CryptoKnocker: A New Approach to Port Knocking Without Shared Secret

Jun Hao Tan
National University of Singapore
junhao@nus.edu.sg

Anselm N. Foong
National University of Singapore
anselm.nicholas@nus.edu.sg

Rongshun Tan
National University of Singapore
rongshun@nus.edu.sg

Te Ye Yeo
National University of Singapore
teye@nus.edu.sg

ABSTRACT
Port knocking is a mechanism, which allows clients to open ports from outside a firewall. This is done automatically when someone sends the correct sequence of knocks. Common implementations consist of a sequence of timed packet "knocks" which are considered a shared secret. These secrets are susceptible to eavesdropping and replay attacks due to the open nature of a public channel.

In our project, we propose a new approach to port knocking by integrating modern techniques of authentication such as public key cryptography, nonce and one time password into the port knocking infrastructure to mitigate the above mentioned attacks.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: Security and protections

General Terms
Security

Keywords
Port knocking, public key cryptography, one time password, Challenge-response authentication

1. INTRODUCTION
This paper presents CryptoKnocker, a new approach to port knocking without the need for pre shared secrets. CryptoKnocker is a lightweight application that allows any user to easily implement and manage a port knocking daemon into their existing infrastructure.

Section 2 describes port knocking: an authentication mechanism that could dynamically hide services behind a firewall until access is validated and authorized. It also describes the existing approaches to port knocking. Section 3 describes how various features of CryptoKnocker such as the challenge response authentication, one time password and port control mechanism are implemented. It also explains how CryptoKnocker mitigates the common attacks that are plaguing other port knockers. Section 4 presents experimental results of the security and reliability of our proposed system as well as its performance impact. Section 5 describes the target beneficiaries as well as the extensibility of CryptoKnocker. Section 6 describes related work in port knocking. Lastly, section 7 presents our conclusion of the paper.

2. PORT KNOCKING
Port knocking is a computer networking technique of externally opening ports on a firewall by creating a connection attempt on a set of pre-specified closed ports. There are many variants of "knocking" style, which we will describe in this section.

2.1 Purpose of Port Knocking
The port knocking's primary goal is to provide an additional layer of protection through security by obscurity of concealing open ports from the public. Networks initially were deliberately designed to interconnect seamlessly and no prior authentication was required before machines are able to communicate. Therefore, port knocking attempts to fill the gap by providing an added layer of security that authenticates users before access is granted to them.

Furthermore, port knocking is also able to keep potentially vulnerable services hidden from the public while, simultaneously, making it accessible to authorized users. This greatly lowers the risk of any adversary being able to target services with zero day attacks or known vulnerabilities.

2.2 Various Schemes of Port Knocking
Port knocking generally operates on top of the 2 common transport layer protocols, namely, TCP or UDP, depending on the developers' implementation [1]. The TCP header is of minimum 20 bytes and contains fields such as the sequence number as well as an acknowledgment number of the packets and therefore is able to hold additional information such as the state of the communication, making them suitable for Multiple Packet Authorization (MPA). The UDP header on the other hand, is only 8 bytes long and is significantly stealthier than TCP and hence more suitable for Single Packet Authorization (SPA) or other undetectable authentication schemes so as to prevent unnecessary divulge of information to potential sniffers.

The earliest notion of port knocking would most likely have originated from a 2002 Intel Research publication by Barham, Hand, Isaccs, Jardetzky, Mortier & Roscoe [2]. Barham et al. introduced three techniques, Spread-Spectrum TCP, Tailgate TCP and Option-Keyed TCP, to address IP network vulnerabilities against denial-of-service (DOS) attacks by concealing services from non-authorized users. These techniques featured the idea of clients sending TCP SYN packets with SHA-1 hash sequences encrypted with a key that was shared with the servers, in order to authenticate themselves to the servers. On the server-side firewall lies a Silent Authentication Service, which only opens the concealed service port when the correct packet containing the hash is received while any other packets are silently discarded. Numerous studies [3] have been conducted on port knocking, and many were driven to extend the de-facto authentication mechanism of knock sequences in order to mitigate malicious attacks such as replay attacks, connection hijacking and denial-of-

service attacks. The general approaches include, but is not limited to[1]:

1) Introduction of One-Time Password (OTP) or nonce.
2) Hash-based Message Authentication Code.
3) Single/Multi Packet(s) Authorization.

