Design, Release, Update, Repeat: The Basic Process of a Security Protocol's Evolution

Young B. Choi
Department of Science, Technology, and Mathematics
College of Arts and Sciences
Regent University
1000 Regent University Drive
Virginia Beach, Virginia 23464-9800
USA

Nathan D. Hunter
Department of Science, Technology, and Mathematics
College of Arts and Sciences
Regent University
1000 Regent University Drive
Virginia Beach, Virginia 23464-9800
USA

***Abstract*—Companies, businesses, colleges, etc. throughout the world use computer networks and telecommunications to run their operations. The convenience, information-gathering, and organizational abilities provided by computer networks and the Internet is undeniably useful. However, as computer and network technology continues to become increasingly advanced, the threat and number of cyber-attacks rises. Without advanced and well-developed security protocols, companies, businesses, colleges, and even ordinary individuals would be at the mercy of malicious hackers. This paper focuses on security protocols such as PGP, 3PAKE, and TLS's processes, design, history, advantages, and disadvantages. Research for this article was conducted through reading numerous scholarly articles, conference articles, and IT and information security textbooks. The purpose of this article is to expose and elaborate on vulnerabilities in popular security protocols, introduce lesser-known protocols with great potential, and lastly, provide suggestions for future directions and modifications to improve security protocols through qualitative and theoretical research.**

***Keywords*—Security; Protocol; 3PAKE; PGP; quantum; service, SSL; TLS; DSQC; QSDC; cyber; hashing; DOMIH; SHA**

I. Introduction

As technology advances in computers, smart phones, networks, etc. and technological features such as Internet of Things and smart homes are implemented, there is a need for improved and better-updated security for the public. For each technological advancement, there comes a price: security vulnerabilities. Various applications on the Internet along with computer operation systems are updated on a regular basis to fix specific, security vulnerabilities which hackers have exploited. There is an invisible war that continuously rages on between these protocols and cyber-attacks. Security protocols are responsible for protecting Internet applications and computer operations systems, thereby securing an individual's confidential data [9]. However, once a security protocol becomes popular, individuals ranging from IT specialists to ordinary citizens tend to take its security promises for granted. A large portion of our research is to remind and display to others the vulnerabilities that exist in popular security protocols and provide theoretical paths and methods to fix these vulnerabilities. Our research is meant to show that all security protocols have different strengths, weaknesses, and vulnerabilities, and many of them can efficiently reduce cyber-crime to a minimum, provided they are continuously researched, updated, and monitored.

II. Pretty Good Privacy

A. PGP History and Breakdown

Pretty Good Privacy (PGP) is a simple and well-known, but yet highly-qualified, security protocol. The idea and implementation of PGP, which was created by Philip Zimmermann, was to formulate a stout and secure encryption scheme that could be used effortlessly by the average individual [15]. Being a hybrid cryptosystem, PGP combines some of the best available cryptographic algorithms. According to Whitman and Mattord, PGP's security solution has six services: authentication through digital signatures, compression, message encryption, key management, e-mail compatibility, and segmentation.

The combination of authentication through digital signatures, message encryption, and key management focus primarily on message integrity. This is usually accomplished by using various Secure Hash Algorithms (SHA). SHA1 takes plaintext from a given message, computes a 160-bit hash code based on the message, encrypts the hash using DSS (Decision Support System) or RSA, and lastly, attaches the encrypted hash to the original message. This allows the recipient to use the sender's public key (in regards to key management) to decrypt the message, and using the same encryption process, the recipient will generate a second hash value. If this newly-generated hash value matches the sender's hash, then the received message is proven genuine, and nobody has tampered with the data. Other variations of message encryption and key management include 3DES (Triple Data Encryption Algorithm), IDEA (International Data Encryption Algorithm), or CAST with a 128-bit session key. This aspect of PGP is extremely vital for integrity and is a great example of its effectiveness. Though without superb compression techniques, PGP's transfer speed and security would drastically decrease [16].

Ke, Hong, and Zu state in an article regarding compression that, "Compression is one of the most important techniques in data management, which is usually used to improve the query efficiency in database" [5]. PGP uses a freeware ZIP algorithm that compresses data to save space and heighten security.

Security is heightened because the smaller the file is, the fewer chances an attacker has to locate or discover patterns in the data with which to perform frequency analysis [16].

