

Re-visiting the One-Time Pad

Nithin Nagaraj¹, Vivek Vaidya², and Prabhakar G. Vaidya¹

(Corresponding author: Nithin Nagaraj)

School of Natural and Engineering Sciences, National Institute of Advanced Studies¹
 Indian Institute of Science Campus, Bangalore-12, India (Email: nithin_nagaraj@yahoo.com)
 Imaging Technologies Lab GE Global Research, John F. Welch Technology Center²
 Bangalore-66, India

(Received Jan. 11, 2006; revised and accepted May 7, 2006)

Abstract

In 1949, Shannon proved the perfect secrecy of the Vernam cryptographic system (One-Time Pad or OTP). It has generally been believed that the perfectly random and uncompressible OTP which is transmitted needs to have a length equal to the message length for this result to be true. In this paper, we prove that the length of the transmitted OTP actually contains useful information and could be exploited to compress the transmitted-OTP while retaining perfect secrecy. The message bits can be interpreted as True/False statements about the OTP, a private object, leading to the notion of private-object cryptography.

Keywords: One time pad, perfect secrecy, private-key cryptography, private-object cryptography, shannon security

1 Introduction

Cryptography, the science and the art of communicating messages secretly has been the subject of intense research for the last 50 years. The field itself is much older, dating as far back as 1900 BC, when Egyptian scribes used a derived form of the standard hieroglyphics for secure communication.

In 1949, Shannon, the father of information theory, wrote a seminal paper (see [8]) on the theory of secrecy systems, where he established the area on a firm footing by using concepts from his information theory [7]. In his 1949 paper, among other important contributions, he established the perfect secrecy of the Vernam cryptographic system, popularly known as the One-Time Pad or OTP for short. OTP happens to be the only known *perfectly secure* or *provably, absolutely unbreakable* cipher till date. Shannon's work meant that OTPs offer the best possible mathematical security of any encryption scheme (under certain conditions), anywhere and anytime – an astonishing result.

There have been a number of cryptographic algorithms

[4] in the last century, but none can provide Shannon security (perfect security) other than the OTP. This is one of our motivations to probe into the OTP and investigate its properties. To the best of our knowledge there has been very little work on the OTP since Shannon. Recently, Raub and others [5] describe a statistically secure one time pad based crypto-system. Dodis and Spencer [1] show that the difficulty of finding perfect random sources could make achieving perfect security for the OTP an impossibility. We shall not deal with the issue of random sources in this paper. The questions we intend to address in this paper are: what can we say about the length of the OTP to be transmitted across the secure channel? We prove a counter-intuitive result in this paper – the length of the OTP to be transmitted need not always be equal to the length of the message and that it is possible to achieve Shannon security even if the transmitted OTP length is actually *smaller* than the message length. Note that we treat the OTP as perfectly random and uncompressible. However, the length of the OTP is one piece of information that is not exploited and is always compromised in its traditional usage. We construct a protocol where this piece of information can be used effectively to reduce the length of the OTP to be transmitted while not losing Shannon security for any of the bits of the message. Although the average reduction in length of the transmitted OTP we obtain is meagre (0.75 – 2 bits), this could prove quite significant if a large number of relatively 'short' OTPs are shared between Alice and Bob (furthermore, one could envisage a protocol where the ensemble of OTPs have random lengths, the random sequence being known only to Alice and Bob). The savings obtained in such a scenario is non-negligible. Our investigation of the OTP also results to an alternate interpretation of the OTP encryption and this leads to a new paradigm of cryptography called private-object cryptography.

The paper is divided as follows. In the next section, we describe the OTP and its traditional interpretation as XOR operation by means of a simple example. In Section 3, we prove the central theoretical result of the paper

– that it is possible to have the transmitted OTP length less than the message length while still retaining perfect secrecy. We first prove a 1-bit reduction of the transmitted OTP length and then generalize for a k -bit reduction for a message of length $N > k$ bits. We also provide an alternative method of compressing the OTP based on the length information. In Section 4, we provide our new alternate interpretation of the OTP as a private-object and the encryption/decryption as equivalent to making statements about the object. Section 5 talks more about the new paradigm of private-object cryptography. We claim that every private-key cryptography is essentially a form of private-object cryptography and can provide theoretical security for at least one message of length equal to the entropy of the crypto-system. We then ask the important question – how should we invest N bits of secret? We hint towards the use of Formal Axiomatic Systems (FAS) for this purpose. We conclude in Section 6.

2 One-Time Pad

In 1917, Gilbert Vernam of AT&T invented the first electrical one time pad. The Vernam cipher was obtained by combining each character in the message with a character on a paper tape key. There were other developments in the 1920s which resulted in the paper pad system. An OTP was used for encrypting a teletype hot-line between Washington and Moscow. OTPs were also used successfully by the English in World War II. These were especially useful in battlefields and remote regions where there were no sophisticated equipments for encryption, all that they used were OTPs printed on silk. The final discovery of significance and theoretical importance of the OTP was made by Claude Shannon in 1949.

