A Secure Authentication System- Using Enhanced One Time Pad Technique

Raman Kumar[1], Roma Jindal [2], Abhinav Gupta[3], Sagar Bhalla[4] and Harshit Arora [5]

[1,2,3,4,5] Department of Computer Science and Engineering,

[1,2,3,4,5] D A V Institute of Engineering and Technology, Jalandhar, Punjab, India.

Summary

With the upcoming technologies available for hacking, there is a need to provide users with a secure environment that protect their resources against unauthorized access by enforcing control mechanisms. To counteract the increasing threat, enhanced one time pad technique has been introduced. It generally encapsulates the enhanced one time pad based protocol and provides client a completely unique and secured authentication tool to work on. This paper however proposes a hypothesis regarding the use of enhanced one time pad based protocol and is a comprehensive study on the subject of using enhanced one time pad technique. This forms the basis for a secure communication between the communicating entities. Several password authentication protocols have been introduced each claiming to withstand to the several attacks, including replay, password file compromise, denial of service, etc. We introduced a new protocol through which the people can communicate to each other securely using one time pads. In this technique, very small one time pad is needed for both the sender and receiver and then they can have a very secure communication between them, so the server which is providing the communication between sender and receiver will also do not have any knowledge about the way to decrypt the text. This protocol is providing the secure communication by not trusting anyone except the sender and receiver which are the only one having the copy of small one time pad file. Therefore, the proposed scheme is secure and efficient against notorious conspiracy attacks.

Key words

One Time Pad (OTP), Random, Attacks Security Threats, Cryptographic Techniques and Information Security.

1. Introduction

Authentication plays an important role in protecting resources against unauthorized use. Since the advent of the computers, man has been looking for various possible means to make communication among entities secure via various authentication processes which range from simple password based authentication system to costly and computation intensive biometric authentication systems. Passwords have proved to be very handy. They are not just some key but also serve several purposes. They authenticate us to a machine to prove our identity a secret key that only we should know. The username is used to identify us and the password validates us. But with time the various weaknesses associated with a password have come to surface. It is always possible for people other than the authenticated user to posses its knowledge at the same time. Password thefts can and do happen on a regular basis, so there is a need to protect them. Rather than using some random set of alphabets and special characters as the passwords we need something new and something unconventional to ensure safety. At the same time we need to make sure that it is easy to be remembered by you as well as difficult enough to be hacked by someone else.

One time pad or OTP, also called vernam-cipher or the perfect cipher, is a crypto algorithm where plaintext is combined with a random key. A one-time pad is a cryptosystem invented by vernam[1]. It's a very simple system and is unbreakable if used correctly. To use a one-time pad, you need two copies of the "pad" which is a block of random data equal in length to the message you wish to encode. The word "random" is used in its most literal possible sense here. If the data on the pad is not truly random, the security of the pad is reduced, potentially to near zero.

One time pads are used in pairs. The more copies of a given pad, the greater the likelihood is that one may be captured, in which case the system is completely broken. One copy of the pad is kept by each user, and pads must be exchanged via a secure channel e.g.: face to face on floppy disks. The pad is used by XORing every bit of the pad with every bit of the original message. Once the message is encoded with the pad, the pad is destroyed and the encoded message is sent. On the recipient's side, the encoded message is XORed with the duplicate copy of the pad and the plaintext message is generated.

On random number generation, but for now just note that the one-time pad requires a truly random sequence of characters. If instead, one used a random number generator to create the sequence of pad characters, such a generator might depend on a single 32-bit integer seed for its starting value. Then there would be only 2^{32} different possible pad sequences and a computer could quickly search through all of them. Thus if a random number generator is used, it needs to have at least 128 bits of seed, and the seed must not be derived solely from something like the current date and time.