2.2.1 One time password approach

1. **C** → **S**: Knock_seq, OTP
2. **S**: Verification of OTP and allow Client

The addition of OTP was introduced to prevent replay attacks. In 2004 BlackHat USA, David Worth introduced a Cryptographic One Time Knocking tool in which the client knocker would send an OTP knock via a UDP packet [4]. The server would, thereafter, verify the validity of OTP and authorized the access subsequently. The client, however, is unable to determine and verify the authenticity of the server.

2.2.2 Hash-based message authentication code

1. **C** → **S**: Payload + E(Ks, MAC_1(payload))
2. **S** : Computes MAC_2(payload) with MAC_1

The client and the server should have a secret key prior to communication and the knock packets will be hashed with a function such as SHA-1 and subsequently encrypted with the shared key. At the receiving end, the message authentication code (MAC) is recomputed by the server and compared with the received MAC. deGraaf, Aycock & Jacobson [5] addressed the need for stronger authentication by proposing a challenge response scheme incorporating HMAC. When the server receives a request from the client, it will issue a nonce as a challenge. The client then computes the MAC of the nonce together with the IP addresses of itself and the server's. The MAC is encrypted with the symmetric key and sent back to the server for validation. This ensures the integrity of the clients' messages, but relies on the common knowledge of the keys. A compromise of the key meant that an attacker could masquerade as either party; a single point of failure in the communication protocol.

2.2.3 Single / Multi Packet(s) Authorization

The traditional concept of port knocking via knock sequences is further reinforced with the idea of Single and Multi Packet(s) Authorization (SPA/MPA). Under the SPA and MPA, the client knocker specially crafts and sends a message with essential authentication information such as the MD5 sum of the current message, to the knock daemon. The daemon would monitor for incoming connections and control access based on the received packet. In the case of MPA, multiple packets are consolidated before the verification [6]. Instead of relying on the limited size of TCP/UDP headers to store the knock sequences, the SPA/MPA is able to enhance the security of the port knock scheme by encapsulating additional information into the payload. The maximum allowable length of the encapsulated information varies by the Maximum Transmission Unit of the communication interface; for example, 1500 bytes for Ethernet frames. [7]. In 2005, the first publicly published SPA port knocker: Fwknop by Michael Rash, provides clients with the option of either AES-128 or GnuPG asymmetric encryption up to 2,048-bit public/private key pair [8].

[1] Other schemes that maybe of interest: TCP steganography, Port-Knock with IPSec, IPv6 Address Knocking [14] [15].

3. CRYPTOKNOCKER

CryptoKnocker is a new approach to port knocking. It eliminates the need of a shared secret by introducing public key cryptography (RSA). It also features two factor authentication (2FA) which is based on time-based one-time password algorithm (TOTP) which is a standard supported by many authenticator including the widely used Google Authenticator and also a challenge response authentication which safeguards against attacks resulting from the loss of the private key and replays attack respectively.

CryptoKnocker consists of two key components, the client and the server. The client will handle the knocking of ports; the server is responsible for ensuring only authorized users are granted access to the service behind the port.

3.1 Architecture

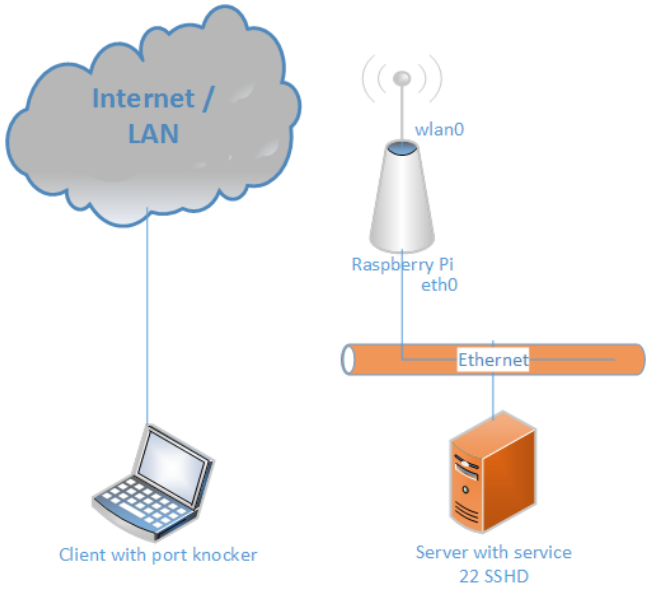

Figure 1 Possible Network Architecture

CryptoKnocker supports many different network architectures. One example of such architecture is illustrated in figure 1 which segregates the wired and wireless portion of the network. We use the above network architecture setup in our test environment with the wireless network simulating an external network and the wired network simulating the intranet.

