Advanced Port Knocking Authentication Scheme with QRC using AES

Vikas Srivastava[1], Alok Kumar Keshri[2], Abhishek Dutta Roy[3], Vijay Kumar Chaurasiya[4], Rahul Gupta[5]

Department of Cyber Law and Information Security
Indian Institute of Information Technology, Allahabad, India
[1]enggvikassrivastava@gmail.com, [2]alokkeshri21@gmail.com
[3]abhishekdutta1987@gmail.com, [4]vijayk@iiita.ac.in, [5]rahulgupta@iiita.ac.in

***Abstract*— Port Knocking is an important concept to secure services provided by the servers. By a predefined port knocking sequence server identify whether the request is a legitimate request for a service. This paper presents an improved authentication scheme over the existing port knocking methods. The existing port knocking methods are prone to reasonable attacks and vulnerabilities. The paper addresses those vulnerabilities, and accordingly provides mechanism to circumvent on the port knocking mechanism. In a client-server communication, request for services from the clients is done by providing them connection to a specific port on the server. For security concerns, all the ports on the server are initially closed and no connection is possible. Port knocking permits a user to request for a port to open for network services. This request takes the form of a sequence of authentication packets across closed ports on the server. Many port knocking schemes have been proposed earlier but all of them suffer from the problems like sequence replay attack, man in the middle attack, use of spoofed packets in knocking sequence and out of order delivery of packets. The proposed algorithm addresses all the above mentioned issues by implementing a secure knock sequence with AES encryption scheme, which cannot be detected or disturbed by the sniffing and use of spoofed packets. The algorithm also addresses the problem of out of order delivery of knock sequence packets as the knock sequence is determined when all the packets are received at the server end.**

Keywords -Port Knocking; One Time Password (OTP)[1][4]; Quadratic Residue Cipher(QRC)[2][4]; Advanced Encryption Standards

I. Introduction

The Internet can be seen as a huge network of nodes connected together to provide different services. The question that comes is how to access the servers in the network. This issue has been addressed by following approaches:

- One can be by using a firewall, which can control the traffic based on IP addresses.
- Another can be by using more customized devices like Intrusion Detection and Prevention Systems.

But still, even after putting such efforts towards securing the network, cases of network-attacks are often reported.

In general, the first step of any attacker is information gathering, in which the attacker tries to find the complete details of the victim system or network like the services running, ports opened, version number of some particular softwares etc. In the second step, attackers try to find out both the existing and zero-day vulnerabilities in the version of services, and may exploit those vulnerabilities to cause breach of confidentiality, integrity or availability issues.

The first line of defence against any attack is system's firewall. The firewall is used to limit the resources of any network connections. A firewall works on predefined set of rules according to which it accepts or rejects any packet. These rules may be based upon the IP addresses or some other characteristics. Since very little information is revealed by the source address of a packet, the attacker may easily disguise the origin of the packet by modifying its contents or inserting its own packets, thereby making it bypass the firewall for potential (ab) use.[5]

Port Knocking [6] is appropriate for users who require access to servers that are not publicly available. Port knocking refers to a method of communication between two computers, usually a client and a server, in which the information is encoded and possibly encrypted in to a sequence of port numbers. This sequence is termed as knock sequence. The server can keep all its ports closed but open it on demand if users have authenticated themselves by providing a specific knock sequence (a sort of password). Initially all the ports on the server are closed for public communication and the server is monitoring all the attempts to connect to the services. The clients initiate a connection by sending SYN packets to some specific ports on the server which are specified in the knock sequence. During the knocking process by the client, server offers no response and it just monitors the knocks silently on the specified ports. When the server detects a valid knock sequence, it triggers a server side process and opens the port for communication with the client. This is a form of the IP communication over closed ports. The definition of valid knock sequence and the server side process is completely user dependent and can result in modification of firewall rules and other administrative system events. This is an authentication mechanism, in which closed ports on the server system are knocked in a predefined sequence to provide services to the legitimate user after successful authentication. It provides an additional layer of security to the server by having additional advantage of stealth also. In other words, port knocking is a mechanism which is used to hide the services running on a hardened server which have users that require continual access to services and data from remote locations and that are not running any common public services. [6][7].