Manuscript received February 5, 2011
Manuscript revised February 20, 2011

One-time pad encipherment can be denoted as:

$C_i = E(P_i, K_i)$

where E is the enciphering operation, P_i is the i-th character of the plaintext, K_i is the i-th byte of the key used for this particular message, and C_i is the i-th character of the resulting ciphertext. Both the key stream K and the enciphering operation E are secret. The key for each individual message is the starting location in the entire random key stream used for this encipherment. For efficiency, it is good practice to start each message near the position following the key byte used for the last character of the previous message. This eliminates the need to keep track of which portions have been used, and removes the danger that a message will be longer than any of the remaining segments.

Although the term byte has been used, and will continue to be used for each unit of the key stream, it is not necessary that each unit of the key be 8 bits long. In fact, a key with larger units gives protection against a known-plaintext attack, since then there may be several values of the key which produce the given ciphertext byte from the known plaintext byte. Similarly, the plaintext could be enciphered in units other than 8-bit bytes. For present purposes, however, encipherment based on bytes is sufficiently general.

In principle, the encipherment E can be achieved by simple look-up in a $256 \cdot 2^k$-byte table, where k is the size of each key unit. Each column of the table, corresponding to one value of K_i is a permutation of the values 0 to 255. This makes it possible for the receiver to decipher the message uniquely. The cipher will be most secure if each of these 2^k permutations is statistically independent of the others and of the identity permutation, and if the enciphering operation is implemented so that the waveforms of the bits of C_i give no clue as to the values of P_i and K_i.

The reason that no portion of the key stream may be reused is that an opponent can detect reuse by the simple Index of Coincidence statistical test. Once the opponent finds two or more portions of messages enciphered with the same part of the key, it becomes possible to decipher those portions.

2. One Time Pad

The secure communication between two persons can only be ensured secure if the key or the password used for encryption is available to only the sender and the receiver. But once this key is compromised then the communication will be no longer secure, but if the key size is equal to the size of the plain text and with the consideration that these keys are only available to sender and receiver then the secure communication is looking very much in sight, which is called as the one time pad. The one time pad cryptography technique is old, and was used during the First World War by Gilbert Vernam (as shown in Figure - 1).

A one time pad is the only currently known unconditionally secure encryption system. Other encryption systems are cryptographically secure which means that they have a cost associated with breaking, this cost will be very high, but it would theoretically be possible to break if enough compute time could be gathered.

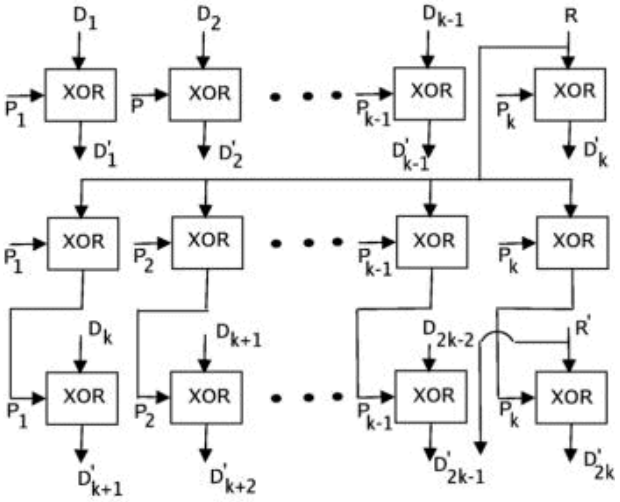

Figure - 1: Cryptographic techniques for one time pad

OTPs are provably unconditionally secure; this means that you can not break them with any amount of compute time. It is mathematically impossible to break an OTP. This is due to convenience, in order to use an OTP you have to trade pads with each person you wish to communicate with. The pad should be a truly random number, so you would ideally want a radioactive decay card or something to generate the pad. The system will critically depend on the true randomness of the pad, and of course on keeping the pad known only to you and the recipient.

This is where the inconvenience comes in, you must somehow trade pads with your intended recipient securely, this means meeting in person or using a trusted courier

2.1 How does it work

Basically you have your random OTP, which both you and your intended recipient have. You have a message M, and you compute the ciphertext C by XORing the message with the OTP:

C = M xor OTP

You send the ciphertext to your recipient, the recipient knowing the OTP also can recover the message by computing the reverse, XORing the ciphertext C with the OTP:

M = C xor OTP

You must never re-use the OTP, other wise it wouldn't be a "One-Time" pad anymore, and it would loose it's unbreakable properties as information would start to be leaked.