2.1 The Classical Interpretation of OTP

We describe the encryption and decryption of an OTP by a simple example. Alice and Bob have shared an OTP ($K = 1011001001$) in complete secrecy (assume that they have met in private and shared the key). One fine day, Alice wants to invite Bob to her house and wishes to send the message ‘COME AT 8 PM’ to him. But she is afraid of the interception of the message by Eve whom she dislikes. She therefore encrypts her message as follows. She first converts her message into binary (assume that she has a dictionary which converts the message into the bits $M = 0010110101$). She then performs an XOR operation to yield the cipher-text $C = K \oplus M = 1001111100$. She transmits this across a public channel. Bob receives the cipher-text C . Since he has the OTP with him, he does the XOR operation of the cipher-text with the OTP to yield the correct message $M = C \oplus K = 0010110101$. He then looks up at the dictionary (this need not be secret) and converts this to the more readable message ‘COME AT 8 PM’.

To summarize (refer to Figure 1):

Alice (encryption)		Bob (decryption)	
K:	1 0 1 1 0 0 1 0 0 1	K:	1 0 1 1 0 0 1 0 0 1
M:	0 0 1 0 1 1 0 1 0 1	C:	1 0 0 1 1 1 1 1 0 0
C:	1 0 0 1 1 1 1 1 0 0	M:	0 0 1 0 1 1 0 1 0 1

Figure 1: The one-time pad encryption and decryption interpreted as XOR operation

- 1) The OTP is a random set of bits which is used as a private-key known only to Alice and Bob.
- 2) The OTP encryption involves an XOR operation of the message M with the OTP to yield the cipher-text C .
- 3) The OTP decryption involves an XOR of the cipher-text C with the OTP to get back the original message M .

The classical interpretation of the OTP as XOR implies the following two important observations.

- 1) The length of the OTP is completely compromised in the process of encryption.
- 2) One bit of the OTP is employed to encrypt exactly one bit of the message and this requires one XOR operation. All bits of the message require the same amount of effort to encrypt and decrypt.

We shall have more to say about the above observations later. But what can we say about the security of the OTP encryption?

2.2 Security of the OTP

Shannon, in his lucid 1949 paper on the theory of secrecy systems [8], defined perfect secrecy as the condition that the *a posteriori* probabilities of all possible messages are equal to the *a priori* probabilities independently of the number of messages and the number of possible cryptograms. This means that the cryptanalyst has no information whatsoever by intercepting the cipher-text because all of her probabilities as to what the cryptogram contains remain unchanged. He then argued that there must be at least as many of cryptograms as the messages since for a given key, there must exist a one-to-one correspondence between all the messages and some of the cryptograms. In other words, there is at least one key which transforms any given message into any of the cryptograms. In particular, he gave an example of a perfect system with equal number of cryptograms and messages with a suitable transformation transforming every message to every cryptogram. He then showed that the OTP actually achieves this. In other words, the best possible mathematical security is obtained by the OTP. Incidentally, this is the only known method that achieves Shannon security till date.

3 Transmitted OTPs of Length Less than the Message Length

It has generally been believed that the OTPs that are *transmitted* are required to have a length equal to that of the message in order for Shannon’s argument to hold (although Shannon himself never mentioned this in his paper). In this section, we show this is not the case. Although the length of the OTP *while* encryption need to be equal to the length of the message, the OTP that is *transmitted* could be less. But this sounds quite paradoxical because the OTP is assumed to have been derived from a *perfect random source* and hence *uncompressible*. Even if we are able to construct a compression algorithm that compresses some of the generated OTPs, it has to expand some other OTPs, it can’t losslessly compress *all* OTPs. This is because of the *Counting Argument* [6] which states that every lossless compression algorithm can compress only some messages while expanding others. However, we prove the central theoretical result of this paper that the transmitted OTP length can be $0 \leq k < N$ bits less than the message length N while still retaining perfect secrecy. Although we might not be able to achieve this reduction all the time, our method *never expands* the transmitted OTP. At worst, our transmitted OTPs are of the length of the message. We first prove an easier case where the OTP could be less than the message by 1-bit and the same idea is employed for the k -bit reduction. We make use of our earlier observation that the OTP encryption compromises its length in its traditional usage which we can actually avoid.

Theorem 1. *For every message of length N bits, it is equally likely that the transmitted OTP was of length $N - 1$ or N bits while still retaining perfect theoretical secrecy.*

Proof. We shall prove this result by constructing a (modified) protocol (Figure 2) where Alice and Bob exchange a message of length N by using an OTP. However, in this modified protocol, there is a 50% probability that the transmitted OTP had a length of $N - 1$ or N bits while still retaining perfect secrecy. We guarantee perfect secrecy for all the N bits of the message. The protocol works as follows:

Step 1: Alice performs a coin flip with a perfect coin. If it falls **HEADS**, she constructs an OTP of length N and if it falls **TAILS** she constructs an OTP of length $N - 1$. It is assumed that Alice has access to a perfect random source to construct the OTP in either events.