Some advantages of using the port knocking technique are as follows:

- Almost impossible to determine whether port knocking is implemented on the server machine or not.
- Detection by sniffing is practically difficult.
- It is a firewall based method for user authentication for non common services.
- Establishes connections to the hosts with no open ports by the subversive use of closed ports.
- Benefits from access control provided by IDS and firewalls.

This paper describes some of the shortcomings existing in the standard port knocking mechanism including some new problems not yet addressed, along with a set of solutions. The algorithm and architecture proposed here deals with the existing problems by providing a cryptographically secure way to determine the knock sequence required to get authenticated to the server.

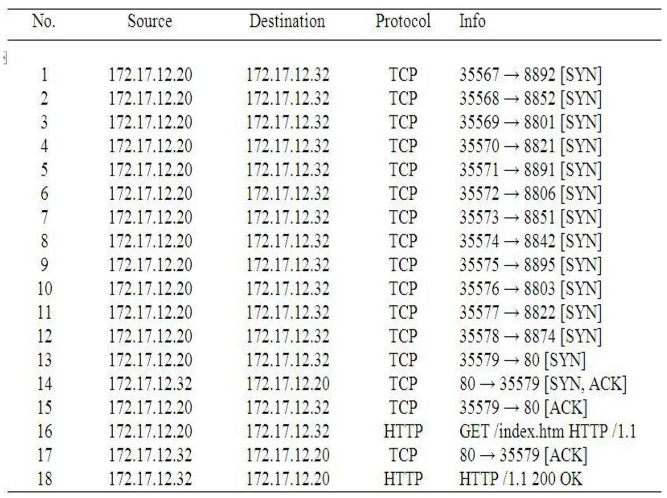
II. Problems In Existing Port Knocking Authentication Techniques

There are certain problems encountered with the existing model of port knocking mechanism. An attacker can discover the actual sequence of packets in port knocking process and launch attacks in some of the following ways:

- A sequence replay attack in which a particular set of packets can be sent to the victim again and again.
- Getting the sequence number by sniffing the packets.
- Observing the pattern of knock sequences in promiscuous mode and running port scans.
- Detection and interpretation of simultaneous knock sequences.
- Load caused on the network and individual systems.
- Packets are delivered and received out of order due to network latency and other factors, thereby exposing the knock sequence in most of the cases.
- Single packet authentication can lead to the disclosure of complete data if the packet is captured by the attacker in between the client and the server machine.
- Knock sequences being influenced by the use of spoofed packets [8].

In the above figure, we may see that the source IP address is same in all the cases and it can be clearly determined that a particular sequence of packets is send to the server machine to acquire some specific services. Anyone who is silently sniffing the network can detect and interpret the sequence of SYN packets and thereby simultaneous knock sequences. This can reveal the services running on the server as well as the particular sequence of SYN packets that are required to be sent to use the services. This also reveals the identity of the client which may lead to the sequence replay attack and Denial of Service attack on the client. Hence the port knocking mechanism used here is not secure.

This paper addresses this problem by proposing a new framework which will provide additional level of security in the existing models of port knocking. This framework will increase the level of complexity for attackers in discovering correct port knocking sequence by randomizing the source ip addresses and the port numbers with the help of Quadratic Residue Cipher (QRC). [2]