2.2 OTP in different languages

OTP in 1 line of perl:

vec($_,0,1);open(P,shift);read(P,$p,length),print $_^$p while<>

OTP in 1 line of C :

main(i,c)int*c;{for(c=fopen(c[1],"r");i=~getchar();putchar (getc(c)^~i));}

2.3 Using the OTP

You must create and exchange a OTP with your recipient. We call the OTP file "pad" here, the message to exchange "msg", and the encrypted ciphertext "cipher".

To encrypt, do (same usage for C or perl versions):

% otp pad < msg > cipher

Now you can send "cipher" to your recipient in the clear, just email will do fine as you will be the only two people who will ever be able to decipher the message.

To decrypt your recipient does:

% otp pad < cipher > msg

to recover the message.

This technique is the only known provably secure encryption scheme in the information-theoretic sense of Shannon [3]. It is based on the concept of using a long stream of random letters (or bits) to modify the plaintext message one letter (bit) at a time. Both the sender and the receiver must have a copy of the same pad and both work through the message one letter at a time. Suppose that the sender has a sequence of bits text(i) that are to be transmitted. The sender and the receiver exchange, through some secure mechanism, a pad of random bits pad(i) so that both sender and receiver have the same pad. To transmit a message to the receiver, the sender encrypts the plaintext text(i) obtaining the encrypted message message(i) as given in the equation (1).

message(i) = pad(i) XOR text(i)……. (1)

and transmits the encrypted message to the receiver. The receiver can recover the plaintext byre-applying the pad and obtains the plaintext as given in equation (2).

text(i) = message(i) XOR pad(i)…….(2)

one time pads have a number of advantages over traditional cryptography systems. They are extremely computationally efficient both in terms of encryption and decryption. They are also unbreakable under the assumption that

1. The pad is truly random
2. The pad is never reused, and
3. The pad is secret to the two parties.

It is also interesting to note that breaking part of the message does not provide any advantage in term of breaking the rest of the message. There is no decrypting key to guess.

3. Anomalies in One Time Pads

Despite Shannon's proof of its security, the one-time pad has serious drawbacks in practice [3]:

- Careful treatment to make sure that it continues to remain secret from any adversary, and is disposed of correctly preventing any reuse in whole or part — hence "one time".
- It requires perfectly random one-time pads
- Secure generation and exchange of the one-time pad material, which must be at least as long as the message.

The theoretical perfect security of the one-time-pad applies only in a theoretically perfect setting; no real-world implementation of any cryptosystem can provide perfect security because practical considerations introduce potential vulnerabilities. These practical considerations of security and convenience have meant that the one-time-pad is, in practice, little-used. Implementation difficulties have led to one-time pad systems being broken, and are so serious that they have prevented the one-time pad from being adopted as a widespread tool in information security. One-time pads solve few current practical problems in cryptography. High quality ciphers are widely available and their security is not considered a major worry at present. Such ciphers are almost always easier to employ than one-time pads; the amount of key material which must be properly generated and securely distributed is far

smaller, and public key cryptography overcomes this problem [6,14].