The fifth service in PGP's security solution is e-mail compatibility. Using a process by the name of Radix-64 which encodes non-textual data, PGP ensures that e-mail systems like SMTP (Simple Mail Transfer protocol) can transfer a given message. This is done before encryption but after the message receives its digital signature. It also maintains the required 8-bit ASCII code blocks. Whitman and Mattord directly explain this process saying that, "The format maps three octets of binary data into four ASCII characters and appends a cyclic redundancy check (CRC) to detect transmission errors."

After all the encryption, compression, and conversion functions have been processed, the final service, segmentation, is performed. Segmentation is the process of subdividing messages into an easier, transferable data stream size. This process occurs at the sender's end. At the recipient's end, PGP reassembles the message blocks into its original form prior to decompression and decryption [16].

TABLE I. PGP'S SIX SERVICES

Service	**PGP's Six Services**	
	Purpose	***Algorithm/Process***
Authetication through digital signatures	Computes a 160-bit hash code based on the plaintext of a given message using SHA-1	SHA1
Compression	Uses the Zip Algorithm to compress the message	ZIP
E-mail Compatibility	Encodes non-textual data with Radix-64 and transfers data translated into ASCII by CRC	Radix-64, ASCII, CRC
Message Encryption	Encrypts the hash code with either DSS or RSA	DSS or RSA
Key Management	Encrypts the hash code with 3DES, IDEA, or CAST	3DES, IDEA, or CAST
Segmentation	Occurs after encryption and conversion functions. From the sender, it subdivides messages into an easier, transferable size. From the receiver, it reassembles the segment's message blocks before decompression and decryption	No Algorithm Used

B. PGP: Weaknesses and Vulnerabilities

In regards to security, PGP is quite effective in securing e-mail messages and encrypting other types of data fields. After researching and studying PGP, we found three main weaknesses in the protocol:

- Since PGP is based on public key encryption techniques, the receiver of any messages or applications secured by PGP must have the same version as the sender. Key exchange relies on the interaction between public and private keys, and they are only identical provided the sender and receiver have the same version [15]. This is a great disadvantage for PGP because if two computers are not running the same PGP version, they will not be able to communicate with each other.
- SHA1 is extremely outdated in terms of efficiency and security, and although more advanced versions like various SHA2 and SHA3 have been released, PGP still does not incorporate the most advanced hashing algorithms. Almost every SHA algorithm is vulnerable to collision attacks which occur when an attacker can change the integrity of the message while maintaining a hash identical to the original [6].
- Messages transferred by PGP are not encrypted until after the message is compressed. This means less text is encrypted, making it easier for hackers to decrypt the message.

C. Suggestions

- In regards to PGP compatibility, a simple way to deal with this dilemma would be to, as a standard, always use the latest version of PGP, thereby increasing security and maintaining compatibility. However, it would be wiser to instruct the developers of PGP to make all versions of PGP compatible with each other. This can be done by either manipulating the method by which PGP's public and private keys interact or disregarding key exchange, focusing more on encryption for security.
- After much research toward hashing algorithms, our conclusion is to replace SHA1 with a more recent variation of MIH (Multi-Index Hashing). In their article regarding MIH, Yanping, Hailin, Hongtao, & Qingtan quoted that, "Multi-index hashing (MIH) is the state-of-the-art method for indexing binary codes, as it divides long codes into substrings and builds multiple hash tables." They admitted that there were still a lot of issues and vulnerabilities imbedded in MIH. In response to this discovery, Yanping, Hailin, Hongtao, & Qingtan proposed, diagramed, and invented a new data-oriented version of MIH called DOMIH (Data-Oriented Multi-Index Hashing). After a lot of research and development, they concluded that, "Experiments conducted on famous datasets show the obvious performance improvement of our method" [17]. Implementing DOMIH in place of any variation of SHA would greatly improve PGP as a security protocol.
- Lastly, PGP should encrypt all messages before and after compression so hackers will encounter much more turmoil in attempting to decrypt the information.

III. SECURITY SOCKETS LAYER AND TRANSPORT LAYER SECURITY

A. History

In 1994, Netscape Company designed and released a security protocol called SSL (Secure Socket Layer) [17]. This protocol is defined as, "A communications protocol used to secure sensitive data during e-commerce" [12]. SSL, over a course of five years, received many updates and re-releases including various SSL 2.0s and SSL 3.0s, ultimately leading to the release/replacement of TLS (Transfer Layer Security) in 1999. The differences between TLS 1.2 (the newest version as of today) and SSL 3.0 are very minute, the most considerable difference being TLS 1.2 is more secure than SSL 3.0 [1,14].