No.	Source	Destination	Protocol	Info
1	172.17.12.20	172.17.12.32	TCP	35567 → 8892 [SYN]
2	172.17.12.20	172.17.12.32	TCP	35568 → 8852 [SYN]
3	172.17.12.20	172.17.12.32	TCP	35569 → 8801 [SYN]
4	172.17.12.20	172.17.12.32	TCP	35570 → 8821 [SYN]
5	172.17.12.20	172.17.12.32	TCP	35571 → 8891 [SYN]
6	172.17.12.20	172.17.12.32	TCP	35572 → 8806 [SYN]
7	172.17.12.20	172.17.12.32	TCP	35573 → 8851 [SYN]
8	172.17.12.20	172.17.12.32	TCP	35574 → 8842 [SYN]
9	172.17.12.20	172.17.12.32	TCP	35575 → 8895 [SYN]
10	172.17.12.20	172.17.12.32	TCP	35576 → 8803 [SYN]
11	172.17.12.20	172.17.12.32	TCP	35577 → 8822 [SYN]
12	172.17.12.20	172.17.12.32	TCP	35578 → 8874 [SYN]
13	172.17.12.20	172.17.12.32	TCP	35579 → 80 [SYN]
14	172.17.12.32	172.17.12.20	TCP	80 → 35579 [SYN, ACK]
15	172.17.12.20	172.17.12.32	TCP	35579 → 80 [ACK]
16	172.17.12.20	172.17.12.32	HTTP	GET /index.htm HTTP /1.1
17	172.17.12.32	172.17.12.20	TCP	80 → 35579 [ACK]
18	172.17.12.32	172.17.12.20	HTTP	HTTP /1.1 200 OK

Figure 1: Captured Data showing standard port knock sequence

III. Proposed Solution For The Authentication Mechanism

There are many implementation of port knocking algorithms and many of them require the client to send a fixed predefined sequence of port knocks to the server machine. The paper is proposing solutions to improve port knocking algorithm problem. Consider a scenario where a server machine is using a port knocking algorithm. Here we are using the concept of One Time Password which will act here as the One Time Key for the complete AES encryption technique along with the Quadratic Residue Cipher (QRC) for spoofing the source IP-address multiple times to increase the complexity in predicting the sequence of IP-address and packets in case of the sniffing and other similar techniques. We are also using the pseudo random number generator (PRNG) to generate the random numbers in the real time which will be used in the QRC as well as key for the AES encryption.

The whole process is initiated by user by sending an SMS to the SMS Server connected to the PRNG requesting for the 256-bit One-Time Key which will act as the key for AES encryption and an 8-bit random number R which should be relatively prime to P and N. The numbers P and Q are prime numbers unique for each user and are already stored in the database server connected to the SMS server for each user. The SMS is sent through a dedicated channel which in out of band for general communication. The reply sent by the server is also send through the same dedicated channel. The only precursor for this process is that the user's mobile number should be registered beforehand with the SMS Server. The message send by the user should be of the form

Request OTP <Allotted User ID><Fixed User Pass-code><source IP-Address>

The OTP is generated with validity time period so that there is no duplicate OTP generated before the use of original OTP or the expiration of time period. This is done to prevent the duplicity of OTP for the same user ID to prevent denial of service attacks on the PRNG server and SMS server. The random number R is of 8-bit which will be used to spoof the IP-address every time the request packet is sent to the application server through any firewall. The reply SMS from the SMS Server should be of the form [9]

<Time Stamp><One Time Key><Random Number R>

Time stamp is used to verify authenticity of the OTP and will be required in the prevention of spoofing of the mobile numbers and DOS attacks. The obtained OTP is then passed to the client to work as the key for the AES which will be used to encrypt the data to be sent i.e. the knock sequence which is our authentication data in this case and R which will be used in the Quadratic Residue Cipher.

The SMS server also sends the same information i.e. 256-bit OTP and 8-bit R sent to the firewall server which will be used in further steps to authenticate the user to access a particular service. As shown in Fig. 3, the OTP and last 8 bits of the Client IP address will be used to encrypt the data using AES encryption technique and generate the knock sequence. Multiple packets are generated to send to the firewall server of the target to provide authentication with the help of Multiple Packet Authentication (MPA). Each time the last 8-bits of the Client IP Address sent is spoofed to generate a new IP address by XOR-ing it with R. This process continues until we have the required number of packets to be sent to get authenticated with the server.