Step 2: Alice communicates the OTP through a secure channel to Bob.

Step 3: On some later day, Alice intends to send a message of length N bits to Bob. If the OTP she generated has $N - 1$ bits, she appends an additional bit at the end of the OTP. This additional bit is set to 1 if the length $N - 1$ is **ODD** and to 0 if $N - 1$ is **EVEN**. In case the

OTP already has N bits, Alice forces the N^{th} bit to 0 if $N - 1$ is **ODD** and to 1 if $N - 1$ is **EVEN**.

Step 4: Alice then performs the XOR operation of the message with the resulting OTP to yield a cipher-text C which has N bits. She transmits C on the insecure public channel to Bob.

Step 5: Bob receives C . Bob checks to see if the OTP he had earlier received from Alice has sufficient bits to decrypt the message. In other words, does it have N bits or $N - 1$ bits. In case the OTP has $N - 1$ bits, he does the exact same trick which Alice did i.e. appends an additional bit and sets it to 1 or 0 depending on whether $N - 1$ is **ODD** or **EVEN** respectively. If the OTP already has N bits, Bob forces the N^{th} bit to 0 if $N - 1$ is **ODD** and to 1 if $N - 1$ is **EVEN**.

Step 6: Bob decrypts C by performing an XOR with the modified OTP and obtains the message.

We need not prove the perfect secrecy of the first $N - 1$ bits as Shannon’s arguments hold. We need to prove that the N^{th} bit is perfectly secure. We shall analyze the situation from the eavesdropper Eve’s perspective. Eve knows of this entire protocol. Eve intercepts the cipher-text C which is of length N bits. She knows that there is a 50% probability that it came from an OTP which originally had $N - 1$ bits or N bits. She has no other strategy but to make a random guess and the probability of success is 50%. Hence, her guess of the N^{th} bit is no better than a 50% success. This proves the perfect secrecy of the N^{th} bit. \square

While this result seems highly theoretical and of little practical value, it actually shows an interesting aspect of the OTP which has been taken for granted. The fact that the length of the OTP contains *information* is usually neglected. Our proof was aimed at achieving theoretical security for one additional bit by using the Least Significant Bit (LSB) of the length of the OTP (by **ODD** we mean LSB= 1 and **EVEN** we mean LSB= 0) and we can do this half of the time. The natural question to ask is – can we make use of the other bits of the length?

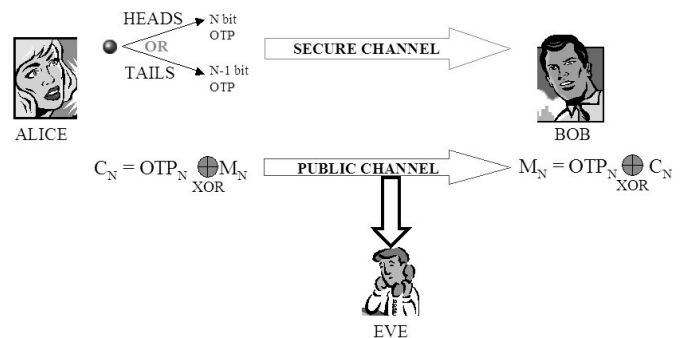


Figure 2: The protocol for the transmitted OTP length reduction by 1 bit

3.1 k Bit Reduction in the Length of the Transmitted OTP

Theorem 2. *For every message of length N and $\lceil \log_2(N - k) \rceil + 1 \geq k$ ($N, k > 0$ are integers), it is possible that the transmitted OTP had one of the lengths $N - k, N - k + 1, \dots, N - 2, N - 1$ or N with respective probabilities $2^{-k}, 2^{-k}, \dots, 2^{-k}$ or $1 - k2^{-k}$ while still retaining perfect theoretical secrecy.*

Proof. We generalize the aforementioned argument for a k -bit reduction in the length of the transmitted OTP. The case $k = 0$ would imply the conventional OTP and there is nothing new to prove. Let $k > 0$. Let the binary representation of the numbers $N, N - 1, N - 2, \dots, N - k$ be the following: $\langle A0_{m_N} \dots A0_2A0_1 \rangle, \langle A1_{m_{N-1}} \dots A1_2A1_1 \rangle, \langle A2_{m_{N-2}} \dots A2_2A2_1 \rangle, \dots, \langle Ak_{m_{N-k}} \dots Ak_2Ak_1 \rangle$ where each of the Ai_{m_j} is binary for all i and j . Also $m_N = \lceil \log_2 N \rceil + 1, m_{N-1} = \lceil \log_2(N - 1) \rceil + 1, \dots, m_{N-k} = \lceil \log_2(N - k) \rceil + 1$ (note that $m_{N-k} \geq 1$ since $N > k$). Alice has a $(k + 1)$ -sided biased coin which produces OTPs of length $N, N - 1, N - 2, \dots, N - k$ with probabilities $1 - k2^{-k}, 2^{-k}, 2^{-k}, \dots, 2^{-k}$ respectively. Assume that the OTP thus generated is $N - r$ bits long where $0 < r \leq k$. She transmits this $N - r$ bits long OTP to Bob over a secure channel. For encryption of a N bit message, Alice first has to lengthen the OTP to N -bits (if it is not already a N -bit OTP). She does this by appending the required amount of bits (in this case r bits) at the end of the OTP and set them to zero. She then forces the last k bits of the OTP to the bits $\langle Ar_k \dots Ar_2Ar_1 \rangle$. Only for the instance when Alice is generating an OTP of length N bits, she ensures that the last k bits never have the same sequence as the other k OTPs before sending it to Bob on the secure channel. Moreover, she ensures that the remaining available combinations for the last k bits which are $2^k - k$ in number have each a probability of occurrence $\frac{1}{2^k - k}$ (equally likely). This way, the last k bits of all the OTPs are perfectly random because the probability of obtaining any particular binary sequence for the last k bits is 2^{-k} . The rest of the protocol remains unchanged.