CryptoKnocker server can be installed on any server that runs on the Linux operating system, example includes the Raspberry Pi, which can act as a firewall for the Server. By default, all ports will be closed with the exception of the knocking port (UDP 8888). Moreover, the knocking port would not respond to any incoming packet unless it is a knock. For the client to access any internal service, the client would have to first perform a knock on the port using the CryptoKnocker Client application. Upon successful authentication, the user would be granted access automatically into the server on the specified port, which is based on the user's IP address. The user may also opt to close the port via the same application once he has finished his existing session.

3.2 Protocol

Communication between the client and the server are done over UDP. Payloads are encrypted with public key cryptography. The daemon by default, would listen on UDP port 8888[2] for incoming knocks. Other than the knocking packets, the daemon would not respond to any traffic. This helps to mask over the presence of a listening port on the server.

3.3 Implementation

3.3.1 CryptoKnocker Client

The CryptoKnocker client consists of a knocker, which allows user to

1) Generate a pair of 2048-bit RSA key in PEM[3] format
2) Add CryptoKnocker public key to the client
3) Port knock CryptoKnocker.

Figure 2 CryptoKnocker Client

For keys generation, each key pairs are stored locally within the CryptoKnocker client folder. It also stores CryptoKnocker's public key within one of its sub folders. The keys' paths are contained in user.ini (user's public and private key) and server.ini (daemon's public key)

For port knocking, the user is required to input his user ID, the server's IP address, desired port to be opened and the OTP from two factor-authenticator. The client application then performs the necessary input validation to ensure input is properly formatted.

The user can either choose to open (knock) or close (lock) a port.

3.3.2 CryptoKnocker Server

The CryptoKnocker server consists of 2 key components, the CryptoKnocker daemon and the web interface. The CryptoKnocker daemon monitors for any encrypted packets from the client and processes the authentication protocol in order to open or close ports. In addition, CryptoKnocker daemon also logs down every authentications. The web interface provides administrators behind the server with a user-friendly platform to manage clients' CryptoKnocker public keys, port status and also registration and de-registration of new clients. The web interface is created with Django and jQuery to reduce development time, maximize web security and embellish the administration experience.

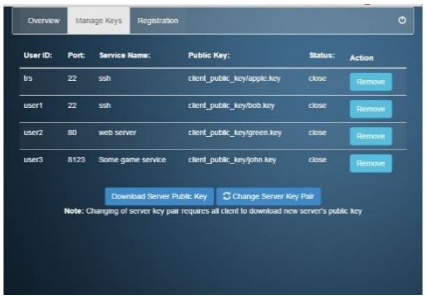

Figure 3 CryptoKnocker's Admin Web User Interface

[2] This port number can be easily change to any other UDP port depending on the requirement of the users

[3] PEM is a standard format for OpenSSL and is described in detailed in RFC 1424 [16]

3.3.3 Challenge response authentication

CryptoKnocker implements a challenge response authentication to authenticate users. The authentication process is as follows:

Legend:

C: Client, **S**: CryptoKnocker Server

PU: Public key, **PK**: Private Key

N: Nonce

Payload$_1$: <UserID, TypeOfRequest, IP, PortToOpen, OTP, N_C>

H: Hash function, SHA256

Challenge-Response Authentication

C → S: $E(PU_S, Payload_1) + Sig(PK_C, H(Payload_1))$

S → C: $E(PU_C < N_C, N_S>) + Sig(PK_S, H(< N_C, N_S>)))$

C → S: $E(PU_C, N_S) + Sig(PK_S, H(N_S))$

In the initial round of communication, the CryptoKnocker client sends a payload with the user's information and the request port to be unlocked on the server. In addition, a nonce value derived from current time stamp is concatenated with the aforementioned details, which will be encrypted with the CryptoKnocker Server's public key.

