One Time Pad

Keane J. Moraes

History

The One Time Pad (OTP) was first described by a California banker named Frank Miller. He published his scheme in the *Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams.* In it, he describes the first one-time pad system, as a superencipherment mechanism for his telegraph code[1]. 35 years later, the scheme was reinvented by Gilbert Vernam and Joseph Mauborgne in 1917 and later patented in 1919. At the time, Vernam worked at Bell Telephone Laboratories and Mauborgne worked in the US Army Signal Corps. Vernam invented a device that would exclusive-OR keystream bits from a paper tape with the Baudot code generated by letters typed on a keyboard; he and Mauborgne realized that if the keystream tape characters were (a) perfectly random, and (b) never reused, "the messages are rendered entirely secret, and are impossible to analyze without the key"[2].

Modern Usage

The NSA called Vernam's 1919 one-time tape (OTT) patent "perhaps one of the most important in the history of cryptography. AT&T marketed the Vernam system in the 1920s as a mode for secure communications. The production, distribution and consumption of enormous quantities of one-time tapes limited its use to fixed stations (headquarters or communications centres). It was not until the Second World War that the US Signal Corps widely used the OTT system for its high level teleprinter communications. In 1923, the system was introduced in the German foreign office to protect its diplomatic messages (see image right). For the rest time in history, diplomats could have truly unbreakable encryption at their disposal. Unfortunately they made the mistake of using a simple mechanical machine to produce the 'random' digits.[3]

The hotline between Moscow and Washington D.C., established in 1963 after the Cuban missile crisis, used teleprinters protected by a commercial one-time tape system. Each country prepared the keying tapes used to encode its messages and delivered them via their embassy in the other country. A unique advantage of the OTP in this case was that neither country had to reveal more sensitive encryption methods to the other.

Information Theoretic Security

Claude Shannon proved that the OTP had "perfect secrecy" in his 1949 paper *"Communication Theory of Secrecy Systems"*. To understand what this means and how he proved it, we must first understand what perfect secrecy is.

Perfect Secrecy

Let us define an adversary named ATK as a malicious third-party who is eavesdropping on a communication channel between 2 parties (say Alice and Bob).

What can ATK do? He can intercept encrypted messages between Alice and Bob.

What does ATK know? He knows what encryption and decryption scheme Alice and Bob employ to encrypt their message and all the ciphertexts sent from Alice to Bob are intercepted by ATK.

What is ATK's goal? He wants to be able to break the encryption scheme employed by Alice and Bob so that he can decipher their transmissions.

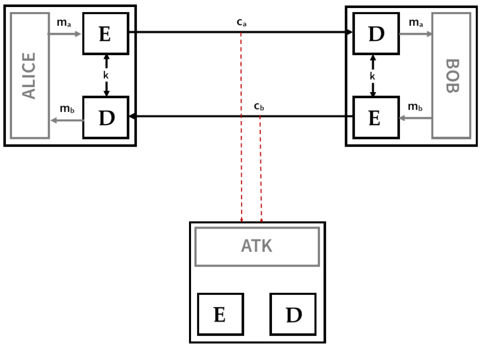
Alice and Bob use a pair of algorithms $(\mathbb{E}, \mathbb{D})$ as an encryption scheme (that is defined over a keyspace, a message space and a cipher text space $(\mathbb{K}, \mathbb{M}, \mathbb{C})$. **They both have a shared key 'k' that is unknown to ATK.**

[1] Steven M. Bellovin. *Frank Miller: Inventor of the one-time pad.* Cryptologia, 35(3):203–222, July 2011.

[2] Ibid.

[3] about how the U.S. Army Security Agency (ASA) were able to exploit this flaw through The Solution and Explotation of the German One Time Pad System.

We prepare an experiment to test this system. Alice prepares 2 messages m_{A0} encrypted as c_{A0} and m_{A1} encrypted as c_{A1}. Bob similarly encrypts m_{B0} as c_{B0} and m_{B1}as c_{B1}. Note that $m_{A0} \neq m_{A1}$ and $m_{B0} \neq m_{B1}$.
They each flip a fair coin. If Alice gets heads, she sends c_{A0} as the transmitted ciphertext c_A as shown below. If tails she sends c_{A1} as the transmitted ciphertext. Bob does a similar coin toss to determine his transmitted ciphertext.