The primary disadvantage with using a one time only pad is that the pad must be as large as the message to be transmitted. So that signifies that we need a mechanism to transfer the keys with sized same as size of the data securely between sender and receiver in order to have a secure communication. But if we would have that mechanism to transfer the keys securely, we would have transferred the data itself. Then one of the solutions suggested by Frank Rubin [9], through which moderate amount of random keys can be used to generate life time supply of keys for one time pads. Thus with the combination of the true one time pads and using them to generate the pseudo random numbers, using pseudo random number generators can be used for making the one time pads practical to implement. In this kind of arrangement we need not transfer the same size as the plaintext, instead once secure transfer of several number of keys can lead to a very lengthy communication. The similar kind of methodology has been suggested by Michael Jenkin and Patrick Dymond [12] for making one time pad applicable for the light weight devices in order to avoid the heavy encryption computation, which will lead to widespread use of one time pad technology. One time pads are helpful in reducing computational overheads which is required in various encryption techniques as one time pad significantly reduce encryption overhead, offering unparalleled security to even the slowest of devices. But for making it possible for the light weight application we can not think of a seed of very high length, because of the computation limitation of the light weight devices and the limitation of the extendible memory of these devices. So we have to think of a mechanism, in which we can increase the complexity for the cryptanalyst without increasing the seed size.

4. Enhanced Cryptographic Techniques

For the above problem we have to reach a solution in which in the lowest of computation we can reach maximum of complexity. That means without taking a seed of very high length we should be able to provide the complexity like the high length seed, only then we can think of providing the secure communication to variety of the devices to make this technique free from the devices constraint, then only we can be able to reach to a security protocol which will be able to transfer the data securely from sender to receiver irrespective of the devices they use. Here instead of taking a single large true one time pad key as a seed, we have taken multiple small sized true one time pad key which is identically stored with both sender and receiver.

We are proposing the use of three random keys together to be taken from the one time pad file which is there with both sender and the receiver and that one time pad file can be generated from any natural source random number generators considered to be a true random numbers then they cannot be predicted. The above said file is only available to sender and receiver, not with any one else, so the service provider through which both are going to communicate will only be responsible for providing the reliable communication. We will use all the three keys one by one but together to generate the entire key stream through which we will encrypt the plaintext by one time pad technique. And once they were used, they will be removed from the one time pad file from both the sender and receiver side and will not be reused again. On the finishing of the all the keys from the file, both sender and receiver should get themselves issued a new file from the server, the file which is same for both sender and receiver by some physical means like pen drive, compact disk, etc. Once the file reaches to the both the users they can again start communicating again securely to a large number of sessions (as shown in Figure - 2).

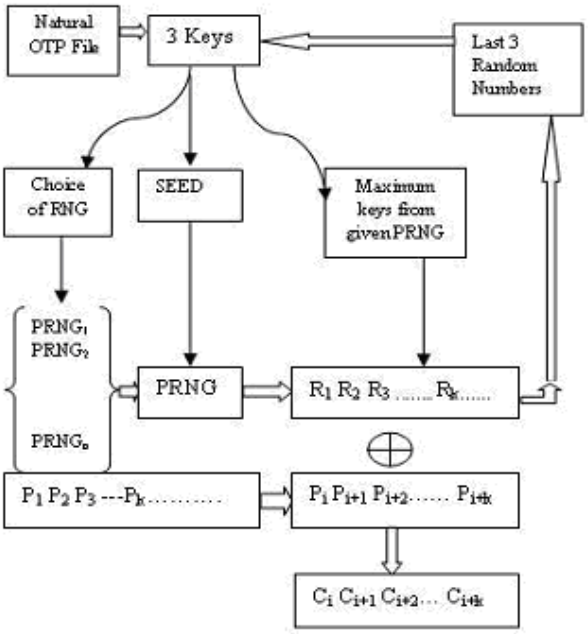

Figure - 2: Enhanced cryptographic techniques for one time pad

Where:-

$PRNG_i$ (i=1 to n) are the choices of pseudo random number generators available for use.

R_i (i=1 to k) are the random numbers generated by particular PRNG.

P_i is the plain text individual element which is to be coded with its corresponding random number.
C_i is the cipher text individual element which has been coded from plain text element and corresponding random number.
In the figure 1 we can see the overview of the new protocol for making the pads a practical solution for communication. Natural OTP (one time pad) file is available with only sender and receiver and from that both sender and receiver will take out three keys and will be used as given below.
The following is the description that how those random keys will be used:

1. It will act as a seed.
2. It will be used to tell the choice of random number generator to be used.
3. It will be used to tell the number of key to be taken from the PRNG (pseudo random number generators) in action.