Once the CryptoKnocker Server receives the client's request, it performs the following verification:

1. Validity of request, request port and OTP
2. Freshness of nonce
3. IP address matches IP header

The CryptoKnocker Server then responds back with the client's nonce together with the CryptoKnocker Server's nonce, which is a random 9-bits integer. A time-based nonce is not used on the CryptoKnocker Server's response, as the nonce is predictable by any adversary including the user. Usage of time-based nonce will thereby allow any adversary to turn stale message into fresh message.

Finally, the client responds back with the CryptoKnocker Server 's nonce. At this point in time, if the server nonce sent by the client is fresh, the server will open the desired port to the specific IP address of the client.

The CryptoKnocker Server is designed to reveal limited information and cease all communication with the client once the payload authorization process fails.

3.3.4 Overcoming Network Address Translation

By embedding IP addresses in the payload, network address translation (NAT) poses a problem in CryptoKnocker as NAT devices would modify the source IP address of a packet causing the source IP address to be different from the one that is stored within the payload.

In order to address the concern, CryptoKnocker's client would first determine if the client is located on the same subnet as CryptoKnocker server. It proceeds as normal if they are in the same subnet.

In the event that the client behind a NAT device and the server is located in a separate network, CryptoKnocker could determine the public IP address of the client through the ipgetter module.

3.3.5 One time password

The private key of the user plays a major role in the authentication process. Therefore, it is crucial that the private key be kept secret and away from others. However, there is still a possibility of the private key being stolen and used. Moreover, the victim may not be immediately aware of it.

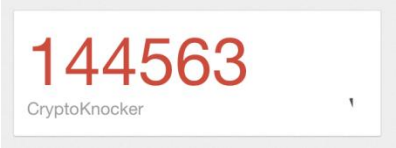

Figure 4 TOTP from Google Authenticator

As a countermeasure, one-time password (OTP), in the form of time-based one-time password algorithm (TOTP), is required when performing port knocking. CryptoKnocker uses TOTP provided by Google authenticator. The use of Google authenticator to get the TOTP is mandatory for all users.

3.3.6 Port control mechanism

The CryptoKnocker server has integrated with the python-iptables library to coordinate the chains and input rules of the Linux kernel firewall for granting access from a particular source IP to a destination service port. With the use of iptables, CryptoKnocker server is able to provide a more dynamic packet filtering environment for the network administrator. Moreover, specific checks are in-place to ensure identical client's IP address will not be able to open the same service port twice. Along with the specific checks, when a port close request is made either by the user or network administrator, the iptables will be filtered to ensure no existing or duplicate rules will coincide with the new DENY rule. These mechanisms aimed to counter a potential overflowing of iptables if users try to invoke multiple open or close requests.

3.3.7 Keys re-generation and revocation

Key revocation is an important aspect in any situations which includes stolen keys and compromised client machines. CryptoKnocker provides key revocation on the server. Keys can be easily revoked at the CryptoKnocker administration page.

User ID:	Port:	Service Name:	Public Key:	Status:	Action
trs	22	ssh	publickeys/apple.key	close	Remove
user1	22	ssh	publickeys/bob.key	close	Remove
user2	80	web server	publickeys/green.key	close	Remove
john	22	SSH	publickeys/john_URcrwqa.key	close	Remove

Figure 5 Keys Revocation administrative page

The user can also re-generate his or her keys using the same key generation function found in CryptoKnocker client. By entering the same user name, CryptoKnocker client will replace any existing key pair with a newly generated pair of public and private keys.

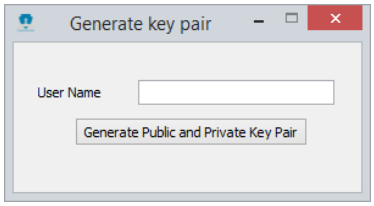

Figure 6 Keys pair generation or re-generation

3.4 Attacks and Mitigations

3.4.1 Man in the middle (MITM) attacks

A successful MITM attack often depends on an attacker's ability to impersonate the various endpoints in the protocol and pretend to be that particular endpoint towards the other party.

CryptoKnocker is not vulnerable to MITM attacks as it uses PKI to perform mutual authentication as part of the knocking protocol.

3.4.2 Replay attacks

Replay attacks are usually done by an attacker by capturing an earlier sequence of packets between a legitimate client and the CryptoKnocker server and thereafter replay the sequence of packets at a later time to impersonate as the client that is trying to communicate to the CryptoKnocker. An example of a replay attack would be an attacker capturing a series of knocks from a client, which informs the CryptoKnocker server to open port 23 to his IP address. When the client leaves, the attacker would just need to request for that client's IP and thereafter, replay the knocks, thus gaining access to that specific port.