Perfect Secrecy is defined by Claude Shannon as follows : Given a ciphertext c, the probability (for ATK) that c is an encryption of m_0 is the same as that of an encryption of m_1.

$$P\Big[\mathbb{E}(k, m_0) = c\Big] = P\Big[\mathbb{E}(k, m_1) = c\Big]$$

So in our system, if Alice and Bob employ a perfectly secure cipher. Then for ATK

$$P\Big[\mathbb{E}(k, m_{A0}) = c_A\Big] = P\Big[\mathbb{E}(k, m_{A1}) = c_A\Big]$$

The same argument applies to Bob's messages. This implies that (for ATK) the cipher text is equally likely to come from either m_{A0} or m_{A1}. To emphasize how strong this security parameter is, let us generalize. If ATK could not distinguish m_{A0} from m_{A1}, then by inference he cannot distiguish m_{A0} or m_{A1} from $m \overset{\text{r}}{\leftarrow} \mathbb{M}$ (any random message from the message space). In essence, ATK's knowledge of the message is as good as guessing any random message in the message space.

A heuristic argument for this is that **the cipher test leaks no 'information' about the plaintext**. Thus, no matter how powerful ATK is, if he only intercepts ciphertexts encrypted with a perfectly secure cipher $(\mathbb{E}, \mathbb{D})$, he cannot mount a cipher-text only attack to break the encryption. **A perfectly secure cryptosystem is considered cryptanalytically unbreakable** if ATK can mount ciphertext-only attacks.

Encryption and Decryption Algorithms

Encryption

Encryption in OTP is done by a XOR of the plaintext and the key. Given a message 'm' and a key 'k', the cipher text c is given by

$$c = k \oplus m$$

If we are to do a bitwise XOR of the key and the plaintext then c_i (the i^{th} bit of the cipher text) can be computed by $(k_i + m_i) \mod 2$. Analysing the 2-bit XOR truth table will show this is true.
By properties of XOR[4], we can see that given an message-ciphertext pair (m, c) , 'k' is

$$k = m \oplus c$$

PROOF THAT OTP HAS PERFECT SECRECY

We will make use of the fact that OTP is deterministic (i.e for any message m in $\mathbb{M}$, there is one and only one ciphertext c in $\mathbb{C}$ such that $\mathbb{E}(k, m) = c$ for a fixed key 'k').

To find $P[\mathbb{E}(k, m) = c]$ for OTP, we define it as follows

$$P\Big[\mathbb{E}(k, m) = c\Big] = \frac{|k|^{*} \text{ for which } \mathbb{E}(k, m) = c}{|\mathbb{K}|}$$

(* - $|k|$ means no. of keys)

If we can show that $\forall$ message-ciphertext pairs(m, c), $|k|$ is constant then QED.
In OTP, $k = m \oplus c$, and by properties of XOR, there exists one and only one k for which this is possible. So for a given pair (m, c); $|k| = 1$. Thus,

$$P\Big[\mathbb{E}(k, m_0) = c\Big] = P\Big[\mathbb{E}(k, m_1) = c\Big] = \frac{1}{|\mathbb{K}|}$$

Hence proven.

Code Implementation:

We will perform a bitwise XOR. We must take into account the conversion of the ASCII character values into 8-bit binary.
Let E_t be the cipher text
P_t be the plain text
K_t be the key
$\overline{P_t}$ be the binary representation of the plaintext where each character is converted to 8-bit binary having a length of 'n'.
$\overline{K_t}$ be the binary representation of the key where each character is converted to 8-bit binary having a length of 'm'.
$\overline{E_t} = E_1 E_2 E_3 \ldots E_n$ where each E_i represents a bit of the encrypted text.
$\overline{K_t} = K_1 K_2 K_3 \ldots K_m$ where m is the length of the key's binary.
i is the **itre position** of the plain text's binary where $1 \leq i \leq n$
j is the **itre position** of the key's binary where $1 \leq j \leq m$
The i^{th} character of the cipher text's binary is -

$$\overline{E_i} = \overline{P_i} \oplus \overline{K_i}$$

Another way of computing it is -

$$\overline{E_i} = (\overline{P_i} + \overline{K_i}) \mod 2$$

The $\overline{E_t}$ is then converted into hexadecimal with spaces between every 8 bits of the cipher text and outputed.

