript associated to $\mathcal{CK}$ is dropped in this figure.

bits. Instead, we use the required bits from other *common* keys which creates some statistical dependence among final keys as explained earlier. The number of *temporary* key bits required for each iteration is significantly smaller than the total number of bits increased in the *common* key. Since the number of bits that we can encrypt is proportional to the size of the total number of *common* key bits in the last iteration, increasing the number of iterations allows us to decrease the ratio of key bits to the information bits in this technique. In this figure, each green rectangular block contributes to the total required number of key bits while the white rectangular blocks do not contribute to the total number of key bits for construction of that particular *common* key after m iterations.

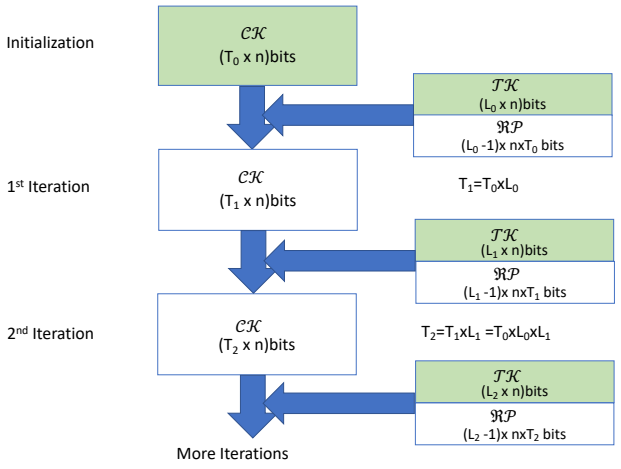

Fig. 2. Each Common Key's row increases its size by a factor of L_i at the i^{th} iteration while there are few bits required to increases its size.

C. Uniqueness of final keys

In practice, each time a temporary key is selected that provides duplicate final keys, we should exclude that temporary key for final key generation. One shortcoming of the proposed approach is the fact that in order to achieve perfect secrecy, we still need to generate the final keys of $\mathcal{K}_i$ which are the same size as information bits in order to make sure that the final keys are unique. This is computationally very complex and clearly is not desirable in many practical applications. A practical solution to avoid comparing all these final keys in advance with previous final keys is to have a very low upper bound on the probability of two final keys being similar. Next theorem demonstrates that we can select random unique temporary keys and with high probability (w.h.p), the ultimate final keys $\mathcal{K}_i$ will be unique.

Theorem 4. *Let each group contains T files and each file consists of n bits. Further, the* common *keys are used for L groups. The final keys that are created with our iterative technique are unique with high probability (w.h.p.) at each iteration.*

Proof. In order to prove this, we first prove this for the first iteration and then demonstrate that this is also true for each subsequent iteration. At the initialization step, we generate *common* keys of size T_0 by n. We assume that T_0 is large enough that we can approximate $\frac{T_0}{T_0-1}$ equal to 1 in order to simplify the presentation of the proof. Using equation (7) and the simplification above we have,

$$L_0 = \lfloor 1/2 + \log_2(\frac{n \times T_0}{e}) \rfloor. \tag{11}$$

Based on the construction of the *common* keys in $\mathcal{CK}$, it is obvious that it is impossible for the keys in one group of files to become identical since each row of $\mathcal{CK}$ is a random uniform and unique vector of length n. However, after T_0 set of final keys $\mathcal{K}$ are created for the first group of files, for the second group, it is possible one or more of T_0 final keys $\mathcal{K}$ become similar to the previous keys. The probability of having unique keys for the second group conditioning on unique keys for the first group is $\frac{2^n - T_0}{2^n}$. This probability is the same for all the keys in this group. The reason is that each key generated in the second group, may be similar to one of the keys in the first group but it is impossible to be the same in its own group based on our *common* key construction scheme. Once the second group of keys are created, for the third group, the probability of having unique keys conditioning on the construction of unique keys in previous groups is $\frac{2^n - 2T_0}{2^n}$. Finally, for the L_0^{th} group of files, the probability of having unique keys assuming we have created unique keys for the previous $L_0 - 1$ groups is equal to $\frac{2^n - (L_0-1)T_0}{2^n} = 1 + \frac{0.5 \times T_0}{2^n} - \frac{T_0 \times \log_2(\frac{n \times T_0}{e})}{2^n}$. In the first iteration, it is easy to see that the conditional probability of having unique keys if the previous step had unique keys is upper bounded by $1 + \frac{0.5 \times T_0 \times L_0}{2^n} - \frac{T_0 \times L_0 \times \log_2(\frac{n \times T_0 \times L_0}{e})}{2^n}$. After m iterations, this probability can be computed as the product of all these individual probabilities. Note that as m increases[5], the conditional probability decreases and moving away from one. The smallest of these probabilities is the last one at the m^{th} iterations, namely, $(1 + \frac{0.5 \times T_0 \times \Pi_{i=0}^{m-1} L_i}{2^n} - \frac{T_0 \times \Pi_{i=0}^{m-1} L_i \times \log_2(\frac{n \times T_0 \times \Pi_{i=0}^{m-1} L_i}{e})}{2^n})$. As n tends to infinity, it is clear that this conditional probability approaches 1. Therefore, the joint probability of all these final keys being unique approaches 1 with high probability.