CryptoKnocker mitigates replay attack by incorporating nonce as part of the communication protocol. A total of two nonces are used throughout the flow of the communication protocol and we define N_S as the CryptoKnocker server's nonce and N_C as the client nonce. The client nonce is always the current timestamp and therefore there will never be an opportunity for collision with an earlier used nonce. The CryptoKnocker's nonce is a 9 bit random integer, which is always used in concurrence with the client nonce and therefore, also has zero chance of a collision with an earlier packet.

Currently, we have assigned the freshness of a client nonce to thirty seconds and this value can be easily tweaked to suit the need of the different environment and requirements.

3.4.3 IP Address manipulation and Insider attacks

Insider attacks are defined in this paper as an authorized client, allowing an unauthorized client access into the restricted system via modifying the IP address that is residing in the payload. We assume the insider is not an administrator of CryptoKnocker. Otherwise, the administrator could provide the unauthorized user with a key.

One important requirement of CryptoKnocker is that it should only open ports to the correct IP addresses. Besides the IP header, CryptoKnocker also embeds the address within the payload.

If the port knocker is relying on the IP header, any attacker could intercept the packet, change the IP address and forward it to the daemon. In such a scenario, the attack could be conducted easily and the attacker is not required to have the knowledge of the payload content regardless of whether the payload is encrypted or not.

Another method is to encode the IP address of the client in the payload. The port knocker daemon will then utilized the address within the payload for ensuing communication instead of the one residing in the IP header. In such a case, the attacker who is intercepting the communication will not be able to modify this IP address as he or she does not have the encryption keys. However, this mitigation will not work if the client lied about his IP address.

In the implementation of CryptoKnocker, we added an extra layer of security by comparing and verifying both the IP address in the header and payload matches before opening the port to the specific IP address. It is noted that this is not an infallible solution

as the attacker and client may collude and manipulate both the IP address in the header and payload to circumvent the checks.

Another viable collusion scenario is when clients transfer all required information (private key, OTP, etc) to the attacker. In such a scenario, CryptoKnocker will also be unable to detect the attacks.

Hence, CryptoKnocker security measure will work well only if there is no collusion between the client and the attacker.

3.4.4 Spoofing attack

Traditional port sequence knocking is vulnerable to spoofing attacks, in which the attacker masquerades as the daemon. With the client performing only passive authentication, verification of the daemon's authenticity is not possible.

With CryptoKnocker, the attacker could still masquerade as the daemon. However, once the CryptoKnocker client receives the presumably firewall's response, the application will come to realize the received digital signature of the hashed nonce cannot be decrypted with the firewall's public key, and thus, connection is dropped.

Similarly, consider the scenario in which the attacker disguise as the client. The attacker could send the encrypted payload concatenated with the digital signature which is encrypted with the attacker's own private key. Upon receiving the request packet, the daemon is not able to successfully decrypt the signature with the client's public key and halts communication. Thus, the client and the daemon could verify each communicating party's authenticity through the challenge response authentication.

3.4.5 Leakage of a client's private key

Given that PKI is used for mutual authentication of endpoints, users may be concerned that if an attacker is able to gain a hold of their client private keys, they could trick the daemon into thinking that they are authorized users and thus, will open the ports for them.

CryptoKnocker mitigates this issue through the use of 2FA. The 2FA method that we have chosen to include into CryptoKnocker is one-time password (OTP) based on the time-based one-time password algorithm (TOTP), RFC 6238 [9].

TOTP was chosen as it is a standard OTP algorithm that is used by various industrial corporates for instance, Google Authenticator [10], Microsoft's One account [11] and Amazon's Web Services [12], and has authenticator standards built for many platforms.

Using TOTP, an attacker would not be able to port knock even if he manages to take control of the client's private key as he would not have the user's authenticator.

4. EXPERIMENT

4.1 Hardware

In our test environment, we have setup CryptoKnocker server on a Raspberry Pi. The setup is similar to figure 1. The Raspberry Pi is installed with "Debian wheezy" which provides it with networking capabilities similar to that of a home networking router. It uses a wireless dongle to connect to external networks i.e Internet. A machine with SSH service is connected to the Raspberry Pi and the port of SSH (22) is port forwarded on it. We have another laptop connecting to the same router as the Raspberry Pi. From there, the CryptoKnocker client on the laptop is ran to port knock CryptoKnocker server on the Raspberry Pi. In our setup, we did not discover any performance related issue.