With this, we have proved by construction that it is possible for the transmitted OTP to have a length lesser than the message length with a non-zero probability while still attaining perfect theoretical secrecy. One can also verify that for the case $k = 1$, this essentially reduces to the earlier protocol. \square

3.1.1 An Example: $N = 9$ and $k = 3$

An example helps in understanding the protocol. We shall take $N = 9$ and $k = 3$. The condition $N \geq k + 2^{(k-1)} \Rightarrow 9 \geq 3 + 2^{(3-1)} \Rightarrow 9 \geq 7$ is satisfied.

Alice generates OTPs of length 9, 8, 7 and 6 with probabilities $1 - 3 \cdot 2^{-3} = 0.625, 2^{-3} = 0.125, 2^{-3}$ and 2^{-3} respectively. The binary representations of 8, 7 and 6 according to the protocol are $\langle 1000 \rangle, \langle 111 \rangle$ and $\langle 110 \rangle$. In order to encrypt a 9-bit message, Alice has to create 9-bit OTPs from the 8, 7 and 6 bit OTPs. Let

us assume that she has generated an 8-bit OTP. She first appends a zero bit at the end of the OTP to make it of length 9. She then re-writes the last $k = 3$ bits of the OTP by the last three bits of $\langle 1000 \rangle$ (namely 000). She then encrypts the message with the resulting 9-bit OTP.

On the other hand, if Alice had generated a 9-bit OTP, she tosses another 3-sided coin which *never* give out the following patterns: $\langle 000 \rangle, \langle 111 \rangle$ and $\langle 110 \rangle$. It produces the other patterns $\langle 001 \rangle, \langle 010 \rangle, \langle 011 \rangle, \langle 100 \rangle$ and $\langle 101 \rangle$ with a probability of $\frac{1}{5}$ each. She uses the resulting 3-bit pattern to overwrite the last 3 bits of the 9-bit OTP. She then performs encryption as before.

As it can be seen, the probability of occurrence of the patterns $\langle 000 \rangle, \langle 111 \rangle$ and $\langle 110 \rangle$ for the last 3 bits of the OTP would be $2^{-3} = 0.125$ because this is exactly the probability that an 8, 7 or 6-bit OTP would be generated. In the event of a 9-bit OTP being generated (with a probability of 0.625), the patterns $\langle 001 \rangle, \langle 010 \rangle, \langle 011 \rangle, \langle 100 \rangle$ and $\langle 101 \rangle$ can each occur with a probability of $0.625 \times \frac{1}{5} = 0.125$. Thus, it is clear that all possible 3-bit patterns (there are eight of them) at the end of the OTP occur with an equal probability of 0.125 and hence ensures perfect secrecy.

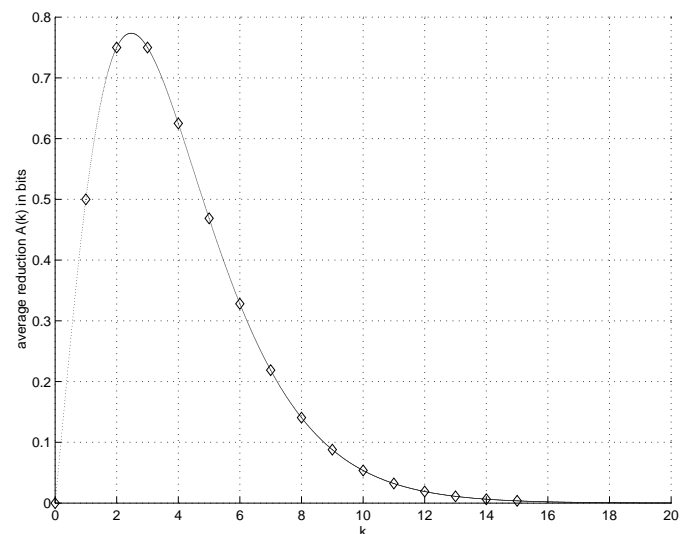


Figure 3: Average reduction in length of OTP $A(k) = k(k + 1)2^{-(k+1)}$ vs. k . The diamond marked points are those for which k is an integer. The maximum occurs at $k = 2$ and $k = 3$ (average reduction = 0.75 bits).