As in one time pads discussed above the decryption and encryption method is just the same, so if sender sends the data in form of the cipher text by XORing the plain text with the random numbers generated above the receiver also have to do the same for getting the plain text back from the cipher text. Now we will discuss the significance of each of the above used random keys and how they will strengthen the security of the key stream.

- As we know that each PRNG need some seed to produce the key stream which is totally based on the seed. If we do not change the seed then the same sequence will be produced, which can be predicted very easily, that is why we are using a true random seed to randomize the entire key stream.
- If we use a single PRNG then it will be easier to guess the key stream from the various outputs we are getting as only one PRNG is involved then the above possibility cannot be ignored. Therefore we have taken various PRNG's together in a choice format, so at every choice given by a true random number a PRNG at that particular choice will be used for that particular session for which we have taken those three random numbers. The choices of various PRNG's to be used are left to the service provider's choice to analyze its compatibility with various PRNG's and also to the device it is going to provide its service.
- Now as we know that each PRNG has a period after which it starting repeating sequence then in order to avoid that and also at a larger length the prediction of PRNG sequence prediction becomes easier so we are bound to break the PRNG sequence in between and starting the process of reseeding from the last three random numbers taken from the sequence itself, it will also help in stopping the user to know about the future data if get hold of some of the data in between somehow.

In the above methodology we can increase the complexity to a very high level as now crypt analyzer has to get the combination of three keys together and also in right sequence to break the entire code. That's not all now crypt analyzer has to work on list of various PRNG and along with their seed to get the entire key stream to decrypt the whole text. As in every session new keys will be used all together from the Natural OTP file then, the above methodology also provide entire new key stream for each session.

5. Encrypting with Enhanced Cryptographic Techniques

Encryption basically is a simple XOR algorithm; the implementation correspondingly is not very complex. I already included some error handling code in the method XorFileWithPad. However, the parameters are more interesting, especially nPadStartPos. With this, I specify where in the pad to start reading bytes for encryption. This is of special interest if the pad is large in relation to the initial file and the pad is to be used for several encryptions. Correspondingly, the function returns also the last byte used for encryption. At the next call of the function, this is where you should continue (+1, obviously).

```
public long XorFileWithPad(string strInputFile, string
strDestinationFile, string strPad, long nPadStartPos)
    {
     if (! File.Exists(strPad))
     {
      throw new ArgumentException("OTP file does not
  exist");
     }
       if (! File.Exists(strInputFile))
     {
      throw new ArgumentException("Input file does not
  exist");
     }
       if (File.Exists(strDestinationFile))
     {
      throw new ArgumentException("Destination file
  must not exist");
     }
     FileInfo infoPad = new FileInfo(strPad);
     FileInfo infoInputFile = new FileInfo(strInputFile);
     long nInputFileLength = infoInputFile.Length;
     long nPadLength = infoPad.Length;
```

```
    if ((nPadLength - nPadStartPos) <
nInputFileLength)
    {
      throw new ArgumentException("Pad is not long
enough to Xor file!");
    }
    FileStream fsOutput =
File.Create(strDestinationFile);
    FileStream fsPad = File.OpenRead(strPad);
    FileStream fsInput = File.OpenRead(strInputFile);
    int nBufferSize = 1000, nInputSize, nPadSize, nXor;
    byte[] abInput = new Byte[nBufferSize];
    byte[] abPad = new Byte[nBufferSize];
    byte[] abOutput = new Byte[nBufferSize];
    while (0 != (nInputSize = fsInput.Read(abInput, 0,
nBufferSize)))
    {
      nPadSize = fsPad.Read(abPad, 0, nBufferSize);
      for (nXor = 0; nXor < nInputSize; nXor++)
        abOutput[nXor] = Convert.ToByte(abInput[nXor]
^ abPad[nXor]);
      fsOutput.Write(abOutput, 0, nInputSize);
    }
    fsOutput.Close();
    fsInput.Close();
    fsPad.Close();
// this method returns the last byte used of the onetime
pad
// never re-use *any* portion of a onetime pad!!
    return (nPadStartPos + nInputFileLength);
  }
```