4.2 Software

4.2.1 Wireshark

A packet captured during the challenge response authentication reveals no useful information on the communication. This is so as all the payloads are encrypted with 2048-bit RSA. Figure 7 shows the UDP dump of the challenge response authentication.

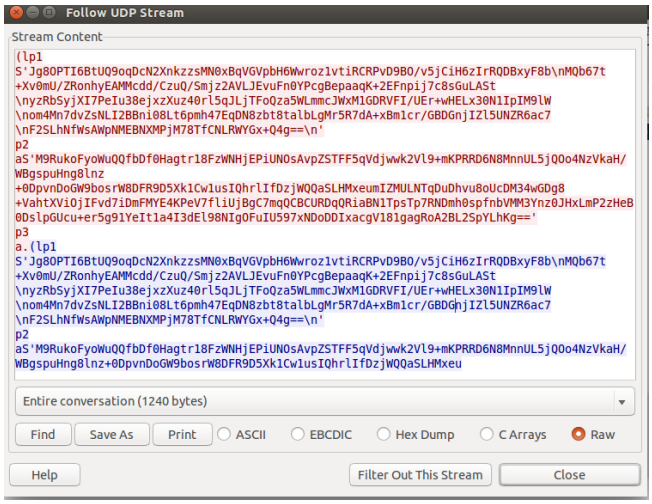

Figure 7 Packet capture of the challenge response authentication

4.2.2 Nmap

By applying the concept of security through obscurity, CryptoKnocker hides services from external networks. A normal port scan using Nmap (figure 8) reveals no opened ports, even though there is an SSH service waiting for connection on port 22.

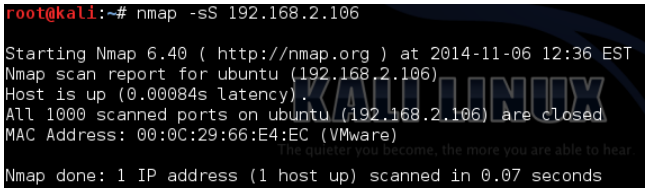

Figure 8 A normal TCP stealth scan on CryptoKnocker

CryptoKnocker server also hides the presence of an open port knocking port, by refusing to respond to any incoming traffic with the exception of port knocks. Nmap (figure 9) UDP scan shows the UDP port 8888 is closed, as CryptoKnocker do not respond to the scan.

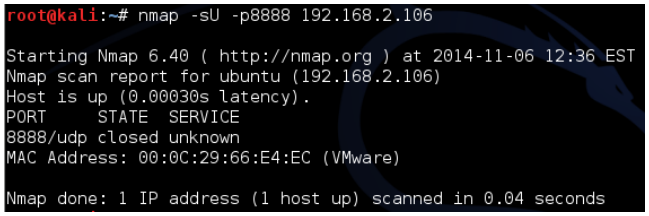

Figure 9 UDP scan on port knocking port

5. DISCUSSION

5.1 POTENTIAL AUDIENCES

CryptoKnocker provides a sophisticated security approach to port knocking technique, with minimal configuration required. It can be easily installed and deployed with a low cost credit card sized computer such as the Raspberry Pi. Therefore, CryptoKnocker is

advantageous for Small Office, Home Office (SOHO), as financial and human resources to maintain computer servers and networking infrastructure are often limited. Moreover, CryptoKnocker could also be installed onto most recent off-the-shelf commercial wireless routers that support third party firmware such as dd-wrt or tomato and thus would not need any additional hardware from the end users.

5.2 EXTENSIBILITY

CryptoKnocker is developed with open source tools and libraries such as Python, Django and iptables. It is available for anyone to download and modify. Hence, anyone with programming knowledge would be able to extend its functionality.

OTP functionality can be also extended to other platforms such as hardware token like YubiKey and SMS token.