3.1.2 Average Reduction in Length of Transmitted OTP

It is interesting to see that for larger reductions (larger values of k), the probability of obtaining a reduction re-

duces. The average reduction is given by:

$$\begin{aligned} A(k) &= 1.2^{-k} + 2.2^{-k} + 3.2^{-k} + \dots + k.2^{-k} \\ &= \frac{k(k+1)}{2} 2^{-k} \\ &= k(k+1)2^{-(k+1)} \text{ bits.} \end{aligned}$$

Figure 3 shows the plot of average reduction $A(k)$ vs. the reduction k in bits. The best average reduction is for $k = 2$ and $k = 3$, where we get 0.75 bits of reduction. Note that in our protocol, we have not violated the assumption that the OTP is perfectly random and otherwise uncompressible.

3.2 Compression of Transmitted OTP Based on Length Information

Alternatively, we can say that the transmitted OTP is *compressible* to the extent it's length information allows. We provide a method of compressing the transmitted OTP given the fact that the messages to be encrypted are always of length N , which is publicly known.

Alice generates an N bit OTP. If the last bit is 1, she deletes it to create an $N - 1$ bit OTP. If the last bit is 0, she deletes all bits which are zeros from the end up to and including the bit which is 1. If the OTP has no 1s in it, then Alice transmits it as is. As an example, consider the $N = 10$ bit OTP '1011001001'. Since the last bit is 1, Alice deletes to create the 9-bit OTP '101100100'. If the $N = 10$ bit OTP happens to be '1011001000', by the above rule, Alice obtains the 6-bit OTP - '101100'. Alice transmits the resulting *compressed* OTP across the secure channel to Bob. Since the length of messages to be encrypted is always $N = 10$, Bob decompresses the received OTP to N bits by reversing the rule. In other words, if Bob receives an $N - 1$ bit OTP, he appends a 1 to make it N bits. If the received OTP is of length $N - k$ bits, where $k > 1$, he appends a 1 followed by $k - 1$ zeros. Thus, the OTP is correctly decompressed by Bob in all instances. Table 1 illustrates the compression method on all possible OTPs of length 4. In practice, the length of OTPs used are much larger, but this serves as a good example. Decompression is easy to see and is omitted for the example.

An interesting thing to observe is that the OTP is compressed for all instances except the case when it has no 1s. There is only one such OTP (all 0s) which is uncompressed by this scheme. At a first glance, one might wrongly infer that we are contradicting the counting argument. However, this is not the case. The counting argument applies only to *memoryless* lossless compression algorithms. In our case, Bob has the *a priori* information about the length N (publicly known) and hence it is not memoryless.

3.2.1 Average Reduction in Length of Transmitted OTP

What are the reductions obtained by this method? We can see that for 50% of the instances, there is a reduction by 1-bit only (the last bit is 1 for 50% of the cases). Among the remaining 50%, one instance is uncompressed (the OTP with all bits 0s) and one instance has a maximum reduction of all N bits (the OTP with a 1 followed by $N - 1$ zeros). For the remaining OTPs, the compression ratios vary depending on the number of 0s in the end. For example, an OTP with m zeros in the end has a reduction of $m + 1$ bits. There are 2^{N-m-1} such N -bit OTPs which will compress to an OTP of length $N - m - 1$ bits, a reduction by $m + 1$ bits. Thus, the average reduction is given by:

$$\begin{aligned} B(N) &= \frac{1}{2} \sum_{m=0}^{N-1} (m+1)2^{-m} \\ &= \frac{1}{2} \{2N(1 - 2^{-N}) - \sum_{m=0}^{N-1} 2(1 - 2^{-m})\} \\ &= N(1 - 2^{-N}) - \sum_{m=0}^{N-1} 1 + \sum_{m=0}^{N-1} 2^{-m} \\ &= N(1 - 2^{-N}) - N + 2(1 - 2^{-N}) \\ &= 2 - (N + 2)2^{-N} \text{ bits.} \end{aligned}$$

Figure 4 shows the plot of average reduction $B(N)$ vs. the reduction N in bits. It is interesting to observe that

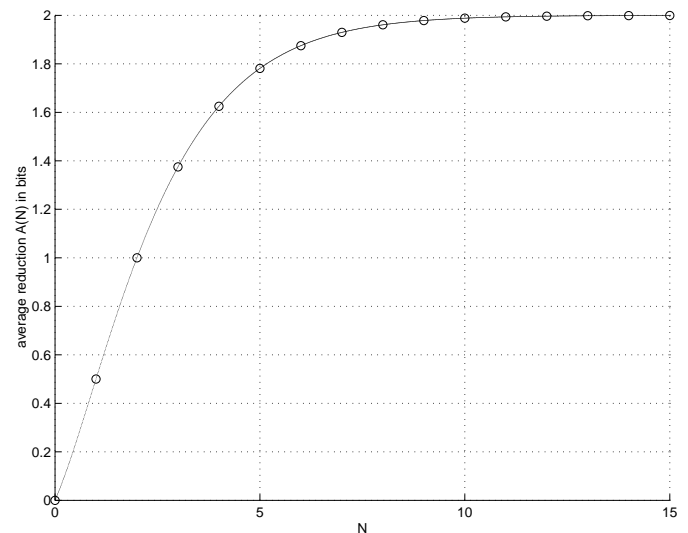


Figure 4: Average reduction in length of OTP $B(N) = 2 - (N + 2)2^{-N}$ vs. N by the second method. The circle marked points are those for which N is an integer.