B. TLS 1.2 Breakdown

The TLS 1.2 protocol has two major procedures: the hand-shaking protocol and the record protocol.

1) The hand-shaking protocol starts by negotiating the cipher suite, authenticating the server and, optionally, the client, and ends with establishing the session keys. Within this sub-protocol, there are three steps:

a) The Hand-shake: Negotiates the version of the protocol, session identifier, compression method, cipher type, and lastly, the master secret.

b) The Alert Protocol: Alerts the user if there is an error in the protocol by either sending a failure alert (exits the user out of the program without a choice) or a warning alert (gives the user a choice to exit the program).

c) The Change Cipher Specification Protocol: Informs the user if the sender wants to change keys and, in response, sets up a new, secure session.

2) After the handshake protocol comes the record protocol. This protocol secures the application data with the established session key and confirms that the application's data has maintained its integrity.

TLS 1.2 is perhaps the most popular security protocol on the Internet because of its ability to effectively provide authentication, integrity, and confidentiality for both the sending and receiving parties. It is also compatible with HTTP (hypertext transfer protocol) and uses the URL (Uniform Resource Locator) scheme which encompasses a great percentage of the Internet's language [14]. Despite being a popular and relatively secure protocol, TLS 1.2 has numerous security vulnerabilities.

C. TLS 1.2: Weaknesses and Vulnerabilities

Most hackers exploit TLS 1.2 through the protocol's services and tools by using remote timing attack, truncation attack and cipher suite rollback, denial of service, change cipher spec dropping, or vulnerabilities in the implementation of the protocol. Errors and defects in TLS 1.2's *hand-shake protocol* allows an attacker to use ARP (Address Resolution Protocol) spoofing to execute a MITM (man-in-the-middle) attack. TLS 1.2 uses a hashing algorithm called MD5 to secure its integrity, but unfortunately, MD5 is vulnerable to collision attacks.

One of the most dangerous and damaging attacks toward TLS 1.2 would involve a malicious server attack. Since TLS 1.2 only protects and secures the link between the client and server, a hacker can simply construct a spoofed malicious server to obtain any client's personal information. Asadzadeh-Kaliahi, Pavandeh, & Ghaznavi-Ghoushchi stated that, "TLS does not provide the real authentication with the use of certificate that only has an identity-based perspective but not an intelligent and behavioral one" [1]. After researching, discovering, and listing all of these vulnerabilities, our conclusion is that TLS 1.2 is not the most popular protocol on the Internet because of security but rather because of convenience.

D. Suggested Modifications

Although a lot of the vulnerabilities in TLS 1.2 would require an elite hacker to exploit, the pure number of vulnerabilities is staggering to say the least. TLS 1.2 basically needs to be completely revised, rebuilt, and re-released. Work on TLS 1.3 is on the way, but little is known of what differences will be included. For TLS 1.3 to be completely secure, many qualities in TLS 1.2 must be improved.

If the hashing algorithm DOMIH was implemented instead of MD5, TLS would no longer experience collision attacks. Asadzadeh-Kaliahi, Pavandeh, & Ghaznavi-Ghoushchi developed and invented a revised *hand-shake protocol* which prevents an attacker from using ARP spoofing to execute a MITM attack. They also studied various trust models to revise TLS in order to defend against malicious server attacks [1]. If these precautions are considered while TLS 1.3 is being developed, the new release may be able to stand secure against numerous cyber-attacks.

IV. Three Party Authentication Key Exchange Protocol (3PAKE)

A. Common Attacks Used Against 3PAKE

3PAKE is a type of authentication key exchange protocol (AKE). AKE protocols are designed to assist two devices that are communicating over an insecure channel to maintain a secure session key to protect their subsequent communication. When 3PAKE was first designed, the most efficient way of breaking the protocol was password guessing techniques such as dictionary attacks or brute force attacks. Recent security breaches and research have revealed five main attacks used against 3PAKE [3]:

- Replay Attack: Occurs when a stream of messages between two parties is copied by an attacker and sent to two or more parties [8].
- Impersonation Attack: Occurs when an attack can masquerade as a communication entity.
- Guessing attack: Occur when an attacker repeatedly attempts to guess a user's password, sometimes using dictionary or brute force attacks [10].
- Modification Attack: Occurs when a hacker either alters the packet header address, directing a given message to a different destination, or modifies the data on a specific computer [4].
- Known-Key Attack: Occurs when an attacker has discovered the session/security key within a protocol's hashing algorithm or key exchange information [7].