6. RELATED WORK

There is one development pertinent to improving port knocking that is called Fwknop. Fwknop stands for the "Firewall knock operator", which uses single packet authorization as authorization scheme and provides both symmetric and asymmetric encryption functionalities such as Elgamal, GnuPG (GPG) and Rijndael [13]. Rather than trying to increase the number of functionalities, CryptoKnocker strives to provide sophisticated port knocking technology to anyone with minimal configuration required. CryptoKnocker is not a conclusion of which approach is better, but rather a system with different trade-offs.

7. CONCLUSION

In conclusion, CryptoKnocker aims to change the perspective of port knocking usage in a SOHO environment by providing a robust design to endpoint communications. Through the practice of a public key infrastructure together with a challenge and response authentication, replay attacks and man-in-the-middle attacks can be avoided. The extra verification supported by authenticators such as the Google Authenticator further reduces the likelihood of an adversary gaining unauthorized access via a stolen client key. The compact size of CryptoKnocker allows the system to be easily deployed in credit-card size computers. Technology-savvy personals with programming knowledge can effortlessly adapt CryptoKnocker to suit their network environment and security policies. We believe CryptoKnocker's small footprint coupled with its security objectives would lead to a wider adoption of port knocking technology within the small office environment.

8. ACKNOWLEDGMENTS

We would like to thank Associate Professor Hugh Anderson for providing us guidance and equipment needed for the development and testing of CryptoKnocker.

We would also like to thank ACM SIGCHI for allowing us to modify templates they had developed.

REFERENCES

[1] M. Krzywinski, "Port knocking from the inside out," 2005. [Online]. Available: http://www.portknocking.org/docs/portknocking_an_introduction.pdf.

[2] P. Barham, S. Hand, R. Isaacs, P. Jardetzky, R. Mortier and T. Roscoe, "Techniques for Lightweight Concealment and Authentication in IP Networks," July 2002. [Online]. Available: http://www.intel-research.net/Publications/Berkeley/012720031106_111.pdf.

[3] M. Krzywinski, "Port Knocking - Resources," Jan 2012. [Online]. Available: http://www.portknocking.org/view/resources.

[4] D. Worth, "COK - Cryptographic One-Time Knocking," Black Hat USA, 2004. [Online]. Available: http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-worth-up.pdf.

[5] R. deGraaf, J. Aycock and M. Jacobson, "Improved port knocking with strong authentication," in *Computer Security Applications Conference, 21st Annual*, 2005.

[6] V. Srivastava, A. Keshri, A. Roy, V. Chaurasiya and R. Gupta, "Advanced port knocking authentication scheme with QRC using AES," in *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, 2011.

[7] M. Rash, "Single Packet Authorization with Fwknop," Cipherdyne.org, December 2005. [Online]. Available: http://www.cipherdyne.org/fwknop/docs/SPA.html.

[8] M. Rash, "Single Packet Authorization – A Comprehensive Guide to Strong Service Concealment with fwknop," Cipherdyne.org, August 2014. [Online]. Available: http://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html.

[9] D. M'Raihi, V. I. S. Machani, D. C. M. Pei, S. J. Rydell and P. I. , "RFC 6238 - TOTP: Time-Based One-Time Password Algorithm," 28 October 2014. [Online]. Available: https://tools.ietf.org/html/rfc6238.

[10] "google-authenticator: Two-step verification," Google, 2013. [Online]. Available: https://code.google.com/p/google-authenticator/.

[11] M. Jeffrey, "Microsoft Account Gets More Secure," Microsoft, 17 Apr 2013. [Online]. Available: http://blogs.microsoft.com/blog/2013/04/17/microsoft-account-gets-more-secure/.

[12] "Multi-Factor Authentication," Amazon Web Services, 2014. [Online]. Available: https://aws.amazon.com/iam/details/mfa/.

[13] M. Rash, "Single Packet Authorization: The fwknop Approach," 10 September 2012. [Online]. Available: http://www.cipherdyne.org/blog/2012/09/single-packet-authorization-the-fwknop-approach.html.

[14] H. Liu, Z. Wang and Y. Liu, "Address Knocking: An Undetectable Authentication Based on IPv6 Address," 2012 13th International Conference.

[15] V. Y. Eugene, H. Nicholas and T. James, "Eugene, Vasserman Y.; Nicholas, Hopper; James, Tyra;," *International Journal of Information Security*, vol. 8, no. 2, pp. 121-135, 2009.

[16] B. Kaliski, "Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services," RSA Laboratories, Feb 1993. [Online]. Available: https://www.ietf.org/rfc/rfc1424.txt.