the average reduction for the first method is independent of the length N whereas the average reduction for the second method is independent of the reduction parameter k . The first method allows one to choose the maximum reduction desired (k) whereas the second method does

Table 1: The transmitted OTP and the bit reductions obtained by the compression method on all possible OTPs of length 4. The average reduction is $\frac{26}{16} = 1.625$ bits.

OTP	Transmitted OTP	Reduction (bits)	OTP	Transmitted OTP	Reduction (bits)
0000	0000	0	1000	–	4
0001	000	1	1001	100	1
0010	00	2	1010	10	2
0011	001	1	1011	101	1
0100	0	3	1100	1	3
0101	010	1	1101	110	1
0110	01	2	1110	11	2
0111	011	1	1111	111	1

not have this feature. However, the second method yields a larger average reduction in the length of the transmitted OTP.

4 An Alternate Interpretation of the OTP as a Private-Object

In the previous section, we saw how we made use of the length of the OTP in obtaining a reduction in its length. The length happens to be a particular *feature* of the OTP, as if it were an *object*. This leads us to the notion of a private-object which we define as follows.

Private-Object: Any object which is known only to the sender and the receiver is defined as a *private-object*.

The above definition is very broad. The object may have any embodiment, not necessarily digital in nature. The object could be a real physical thing or it could be an one time pad (could even be multi-dimensional). An important thing to note is that every private-object enables theoretically secure communication. This leads us to a notion of ‘entropy’ of the private-object which is determined by the *number of independent True/False statements* that can be made about the object without revealing any information about it. The way a message is transmitted by means of a private-object is described below.

Alice and Bob share a private-object P , known only to them. Alice intends to send a message M (as an example, the statement ‘COME AT 8 PM’ to Bob). The protocol is as follows:

Step 1: Alice converts message M into binary representation (using a publicly known dictionary). Say ‘COME AT 8 PM’ translates to $M = 00101110101$.

Step 2: Alice substitutes $0 = TRUE = T$ and $1 = FALSE = F$. Therefore $M = TTFTFFFTFTF$.

Step 3: For each bit of the message M , Alice makes statements about the private-object P which is

TRUE (if the bit is T) or FALSE (if the bit is F) to obtain the cipher-text C . In other words $C = \langle statement_1 \rangle \langle statement_2 \rangle \dots \langle statement_{10} \rangle$ where $\langle statement_1 \rangle$ is TRUE, $\langle statement_2 \rangle$ is TRUE, $\langle statement_3 \rangle$ is FALSE etc. As a crude example, assume that the private-object is a physical object which has 3 eyes, 2 hands, 5 legs etc. Alice could make a statement like ‘ P has 3 eyes’ which is TRUE or a statement like ‘ P has 4 legs’ which is FALSE (the number of legs and hands in this hypothetical object are independent of each other).

Step 4: Bob receives the cipher-text C which is a collection of statements about P . He verifies each statement and determines whether they are TRUE (T) or FALSE (F). He obtains a string of T s and F s by this process ($M = TTFTFFFTFTF$).

Step 5: Bob substitutes $0 = TRUE = T$ and $1 = FALSE = F$ in M to obtain the binary message $M = 00101110101$.

Step 6: Bob looks up at the dictionary for M to obtain the message ‘COME AT 8 PM’.

The OTP can be thought of as a private-object P and the above protocol can be used for secure communication. For our previous example of Section 2, the set of statements which Alice would make are $C =$ ‘the first bit of the OTP is 1’, ‘the second bit of the OTP is 0’ \dots ‘the tenth bit of the OTP is 0’. Bob verifies these statements since he has the OTP with him and obtains the correct message.

5 Private-Object Cryptography

In the previous section, we saw how the OTP could be viewed as a private-object and statements about the object can be made to transmit information securely. So long as the statements are *independent* of each other, we are guaranteed to achieve perfect secrecy. This is because every statement encrypts one bit of the message and is

making use of a unique feature of the private-object. For the OTP, every bit is its unique independent feature. For private-objects of the real physical world, the features could be the number of edges or the number of faces etc. Determining the number of *unique* and *independent* features in a physical object might be difficult. This means that the *entropy* of the object is difficult to compute. The amount of information that can be securely transmitted by this method is upper bounded by the entropy of the object in bits. Private-key or symmetric-key cryptography is a subset of Private-object cryptography where the key happens to be a set of bits on which various mathematical operations are made. In effect, every private-key crypto-system is only making statements about the key which is the private-object. Since the key of a private-key is usually much shorter than the message, the statements are not *independent* of each other. They formally map to complex statements about the key.