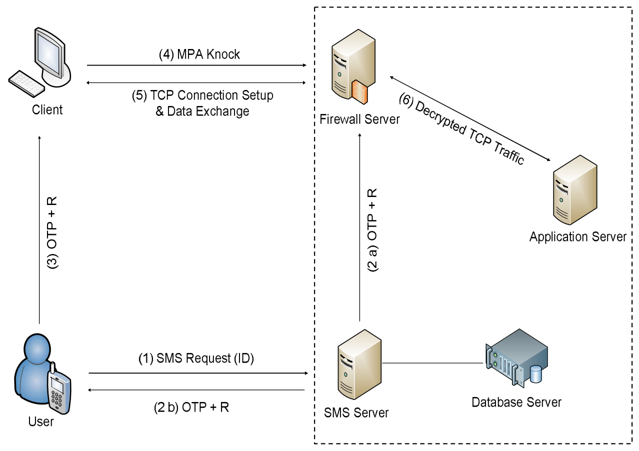


Figure 2: Port Knocking Architecture

The server is constantly monitoring the incoming packets silently. Since the server already has the values of P, Q, R and OTP key, it also generates the required knock sequence. The received packets are stored in a queue to be decrypted until it has received the required number of packets. Once server has the required number of packets, they are being decrypted with help of the One Time Key and AES cipher. If the obtained sequence is matched with the generated sequence at the server, access is granted to the client by opening the port and allowing the connection to be initiated; otherwise the request is solely denied at the firewall only as depicted in the fig. 4.

Flowchart of the proposed algorithm for client and server side is given in fig. 3 and 4.

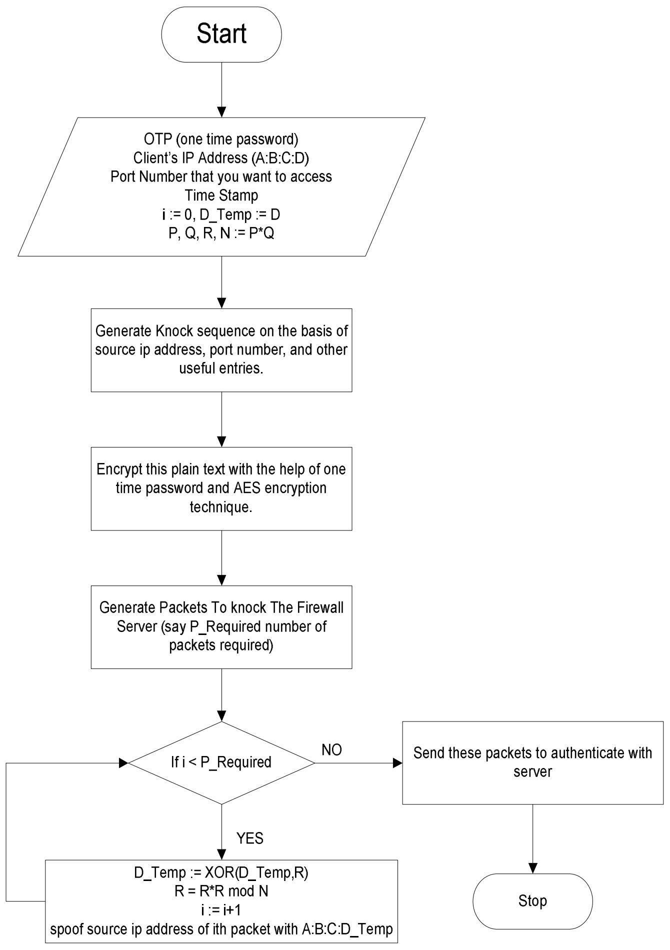

Start
OTP (one time password)
Client's IP Address (A:B:C:D)
Port Number that you want to access
Time Stamp
i := 0, D_Temp := D
P, Q, R, N := P*Q
Generate Knock sequence on the basis of source ip address, port number, and other useful entries.
Encrypt this plain text with the help of one time password and AES encryption technique.
Generate Packets To knock The Firewall Server (say P_Required number of packets required)
If i < P_Required
NO
Send these packets to authenticate with server
YES
D_Temp := XOR(D_Temp,R)
R = R*R mod N
i := i+1
spoof source ip address of ith packet with A:B:C:D_Temp
Stop