[4] To see this property's proof, refer *Lemma 1.1* of the Encryption section of Feistel in Modern Ciphers.

Why hexadecimal?
This is becasue the 8-bit encrpyted text will not always be an ASCII printable character. An example will clarify this. Suppose our message contained a letter ‘H’ some position ‘p’ and suppose that the key has the character ’W’ at the same position. The XOR of the message (01001000) and the key would be (01010111) would be (00011111) which is 31 in decimal and NOT an ASCII printable character.
So converting back into String is not possible. The remaining choices are hexadecimal, octal, decimal or just output as binary. Hexadecimal is the most compact and thus is the most logical choice.

Why spaces between the 8 bits?
Another example will help. Let the ciphertext in bits be ‘00001010 00010100’ which corresponds to ‘a 14’ in hexadecimal. Suppose we did not put spaces and fed this output into the decryption machine. The decryption algorithm first converts hexadecimal to 8 bit binary before XORing with the plaintext. The converted cipher text would be ‘00001010 00000001 00000100’. But our initial cipher text had only 2 bytes while this one has 3 bytes. Note that this is not a problem implicit in hexadecimal, rather this problem persists with any number base and hence we have to space out the bytes.

Decryption

Decryption is just as simple as encryption. To retrive the message m, simply perform

$$m = c \oplus k$$

Code Implementation

First the hex string is converted back into 8-bit binary and then XORed with the key. This gives us the plaintext in 8-bit binary which we know for a fact converts into printable ASCII values. After converting we output the String. Unfortunately, the spaces in the original plaintext will not have survived. The decryption process uses the same function as the encryption process, the only difference being the format from which the 8-bit binary us derived from.

Criteria for Perfect Secrecy (in implementation)

Although we have proved that OTP has perfect secrecy, certain measures must be taken during physical implementation to ensure that it is indeed ‘unbreakable’. The criteria are listed below:

1. Each key-message pair must be used once and only once. **The key must never be reused.**

2. The key must be as long as the plaintext. (If not, we violate Rule No. 1). Generally, a good rule of thumb is to keep key lengths ≥ 128 bits in order to be immune to key search attacks.

3. The key must be randomly generated.

4. The key must be kept secret - leakage of the whole key or even part of it will comprimise the security of the system.

Stream Ciphers

Bad News Lemma
Shannon also proved that the key length must be as long as the plain text length in order for OTP to have perfect secrecy. This is a problem because if we had to encrypt a message that is gigabytes large, we would have to operate and store keys that are also gigabytes large. This is the major drawback to OTP which makes it unfeasible for real world implementation. So a workaround was developed - Stream Ciphers.

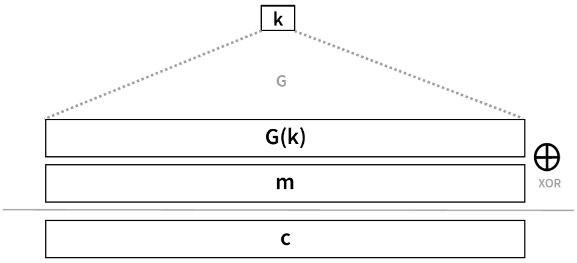
To introduce the idea of a stream cipher, one must first understand what a PRG is :

Psuedo Random Generators (PRGs):

A Pseudo Random Generator is a *deterministic* function that maps any given m-bit binary strings to n-bit binary strings where $n \ggg m$. (for e.g. $m \approx 10^3$ (kilobytes),$n \approx 10^9$ (gigabytes)). Formally, this PRG function $\mathbb{G}$ is defined as follows -

$$\mathbb{G} : \{0,1\}^m \rightarrow \{0,1\}^n$$

Now our keyspace $\mathbb{K}$ can be set to $\{0,1\}^m$. We randomly pick a key from this 'small' set and expand using our PRG to be as long as the message. This can then be XORed with the message.