Every symmetric-key crypto-system can encrypt exactly one binary message having a length equal to the entropy of the crypto-system with perfect theoretical secrecy. One can always make a certain number of *unique* and *independent* statements about the crypto-system. We can treat the crypto-system with its unique parameters as a private-object having a certain entropy. These statements are *finite* in number and can be used to communicate a finite length binary message with perfect secrecy (equivalent to an OTP of the same entropy). The length of the message can be at most equal to the entropy of the crypto-system without sacrificing Shannon security. Finding the entropy of the crypto-system may not always be easy.

Another interesting off-shoot is the definition of the *entropy of an object* of the real world. We can define the entropy of an object as the number of bits of information that can be transmitted with perfect secrecy by making independent statements about the object. In other words, we claim that there exists a mapping from every object of the real world to an OTP and the entropy of that OTP is the entropy of the object. It may be hard in practice to determine the entropy of objects. It is not known whether this notion of entropy is the same as Shannon's entropy.

5.1 Investment of N-bits of Secret

Let us now relax the *perfect secrecy* constraint since we need to send long keys (if not as long as the message) for achieving this. Assume that we have a fixed bit-budget, say N bits of secret. We wish to know what is the best private-object to invest these N bits of secret so as to achieve a high encryption efficiency. Here, we do not wish to achieve perfect secrecy, but breaking the system should be very hard. Here, we are being vague in our definition. It suffices to say that we wish to obtain a method where currently known methods of cryptanalysis have a hard time in breaking, if not impossible. We wish

to propose using a Formal Axiomatic System (FAS) for investing these N bits. This part of the paper is mainly a motivation towards potential future research.

5.2 FACtS: Formal Axiomatic Cryptographic System

A Formal Axiomatic System or FAS for short, refers to a system of axioms and rules of inference which together define a set of theorems [2]. An example of a FAS is Typographical Number Theory (TNT). Hilbert's program was to completely formalize the whole of Mathematics using TNT. This ambitious plan was derailed by Gödel who proved that all consistent and sufficiently powerful axiomatic systems contain *undecidable* propositions. Because of this, Formal Axiomatic Systems are fascinating objects.

We can view a FAS in another interesting way – the *compression* view-point. A FAS is actually a *compressed* version of all its theorems which can be proven within the system. It is this viewpoint that motivates us to consider an FAS as a private-object which is shared between Alice and Bob. If Alice were given a bit-budget of N bits, she could invest it in the construction of a FAS which is *consistent* and *sufficiently strong*. These are the only two requirements. She would have to define a set of axioms and rules of inference to completely specify the FAS. She shares this as a *private-object* with Bob over a secure channel. The way Alice and Bob can now exchange information is to make statements or strings in the FAS. The receiver can *verify* whether a particular statement or string is TRUE or FALSE in the FAS which they share. If it is TRUE, then it implies that the string is a *Theorem* and the bit conveyed is 0. If the string is FALSE, then it is a *Non-theorem* and conveys the bit 1. We basically use the private-object paradigm with Theorems and Non-theorems of the FAS acting as binary representations for 0 and 1 respectively. We name such a system as Formal Axiomatic Cryptographic System (FACtS). Figure 5 shows the string space of a FAS [2].

Since the FAS is sufficiently strong, it would contain Gödelian statements which are undecidable (the system is incomplete). We believe that it may be possible to *confuse* and *diffuse* the cryptanalyst by a clever use of Gödelian statements in the cipher-text. This is a speculation on our part, because we do not know of any procedure which would enable us to construct such statements in large numbers.

One of the biggest advantages of such a set-up is the difficulty of breaking the system for Eve using brute-force attack. In conventional systems such as the RSA and other public-key and private-key methods [4], brute-force attack would involve trying out all possible keys in the *key-space*. For example, if the key length is 128-bit, it would mean trying out 2^{128} (a huge number) guesses for the private-key. A computer could mechanically try out these number of possibilities until it found the

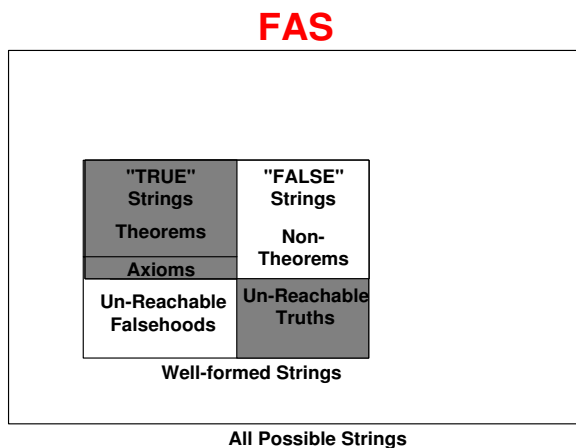


Figure 5: The string-space of a Formal Axiomatic System (FAS) [2]

right key. This would probably take a long time but with a number of computers in parallel or by using Quantum computers, this time could be sufficiently reduced. The important thing to realize in this scenario is that there is a *mechanical procedure* for trying out all the combination and with exponential increase in computational power over time, it could be eventual broken (eg: the RSA-128 is already broken). In our system, the equivalent would be to try out all possible Formal Axiomatic Systems of a given length N . However there would be several systems which are *duds*, those that are inconsistent or meaningless. Computers which are designed to try out different FASs might have a difficult time to find out inconsistencies. They might have to deal with the Turing Machine Halting problem [3].