Figure 3: Flowchart of activity at the client side

In the proposed architecture, the client sends the request to generate the one time key and the random number R which is relatively prime to the SMS server which has an attached pseudo random number generator (PRNG). Assuming that the client's IP address is A.B.C.D, we generate the knock sequence based upon the source IP address D, random number R required for QRC, port number associated with the desired service and N which equals P*Q where P and Q are two prime numbers which are a unique combination for each of the registered user and is allotted and stored beforehand with the SMS server. This knock sequence is encrypted with the AES encryption technique and the one time key generated in the beginning acts as the key for the cipher in this case. Then packets are generated in order to send to the server to get authenticated. Assuming that the required number of packets are P_Required, we spoof the IP address of the source for each packet by the use of QRC by XOR-ing the last eight bits of IP address i.e. D with the random number R. For each subsequent packet, the eight bits are taken from the IP address of the preceding packet. This process is continued until all the packets are spoofed in this way and we have a different IP address for each packet. This is done in order to overcome the problem of sniffing of packets through which one can detect and interpret the sequence of packets and hence the successive knock sequences required to get access to the services on the server. The multiple packets sent in this way provide a multiple packet authentication knock sequence which prevents the disclosure of information against the single packet authentication in which if the packet is captured in between, all the relevant data and information may be disclosed. Even an automated tool used for packet sniffing over the network will just be able to record the packets but since the IP address would be different every time, it will be very difficult to detect or predict the sequence of knock sequences or even if any of the port knocking algorithms are implemented on the server.