The encryption is a simple XOR, the dominant part of the function deals with reading and writing files. Here we also wrote a command line wrapper application:

padxor.exe inputfile padfile padstart outputfile

The source code can be found in padxor.cs (as with padgen.cs, user friendliness needs to be improved upon urgently). Now we can get going encrypting and decrypting. Here an example sequence of commands:

padgen mypad.bin 200000
padxor otp.cs mypad.bin 0 otp.cs.enc.txt
padxor otp.cs.enc.txt mypad.bin 0 otp.cs.dec.txt

The file otp.cs.enc.txt is sent to the recipient who has received mypad.bin via another (non-electronic) means of communication beforehand. He can then decrypt otp.cs.enc.txt using the pad, as shown here in otp.cs.dec.txt.

5.1 One Step Beyond

Even though our message can be now sent in an absolutely unbreakable manner, there is a catch - it cannot be broken. This may sound paradox, but let me explain: encryption using onetime pads results in the message being close to random in its statistics and thus barely compressible - this is a quite strong indication for the use of a onetime pad. You might therefore be asked to decrypt the message as a 'matter of courtesy'.

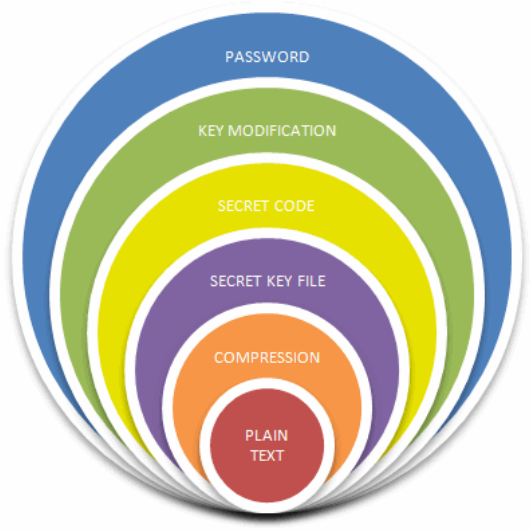

Figure - 3: Enhanced cryptographic techniques for one time pad

To circumvent this 'courteous' request, there is another trick at hand: we encrypt the original message to Ciphertext1. This Ciphertext1 we now encrypt yet again using Vernam, this time however using a completely harmless dummy message. This then yields Ciphertext2, which we send to the recipient, deleting the onetime pad immediately. Secure enhanced cryptographic techniques for one time pad may compose of plain text, compression, secret key file, secret code, key modification and password. Figure – 3 illustrates the functioning of enhanced cryptographic techniques for one time pad.

6. Applications of enhanced one-time-pad

- Enhanced one-time pads are practical in situations where two parties in a secure environment must be able to depart from one another and communicate from two separate secure environments with perfect secrecy.
- The algorithm most commonly associated with quantum key distribution is the one-time pad.
- The enhanced one-time pad can be a part of an introduction to cryptography
- The enhanced one-time-pad can be used in super encryption.

- The enhanced one-time-pad is one of the most practical methods of encryption where one or both parties must do all work by hand, without the aid of a computer; this made it important in the pre-computer era, and it could conceivably still be useful in situations where possession of a computer is illegal or incriminating or where trustworthy computers are not available.
- The enhanced one-time-pad is the only cryptosystem with theoretically perfect secrecy.

7. Conclusion

In this paper we have proposed a new methodology through which one time pads can be practically used for communication for large range of devices. Through above methodology we will be able to decrease the key size without compromising the complexity which has come down by reduction of the key size. By the above proposed hypothesis we are making one time pads a practical solution between two entities which wants to communicate securely to each other, and as the one time pad file is not even with the server through which they are communicating, we are trying to remove any vulnerable point in between. So even the server got compromised the users need not have to worry about the security of there communication. Therefore, the proposed scheme is secure and efficient against notorious conspiracy attacks.