Security of PRGs

To understand the security theorems of PRGs, let us introduce some terminology.

Unpredictability
A PRG is said to be unpredictable if given 'r' consecutive bits, an adversary cannot 'predict' the rest of the bits. Let $\mathbb{ALG}$ be some 'efficient' algorithm which computes the remaining bits in an $n-$bit given the first r bits. For simplicity, we say that efficient means the algorithm runs in polynomial time. Let P() be the probability of an event.
In formal notation, given any $n-$bit binary PRG output

$$\forall r \text{ where } 0 \leq r < n \nexists \text{ any 'efficient' } \mathbb{ALG} \text{ such that}$$

$$P\Big(\mathbb{ALG}\big[\mathbb{G}(k)|_{1\dots r}\big] \Rightarrow \mathbb{G}(k)|_{r+1\dots n}\Big) \geq 1/2 + \epsilon$$

where ϵ is some non-negligible constant (in practice $\epsilon \geq 1/2^{30}$ is non-negligible).

Statistical Tests
A statistical test is an algorithm that given an $n-$bit binary string will output whether it is random or not. Let $\mathbb{A}$ be such an algorithm

$$\mathbb{A}(\{0,1\}^n) = \begin{cases} 0 & \text{if the string is not random} \\ 1 & \text{otherwise} \end{cases}$$

Usually, more than one statistical test is used to test if a PRG's output is random or not.

Computational Indistinguishability
We say that two probability distributions are 'computationally' indistinguishable if $\nexists$ any 'efficient' statistical test $\mathbb{A}$ that can differentiate between them (i.e no test can say with a high probability that any given $n-$bit string is sampled from $\mathbb{P}_1$ or $\mathbb{P}_2$).

$$\text{Given probability distributions } \mathbb{P}_1 \text{ and } \mathbb{P}_2, \text{ if } \forall \text{ 'efficient' statistical tests } \mathbb{A}$$

$$\Bigg| P\Big[\mathbb{A}(x)_{x \xleftarrow{r} \mathbb{P}_1} = 1\Big] \;-\; P\Big[\mathbb{A}(x)_{x \xleftarrow{r} \mathbb{P}_2} = 1\Big] \Bigg| < \text{ negligible}$$

then $\mathbb{P}_1 \overset{\text{poly}}{\approx} \mathbb{P}_2$ (the notation states that they are indistinguishable)

We will use these to proceed with the definition of PRG security. There are two ways to show the security of PRFs and both are stated below.

ADVANTAGE

$$\mathbb{ADV}_{PRG}\big(\mathbb{A},\mathbb{G}\big) = \left| P\left[\mathbb{A}\big(\mathbb{G}(k)\big)_{k \xleftarrow{r} \mathbb{K}} = 1\right] \; - \; P\left[\mathbb{A}\big(x\big)_{x \xleftarrow{r} \{0,1\}^n} = 1\right] \right|$$

PRG is secure if $\forall$ ‘efficient’ tests $\mathbb{A}$, $\mathbb{ADV}_{PRG}\big(\mathbb{A},\mathbb{G}\big) <$ negligible

INDISTINGUISHABILITY

$$\{\mathbb{G}(k) \mid k \xleftarrow{r} \mathbb{K}\} \overset{\text{poly}}{\approx} \text{uniform}\big(\{0,1\}^n\big)$$

What do these mean?
Both these security theorems mean the same thing, they are simply two different methods of writing it. Each theorem states that a statistical test will identify a pseudorandom string to be random approximately the same no. of times that it identifies a random string to be random. Lets try to think of cases - $\because$ we take absolute value $\therefore$ $\mathbb{ADV}_{PRG} \in [0,1]$.

If advantage is close to 1, this implies that there is a notable difference between the probability that a pseudorandom string is called random $P\big[\mathbb{A}\big(\mathbb{G}(k)\big)_{k \xleftarrow{r} \mathbb{K}} = 1\big]$ and a random string is called random $P\big[\mathbb{A}\big(x\big)_{x \xleftarrow{r} \{0,1\}^n} = 1\big]$. This means that the test $\mathbb{A}$ was able to distinguish between the pseudorandom string and the random string (even if its the wrong distinction, it matters). So, our test $\mathbb{A}$ broke the PRG $\mathbb{G}$ ($\mathbb{G}$ is insecure).