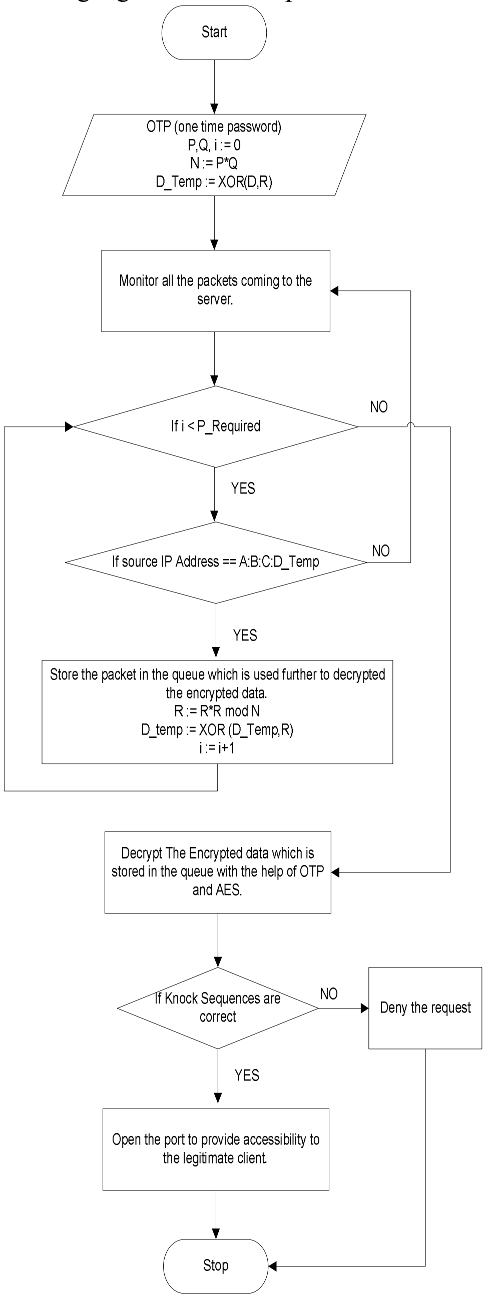

Start
OTP (one time password)
P,Q, i := 0
N := P*Q
D_Temp := XOR(D,R)
Monitor all the packets coming to the server.
If i < P_Required
NO
YES
If source IP Address == A:B:C:D_Temp
NO
YES
Store the packet in the queue which is used further to decrypted the encrypted data.
R := R*R mod N
D_temp := XOR (D_Temp,R)
i := i+1
Decrypt The Encrypted data which is stored in the queue with the help of OTP and AES.
If Knock Sequences are correct
NO
Deny the request
YES
Open the port to provide accessibility to the legitimate client.
Stop

Figure 4: Flowchart of activity at server side

On the server side, we already have the entries generated for the particular client by the SMS server i.e. the one time key and the random number R. Initially the server is monitoring all the incoming packets and stores all the incoming packets in the processing queue until all the packets from the client are received at the server end. This feature ensures that even if packets are delivered out of order due to network latency or certain other factors, there is no effect on the knock sequence and no disturbance is created in between because of unavailability of the packets at any instant of time. The monitoring process is continued until the source IP address gets being duplicated with the packets already in the processing queue. When the desired number of packets is achieved, they are sent into the decryption queue where they are stored until all the packets are decrypted to get the information. For each packet, the last eight bits of the IP address A.B.C.D i.e. D is obtained by XOR-ing it with the random number R which is in turn obtained as R*R mod N. This process continues all the packets are sent into the decryption queue where they are decrypted by the help of one time key of the user session for the AES encryption scheme. The result of the decryption process is the knock sequences sent by the client system. If the decrypted knock sequence matches the correct knock sequence as defined by the server, a server side process is triggered which alters the firewall rules and other administrative events at the server by opening the targeted port and grants the authentication to the client to gain access to the particular service for which the knocking sequence is applied for. On the other hand, if the knock sequence provided by the user does not matches the required sequence, the authentication request is complete denied without any notification to the user. This is done to maintain the anonymous presence of the port knocking algorithm on the server so that the determination of such type of service remains a challenge for the attacker. When the server triggers a process for granting access to the user, the firewall and IDS rules are temporarily modified for a particular period of time after which the whole system returns to its initial state and the ports are again closed for the access.

IV. ADVANTAGES OF THE PROPOSED ALGORITHM

Following are the advantages of the proposed framework over the existing port knocking authentication mechanisms:

- The detection and improvisation of the subsequent and consecutive knock sequences is protected by spoofing the IP address of the source client for each time a packet is received by the server. Even an automated sniffing tool running on the network will just be able to determine the packets and since each packet would be coming from a different source, as it would appear, correlation of data by the tools will not be an easy task.
- The out of order delivery problem of packets due to network latency and other problems are encountered by initially collecting all the packets from any source at the server and then applying the decryption algorithm so that the correct sequence of knocks may be obtained.
- The problem of single packet authentication is overcome by implementing the multiple packet authentication mechanism

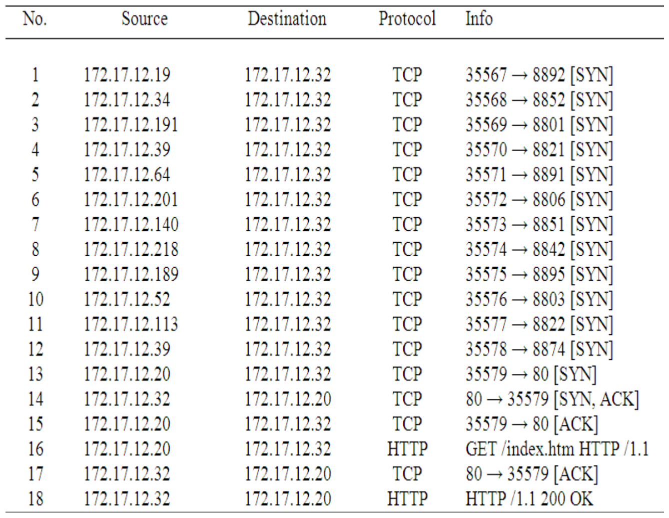
so that even if one of the packets is captured by any attacker, the complete information and data cannot be revealed.