Acknowledgments

I (Raman Kumar) deeply indebted to my beloved master, supervisors, my parents and my research laboratory whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this paper for journal. The authors also wish to thank many anonymous referees for their suggestions to improve this paper.

References

[1] Andrew S. Tanenbaum and Maarten Van Steen, “Distributed Systems”, Pearson Education, 2008.
[2] C. M. Chen, and W. C. Ku, “Stolen-verifier Attack on two New Strong-password Authentication Protocol,” IEICE Transactions on Communications, Vol. E85-B, No. 11, pp. 2519—2521, November 2002.
[3] C.E. Shannon, “Communication theory of secrecy systems”, Bell System tech. J., 28:657-715, 1949.
[4] Cristian Darie, Bogdan Brinzarea, Filip Chereches-Tosa, and Mihai Bucica, AJAX and PHP: Building Responsive Web Applications, Paperback, March 1, 2006.
[5] D. Kahn, “The Codebreakers”, MacMillan, New York, 1967
[6] Erskine, Ralph, "Enigma's Security: What the Germans Really Knew", in "Action this Day", edited by Ralph Erskine and Michael Smith, pp 370–386, 2001.
[7] F. Belli, “A Holistic view for Modeling and Testing User Interactions using Finite-State Techniques,” Proceedings of the 1st South-East European Workshop on Formal Methods, SEEFM'03,Thessaloniki, Greece, November 2003.
[8] F. Belli, “Finite-State Testing and Analysis of Graphical User Interfaces”, Proc. 12th ISSRE, pp. 34-43, 2001.
[9] F. Belli, K.-E. Grosspietsch, “Specification of Fault-Tolerant System Issues by Predicate/Transition Nets and Regular Expressions – Approach and Case Study”, IEEE Trans. On Softw. Eng. 17/6, pp. 513-526, 1991.
[10] Frank Rubin. One-Time Pad Cryptography, Published in Cryptologia, 20(4): 359 – 364, 1997.
[11] Id Quantique White Paper, Version 2.0, 2004
[12] Michael E.Gruen, “A secure low-power approach for providing mobile encryption”, Proceedings of the Eleventh annual CCSC northeastern conference, 2006
[13] Michael Jenkin and Patrick Dymond, “Secure communication between lightweight computing devices over the internet”, Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002.
[14] Robert Wallace and H. Keith Melton, with Henry R. Schlesinger, Spycraft: The Secret History of the CIA's Spytechs, from Communism to al-Qaeda, New York, Dutton, 2008, ISBN 0-525-94980-1
[15] Sandirigama, A. Shimizu, and M. T. Noda, “Simple and secure password authentication protocol (SAS),” IEICE Transactions on Communications, vol.E83-B, pp.1363-1365, June 2000.
[16] Security Guidelines: Prevention and Response and Hacker Attacks, Digital Edition, June 1, 2001.
[17] Vijay K. Bhargava, H. Vincent Poor, Vahid Tarokh, and Seokho Yoon, “Communications, Information and Network Security”, Hardcover, December 31, 2002.
[18] William Stallings, “Cryptography and network security design and principles”, Pearson Education, 2008.
[19] Y, Xiao, “Security in Distributed and Networking Systems” Computer and Network Security, Hardcover, September 30, 2007.

Mr. Raman Kumar working as an Assistant Professor with the Department of Computer Science and Engineering, D A V Institute of Engineering and Technology, Jalandhar. Before joining D A V Institute of Engineering and Technology, Jalandhar, He did his Bachelor of Technology *with honours* in Computer Science and Engineering from Guru Nanak Dev University; Amritsar *(A 5 Star NAAC University).* He did his Master of Technology *with honours* in Computer Science and Engineering from Guru Nanak Dev University; Amritsar *(A 5 Star NAAC University).* His major area of research is Cryptography, Security Engineering and Information security. He has various publications in National as well as International Conferences and Journals on his research areas.