On the other hand if the advantage is tending to 0 (i.e negligible), then the test $\mathbb{A}$ was not able to distinguish between the pseudorandom string and the random string. So our PRG $\mathbb{G}$ ‘fooled’ the test.

One more fundamental statement of the security of PRGs is that **all secure PRGs are unpredictable**. This proof is left as an open exercise to the reader. *(HINT : Develop a statistical test* $\mathbb{B}$ *around a predicting algorithm* $\mathbb{ALG}$ *and use the contrapositive of the statement to complete the proof.)*

The security theorems of PRGs allow us to define a security theorem for Stream ciphers which is the equivalent to that of Information Theoretic Security. This security parameter is called Semantic Security. This is not discussed here but the interested reader can find more information in the Further Reading section.

Real World Stream Ciphers

RC4
This stream cipher was developed by Ron Rivest of RSA Security in 1987. RC4 (which stands for Rivest Cipher 4) has found its way into several encryption schemes that are widely used today, for e.g. WPA, TLS and WEP. This stream cipher uses a permutation and a key-scheduling algorithm and avoids the use of Linear Feedback Shift Registers. However, recently several vulnerabilities have been found rendering it insecure. In 2015, after speculation that these vulnerabilities could break the TLS scheme[5], The International Engineering Task Force (IETF) prohibited its usage.

Salsa20 | ChaCha
Both these stream ciphers were developed by Daniel Bernstein in 2005. Both ciphers are built on a pseudorandom function based on add-rotate-XOR (ARX) operations — 32-bit addition, bitwise addition (XOR) and rotation operations. The core function maps a 256-bit key, a 64-bit nonce and a 64-bit counter to a 512-bit block of the key stream. To read Daniel Bernstein’s paper introducing these, refer to The Salsa20 family of stream ciphers. To learn about its core function, read The Salsa20 core.

Other ciphers include those submitted to the eSTREAM project like SOSEMANUK and MICKEY. Many stream ciphers are also built around a LSFR (Linear Feedback Shift Register)[6]

[5] Read about it here That earth-shattering NSA crypto-cracking: Have spooks smashed RC4?

[6] Read about it on the Wikipedia article on LSFRs

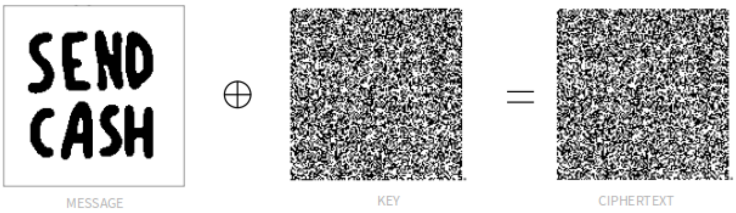
Cryptanalysis

We have already shown perfect secrecy for OTP as long as it meets certain criteria. But what if the user's implementation doesn't meet the criteria? In what ways is the system compromised ?

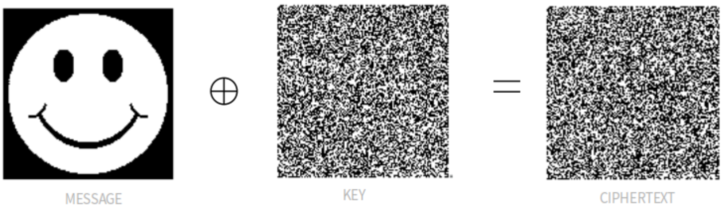
Two Time Pad Attack

Suppose a user does not know that a OTP key should never be reused. After all, they've taken much time to generate a good random key, why throw it away after using only once? There were quite a few such cases throughout history wherein this simple mistake comprised the user's OTP implementation. Refer to *Further Reading* to read about some famous historical two time pads. To emphasize why a OTP key should never be reused, a "proof" through pictures is provided below.[7]