- The other problems of replay attacks and man in the middle attack along with the denial of service attack is prevented by protecting the identity of the client system since the actual IP address is not send to the server machine repeatedly.

V. Result

In Figure 5, we assume that source IP-Address is 172.17.12.20 and the value of p = 11, q = 17. Firewall server will randomly pick the value of R (here R = 7), which is relatively prime to N (=p*q). After the generation of knock sequences on the basis of source IP-Address, the destination port number and other entries are encrypted using AES and OTP. Once all the packets are generated to knock the firewall server, the algorithm follows the flowchart depicted in Fig. 3. The source IP-Address has been spoofed of each packet that will be involved in multiple- packet knocking (MPA). If we XOR 20 and 7, we get 19. Therefore, the first source IP-Address is 172.17.12.19. The value of 7^2 mod (11*17) = 49 and XOR of 19 and 49 is 34. Therefore the second source IP-Address in the sequence is 172.17.12.34. This process is repeated and the following table is generated.

The above figure shows the flow of data from client to server using the proposed architecture. Comparing Figure 1 and Figure 5, we can see that the source IP-Address in the second one is varying, and not the same as shown in Figure 1. Even if the attacker captures the packets of the network, it is very difficult to analyze and identify the relationship between the observed packets. AES is used as the encryption algorithm since it is highly reliable and secured. Using One Time Password, we are generating a randomized key for each request, thereby strengthening the existing port knocking mechanism.



No.	Source	Destination	Protocol	Info
1	172.17.12.19	172.17.12.32	TCP	35567 → 8892 [SYN]
2	172.17.12.34	172.17.12.32	TCP	35568 → 8852 [SYN]
3	172.17.12.191	172.17.12.32	TCP	35569 → 8801 [SYN]
4	172.17.12.39	172.17.12.32	TCP	35570 → 8821 [SYN]
5	172.17.12.64	172.17.12.32	TCP	35571 → 8891 [SYN]
6	172.17.12.201	172.17.12.32	TCP	35572 → 8806 [SYN]
7	172.17.12.140	172.17.12.32	TCP	35573 → 8851 [SYN]
8	172.17.12.218	172.17.12.32	TCP	35574 → 8842 [SYN]
9	172.17.12.189	172.17.12.32	TCP	35575 → 8895 [SYN]
10	172.17.12.52	172.17.12.32	TCP	35576 → 8803 [SYN]
11	172.17.12.113	172.17.12.32	TCP	35577 → 8822 [SYN]
12	172.17.12.39	172.17.12.32	TCP	35578 → 8874 [SYN]
13	172.17.12.20	172.17.12.32	TCP	35579 → 80 [SYN]
14	172.17.12.32	172.17.12.20	TCP	80 → 35579 [SYN, ACK]
15	172.17.12.20	172.17.12.32	TCP	35579 → 80 [ACK]
16	172.17.12.20	172.17.12.32	HTTP	GET /index.htm HTTP /1.1
17	172.17.12.32	172.17.12.20	TCP	80 → 35579 [ACK]
18	172.17.12.32	172.17.12.20	HTTP	HTTP /1.1 200 OK

Figure 5: Data Flow where values are p=11, q=17, R=7 and source IP-Address=172.17.12.20

VI. Conclusion

It can be observed that the source IP-Address do not follow any particular pattern or sequence, as a result it will become difficult for the attacker to interpret the knock sequence, even if he/she tries to monitor the traffic at the server end.