□

This is a very strong result which makes this scheme practical and with high probability, provides perfect secrecy without any need to create and compare all final keys in advance to check for uniqueness. The fact that the ratio of key bits to information bits in this scheme is decreasing by increasing the number of iterations or equivalently, by using more computational power, is very important specially since we will soon have quantum computers capable of providing large number of operations. This is a significant reduction in key requirements for achieving perfect secrecy.

[5] m is a finite value and in practical systems, it is usually a number between 3 and 5.

In this approach, the secret keys consist of *common* keys at the initialization stage $\mathcal{CK}_i, 1 \leq i \leq L_0$, *temporary* Keys $\mathcal{TK}$ from all of the m iterations, T_0, and m. Table I summarizes the list of the secret keys that are used in this approach.

Secret Key List	
Common keys at initialization stage	$\mathcal{CK}_i, 1 \leq i \leq L_0$
Temporary Keys required from all of the m iterations	$\mathcal{TK}$
Initial value of T	T_0
Number of iterations	m

TABLE I
TABLE OF SECRET KEYS

For example, suppose that we want to generate random temporary keys and using them to generate $\mathcal{K}_i$ without checking them with previously generated final keys. Using the values of $n = 1000$ and $T_0 = 100$, it can be easily shown that the probability of initial step of having similar final keys is equal to approximately 1467×2^{-1000}.[6] This probability can be similarly computed for all the iterations to show very low probability. Therefore, in many practical applications we don't need to create the final keys $\mathcal{K}_i$ in advance and compare them with previously generated final keys $\mathcal{K}_i$ for uniqueness.

The above computation on the number of possibilities may imply that encoding of this approach is computationally expensive. As a matter of fact in order to encode each bit, we need to use equation (1). In order to encode every two bits, we need to compute γ_{j_1}, γ_{j_2}, γ_1, and γ_2 for each iteration and then use (1) to derive two keys and finally XOR each key with an information bit. This is equivalent of only $(2m + 1)$ XOR operations per information bit. However, the challenge is that the encoding should be carried for a large number of packets/files simultaneously since we are computing many keys simultaneously. The decoding operation also requires to build these large number of keys even if we may only use a portion of the keys for decrypting a given file. This implies that this technique is well suited for archival data that are highly sensitive but we do not require to decode them often. For data that requires frequent access by users, decoding becomes computationally complex because of the complicated construction of the keys. If computational capability is available, then this approach can be used for data that are accessed frequently.

D. Comparison with One-time Pad Approach

Shannon one-time pad approach [1] states that the ratio of the key to information bits is 1. This result is also known as Shannon impossibility result [14]. In this paper, we have shown that this ratio can arbitrary tend to zero by increasing the encoding and decoding computational complexity. One important question is that *why are we able to achieve such result contrary to Shannon approach?* By reviewing Shannon problem formulation, we observe that the key construction is based on designing a key for a single packet (file). The only connection between different files is the assumption that the uniformly distributed keys are unique and different from each other. Therefore, for this particular problem formulation, we only have a single parameter (one degree of freedom) to work with. Contrary to Shannon's approach, we formulated the problem by providing perfect secrecy for multiple groups of files together. Further, we used single *common* keys for many files. We also introduced *temporary* keys and random pairing concept to produce additional entropy. Unlike Shannon approach that creates the entropy by only randomizing the bits, in our approach the entropy is created from three sources, namely, *common* keys, *temporary* keys, and random pairing of *common* keys to create the final keys. Interestingly, most of the entropy in our approach stems from random pairing which requires significant number of bits. By using *common* keys from other groups of files, we were able to overcome this problem and not using any additional bits.