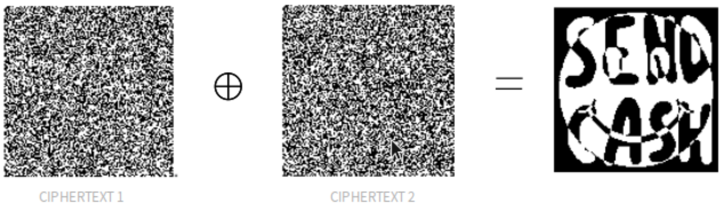
Our user, say Charlie, initially wants to ask his friend Dave for money, but he doesn't want anyone else to know. He generates a strong random key the same size as the plain text. Now he encrypts the message using OTP as follows -



Dave, being a good friend, sends the money over. Charlie replies with a confirmation of receiving the money. But he makes the terrible mistake of using the same key that was used for the first message.



Now Eve is spying on this conversation and knows of Charlie's fatal mistake. She somehow gets both the ciphertexts and is well versed in properties of XOR. She knows that if $c_1 = k \oplus m_1$ and $c_2 = k \oplus m_2$ then $c_1 \oplus c_2 = m_1 \oplus m_2$. She attempts this on the ciphertexts sent from Charlie to Dave.



As shown, Charlie's 'perfectly secure' system is comprised because of an error in implementation.

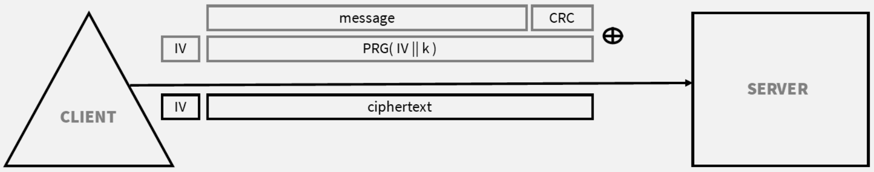
This should reinforce the fact **OTP KEYS SHOULD NEVER BE REUSED.**

[7] This example is heavily borrowed from A visual demonstration of the perils of key reuse in a one-time pad

REAL WORLD TWO-TIME PAD ATTACK

A real world example of a TTPA is in 802.11b WEP. From a cryptography standpoint, WEP is a disaster. It should never be used as a encryption scheme. One of the flaws in WEP is the short IV which allows for a Two Time Pad Attack.

Suppose a client and a server have an established connection and a shared key. The client sends the packet encrypted via 802.11b WEP (which uses a RC4 stream cipher.)



The IV is only 24 bits long while the key is 104 bits. The IV is randomly generated and pre-appended to the secret key 'k'. This is fed into the PRG and XORed with the message and CRC (a checksum for validating the message).

So after 2^{24} frames, ($\approx 16 \times 10^6$ frames), the IV cycles and we have a two time pad. 2^{24} is not that many frames in a busy network and consequently WEP is broken without much effort. An attack by Fluhrer, Mantin and Shamir showed that given upto 10^6 frames, WEP's secret key could be retrieved.[a] Both attacks completely cripple the WEP encryption scheme and the reader should note **never to use WEP.**

[a] Fluhrer S., Mantin I., Shamir A. (2001) Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay S., Youssef A.M. (eds) Selected Areas in Cryptography. SAC 2001. Lecture Notes in Computer Science, vol 2259. Springer, Berlin, Heidelberg

Further Reading

Semantic Security

For information regarding Semantic Security, use this crypto.stackexcange link - Definition and meaning of "semantic security"
The Wikipedia article on Semantic Security gives a detailed explanation.

Two Time Pad Attacks

For more information on Two Time Pad Attack, study about **Project Venona** - a counterinteliigence program by the US during WWII that was able to crack over 300,000 Russian encrypted messages because of reused keys. This historynet.com article 'The Venona Project' gives historical context while the Wikipedia article shows the attack.

A more recent example of Two Time Pad Attacks is the MS-PPTP authentication protocol. Read all about it here Cryptanalysis of Microsoft's PPTP Authentication

Information Theoretic Security

To understand more about this concept read the chapter "Information Theory" in Introduction to Mathematical Cryptography by Hoffstein, Pipher and Silvermann.
To read Shannon's original paper where he proved perfect secrecy, use Communication Theory of Secrecy Systems.

Salsa20 and RC4

To learn more about the working of these ciphers use these links - RC4, Salsa20