6 Conclusions

To summarize, the central contribution of this paper is a new result in the OTP literature. We have shown that the length of the OTP which is traditionally compromised in encryption could be avoided. We proved that it is possible to reduce the key-length of the transmitted OTP (which is perfectly random and uncompressible otherwise) while still retaining perfect secrecy. Even though this reduction is small, it is nevertheless useful in saving band-width for crypto-systems which use OTPs on a regular basis (we showed that we never expand the OTPs in any case unlike compression algorithms which always expand some). We also gave an alternate method of compression of the transmitted OTP based on the length information. We obtained analytical expressions for the average reduction of the length of the OTP (in bits) for both the methods.

We have conceived a new paradigm called private-object cryptography which makes use of statements about an object (private to the communicating parties) for secure message transmission and showed how the OTP can be re-interpreted in this new paradigm. We

also claimed that all existing private-key crypto-systems are a form of private-object cryptography. Further, they are in essence making statements about the secret key. We believe that these statements are not independent but are necessarily more complex. We then suggested the investment of N bits of secret in a FAS. The verification of strings or statements of the FAS as theorems or non-theorems could convey a bit of information. It may be the case that the structure of the FAS and the space of theorems and non-theorems could be designed so that it is *sufficiently random* for cryptographic purposes. More research needs to be done in these directions.

Acknowledgements

Nithin Nagaraj would like to express sincere gratitude to the Imaging Technologies Lab, John F. Welch Technology Center, General Electric Global Research, Bangalore for providing support for part of this research work.

References

- [1] Y. Dodis and J. Spencer, "On the (non)Universality of the One-Time Pad," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pp. 376-388, 2002.
- [2] D. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*. 20th Anniv. edn. Basic Books, 1999.
- [3] J. E. Hopcroft, R. Motwani and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. 2nd edn. Addison Wesley, 2000.
- [4] A. Menezes, P. C. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*. Florida: CRC Press, Boca Raton, Florida, 1996.
- [5] D. Raub, R. Steinwandt and J. Mueller-Quade, "On the Security and Composability of the One Time Pad," *Cryptology ePrint Archive*, Report 2004/113. (<http://eprint.iacr.org/2004/113/>)
- [6] D. Salomon, *Data Compression: The Complete Reference*. New York: 2nd edn. Springer-Verlag, 2000.
- [7] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.
- [8] C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal*, vol. 28, pp. 656-715, 1949.



Nithin Nagaraj was born on November 16, 1977 in India. He received his Bachelor's degree in Electrical and Electronics Engineering from Karnataka Regional Engineering College, Surathkal, India, in 1999. He received the Master's degree in Electrical Engineering from Rensselaer Polytechnic

Institute (RPI), Troy, NY, in 2001, working in the Center for Image Processing Research Lab at RPI on wavelet based color video compression schemes.

He worked as a Research Engineer in the Image Communications Center of Excellence at the Imaging Technologies Laboratory, John F. Welch Technology Center, GE Global Research, Bangalore, India from 2001-2004 where he conducted research in the areas of entropy coding, wavelet transforms, video compression, medical image processing, data embedding, and image segmentation.

He is currently a Ph. D. student at the Mathematical Modelling Unit, School of Natural and Engineering Sciences, National Institute of Advanced Studies, IISc Campus, Bangalore. His areas of research interest include Coding, Cryptography and Chaos Theory.



Vivek Vaidya received a B.Sc in Computer Science from Washington State University in 2002. He is currently working at the Imaging Technologies lab in GE Global Research, Bangalore. His current research interests include: visualization, segmentation, virtual reality, and cryptography.



Prabhakar G. Vaidya is currently the Dean and Professor, School of Natural Sciences and Engineering, Professor of Mathematical Modelling at NIAS. He obtained his B.E. from University of Bombay and M.Sc. (Engg.) and Ph.D. (Acoustics) from the Institute of Sound and Vibrations Re-

search, University of Southampton. He has worked at Lockheed, Boeing and NASA and taught at Purdue and Washington State University for many years before returning to India. His research publications have spanned from sonic Booms to Pathogenesis. His work at NIAS involves applications of Advanced Mathematics to diverse problems. Recent papers with this theme are in the areas of chaotic synchronization and cryptography, speaker identification, Advanced image processing and cardiophysics.