Note that by using *common* and *temporary* keys and uniquely combining them, we still need to create final keys $\mathcal{K}_i$ that have exactly the same size as information bits. Therefore, we have not violated Shannon impossibility theorem but instead proposed a smart way of creating these keys without actually requiring to store the same amount of key bits as information bits.

Another interesting observation is the fact that current advances in computational complexity and future use of quantum computers seems to create a major challenge for computationally secure encryption techniques. We changed this challenge into opportunity by taking advantage of this computational complexity and availability of large amount of data to store. Our approach used this capability to provide perfect secrecy that is even immune to the advancement of quantum computers that will be developed in the future.

Shannon perfect secrecy requires three distinct conditions [15] to achieve perfect secrecy. First, the size of the set of keys should be the same as the size of the set of messages which can be achieved by using equal key and information packet lengths. Second, each key should be unique. Finally, each key should be uniformly distributed. These three conditions are achieved in the proposed protocol for key generation.

We have simulated an image file with a size of 2.8 MB (22.4×10^6 bits). The original image includes approximately 71% zeroes and the rest are ones. We have divided this file into smaller files of size 5 KB (n =40 Kbits). After using the proposed algorithm for encryption using one iteration, the bit distribution tends to uniform distribution with the average ratio of zeros and ones being equal to 0.4999999973 and 0.5000000027, respectively.

VI. CONCLUSIONS

This paper introduces a novel approach to achieve perfect secrecy with minimum key size requirements. The proposed technique encrypts T packets of length n together. The approach uses two types of *common* and *temporary* keys along

[6]Note that the total number of atoms in the universe is estimated to be around 2^{300}. This result implies that if we choose one of the atoms in the universe with equal probability, this probability is significantly higher than having two final keys being the same.

with a novel random pairing technique to produce the necessary entropy to achieve perfect secrecy. An iterative approach is also introduced to reduce the ratio of key bits to information bits. The approach guarantees perfect secrecy for encrypted data. Future quantum computing capabilities actually allows us to use larger number of iterations which will lead to reduce the ratio of key bits to information bits.

Future work will focus on extending the results to applications such as wireless cellular and ad hoc networks.

ACKNOWLEDGEMENT

The author would like to thank Option Quant (Oquant) Inc. for sponsoring this project.

REFERENCES

[1] Claude E Shannon. Communication theory of secrecy systems. *Bell Labs Technical Journal*, 28(4):656–715, 1949.

[2] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[3] Stefan Dziembowski and Ueli Maurer. Tight security proofs for the bounded-storage model. *In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 341–350, May 2002.

[4] C. Cachin and U. Maurer. Unconditional security against memory bounded adversaries. *In Advances in Cryptography-Crypto*, pages 292–306, 1997.

[5] Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security. *In Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–26, March 2002.

[6] Ning Cai and Raymond W Yeung. Secure network coding. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, page 323. IEEE, 2002.

[7] Kapil Bhattad, Krishna R Narayanan, et al. Weakly secure network coding. *NetCod, Apr*, 104, 2005.

[8] Swanand Kadhe and Alex Sprintson. On a weakly secure regenerating code construction for minimum storage regime. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 445–452. IEEE, 2014.

[9] Swanand Kadhe and Alex Sprintson. Weakly secure regenerating codes for distributed storage. In *Network Coding (NetCod), 2014 International Symposium on*, pages 1–6. IEEE, 2014.

[10] Muxi Yan, Alex Sprintson, and Igor Zelenko. Weakly secure data exchange with generalized reed solomon codes. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1366–1370. IEEE, 2014.

[11] Muxi Yan and Alex Sprintson. Algorithms for weakly secure data exchange. In *Network Coding (NetCod), 2013 International Symposium on*, pages 1–6. IEEE, 2013.

[12] Mohsen Karimzadeh Kiskani, Hamid R Sadjadpour, Mohammad Reza Rahimi, and Fred Etemadieh. Low complexity secure code (LCSC) design for big data in cloud storage systems. In *Communications (ICC), 2018 International Conference on*. IEEE, 2018.

[13] G David Forney Jr. On the role of MMSE estimation in approaching the information-theoretic limits of linear gaussian channels: Shannon meets wiener. *arXiv preprint cs/0409053*, 2004.

[14] Yevgeniy Dodis. Shannon impossibility revisited. *In International Conference on Information Theoretic Security*, pages 100–110, Berlin, Heidelberg, 2012.

[15] Matthieu Bloch and Joao Barros. *Physical-layer security: from information theory to security engineering*. Cambridge University Press, 2011.