References

[1] One Time Password http://en.wikipedia.org/wiki/One-time_ password.

[2] Quadratic Residue Cipher http://en.wikipedia.org/wiki/Quadratic _residue

[3] Advanced Encryption Standards http://en.wikipedia.org/wiki/ Advanced_Encryption_Standard

[4] W. Stallings, Cryptography and Network Security, 4th Edition, Pearson Education, 2005.

[5] Rennie deGraaf, John Aycock and Michael Jacobson, Jr., Department of Computer Science, University of Calgary, Alberta, Canada, "Improved Port Knocking with Strong Authentication", 21st Annual Computer Security Applications Conference (ACSAC 2005), http://www.acsac.org/2005/papers/156.pdf , pp. 1-2

[6] Port Knocking http://www.portknocking.org/

[7] M. Krzywinski, "portknocking.org", "Port Knocking from the inside out", [Online] Available: http://www.portknocking. org/docs/portknocking_an_introduction.pdf, accessed Jan 2011, pp. 6-7.

[8] Hussein Al-Bahadili and Ali. H. Hadi, Arab Academy for Financial Sciences, faculty of Information Technology, Amman-Jordan, "Network Security Using Hybrid Port Knocking", Vol.10 No.8, August 2008, http://www.uop.edu.jo/download/research/members/382_1316_Network _Security_Using_Hybrid_Port_Knocking.pdf, pp. 2

[9] Jiun-Hau Liew, Shirly Lee and Ivy Ong, Department of Ubiquitous IT, Graduate School of Design and IT, Dongseo University, Busan, Korea and Hoon-Jae Lee and Hyotaek Lim, Department of Computer Engineering, Dongseo University, Busan, Korea, "One-Time Knocking Framework using SPA and IPSec", 2nd International Conference on Education Technology and Computer (ICETC), [Online] Available: http:// www.gbv.de/dms/tib-ub-hannover/635696215.pdf, pp. 4.

[10] Antonio Izquierdo Manzanares, Joaquin Torres Marquez, Juan M. Estevez-Tapiador and Julia Cesar Hernandez Castro, Universidad Carlos III de Madrid, Avda de la Universidad 30, 28911 Legan'es (Madrid), Spain, "Attacks on Port Knocking Authentication Mechanism", [Online] Available: http://www.users.cs.york.ac.uk/jet/papers/2005iccsa2.pdf

[11] Martin Krzywinski, Port Knocking, Linux Journal http://www.linuxjournal.com/article/6811

[12] Port Knocking, Webopedia http://www.webopedia.com/TERM/P/port_knocking.html

[13] Tom Eastep, Port Knocking http://www.shorewall.net/PortKnocking.html

[14] Jonathan Yarden, "Use Port Knocking to bypass firewall rules and keep security intact" [Online] Available: http://www.techrepublic.com/article/use-port-knocking-to-bypass-firewall-rules-and-keep-security-intact/5798871

[15] Mark Sanborn, "Add Port Knocking to SSH for Extra Security" [Online] Available: http://www.marksanborn.net/linux/add-port-knocking-to-ssh-for-extra-security/

[16] Port Knocking http://www.commandlinefu.com/commands/ view/2785/port-knocking

[17] Single Packet Authorization and Port Knocking Authentication system for GNU/Linux, Aldaba Port Knocking, http://www.aldabaknocking.com/

[18] Teemu Maki, Helsinki University of Technology, "Explicit Mechanisms for Controlling NAT/Firewall Systems Dynamically", TKK T-110.5190 Seminar on Internetworking 2007, [Online] Available: http://www.tml.tkk.fi/Publications/ C/23/papers/Maki_final.pdf

[19] Three locks for your SSH door, IBM developerWorks http://www.ibm.com/developerworks/aix/library/au-sshlocks/index.html

[20] Use Port Knocking #18 http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html.