B. 3PAKE Models

There are three ways to secure 3PAKE: using a server public key, using symmetric cryptosystems, or using neither server public keys nor symmetric cryptosystems. In 2009, a woman named Huang proposed a new variation of 3PAKE which used neither server public keys nor symmetric

cryptosystems. Unfortunately, this security protocol variation was shown to be vulnerable to numerous impersonation attacks. This theme carried on into 2011 when Chang et al. tried to improve on this protocol by implementing XOR functions to likewise remove both server public keys and symmetric cryptosystems. Although this did result in constructing a communicationally effective 3PAKE protocol, experts discovered this protocol variation was not only weak to key compromise impersonation (KCI) attacks (a specific category of impersonation attacks) but also password guessing attacks [3,10].

Recently, Xiong et al. revealed that every 3PAKE protocol lacking server public keys are not secure against Key Compromise Impersonation (KCI) attacks. Therefore, the 3PAKE protocols which use server public keys only to prevent password guessing and impersonation attacks are more secure and applicable than the other two approaches. Using this new information, a man named Tso created yet another variation of 3PAKE using server public keys only to prevent guessing attacks. However, in a scholarly article, Farash & Attari explain that Tso's 3PAKE protocol is still vulnerable to impersonation and guessing attacks. Farash & Attari refute Tso's protocol piece by piece while also implementing their own 3PAKE protocol, a protocol immune to impersonation and guessing attacks [13]. Below is a table summing up each designers' protocol choices and vulnerabilities.

TABLE II. 3PAKE MODELS

Security Choice and Attacks	**3PAKE Models**			
	Huang's Model	***Chang et al.'s Model***	***Tso's Model***	***Farasha & Attari's Model***
Security Choice	Uses neither public server keys nor symmetric cryptosystems	Uses neither public server keys nor symmetric cryptosystems	Uses only public server keys	Uses only public server keys
Replay Attack	Secure	Secure	Secure	Secure
Impersonation Attack	Insecure	Insecure	Insecure	Secure
Guessing Attack	Insecure	Insecure	Insecure	Secure
Modification Attack	Secure	Secure	Secure	Secure
Known-Key Attack	Secure	Secure	Secure	Secure

C. Conlcluding Suggestion

Our research of 3PAKE protocols has concluded that Farasha & Attari's model is theoretically invulnerable to the most common and damaging 3PAKE cyberattacks. Implementing this model as the standard 3PAKE version would greatly enhance the security of 3PAKE.

V. QUANTUM SECURITY PROTOCOLS

A. Origin and History

Although various methods of quantum secure communication protocols have been around since before 1984, it was not until Shimizu and Imoto produced the first DSQC (Deterministic Secure Quantum Communication) using Bell states in 1999, and Bostrom and Felbinger invented and released a Bell state based QSDC (Quantum Secure Direct Communication) protocol in 2002, popularly known as the ping-pong protocol, that these protocols began to draw a considerable amount of attention.

The idea of the first quantum protocol, quantum key distribution (QKD), was to establish a secure connection and key between two legitimate users through the transmission of quantum bits (qubits). DSQC was invented and implemented because more direct quantum methods lacking a prior established key became more favored. QSDC, on the other hand, was not necessarily an improvement over DSQC, but rather a different type of quantum protocol widely considered to be more secure. This is mostly because QSDC was and is more popular [11].

B. Dense Coding: Positive or Negative?

Not long after 2004, to improve Cai and Li's version (latest version at the time) of QSDC, Deng, Long, and Lui constructed a one way two step protocol, lacking the prior dense coding in Cai and Li's version. In an article concerning QSDC and DSQC, Shukla, Banerjee, & Pathak stated that, "This simple idea of inclusion of dense coding to increase the efficiency of a secure direct communication protocol has considerably influenced the future development of QSDC and DSQC protocols."

Over the years, there had been great controversy as to whether dense coding was beneficial or detrimental to quantum security protocols. This dilemma revolved around the process of these protocols. In QKD, DSQC, and QSDC, data transfer is secured by splitting information into at least two different pieces: the first being the quantum piece and the last being the classical piece.

The quantum piece will arrive first, and the quantum protocol will check to assure that the integrity of the quantum piece has not been altered or eavesdropped. Next, the classical piece will be sent, and the encoding process begins. For data to be retrieved from a quantum spectrum, the receiver must have simultaneous access to both pieces. If a hacker intercepts, tampers with, or eavesdrops on the quantum piece, the classical piece will not be sent. In other words, the hacker cannot hold on to the quantum piece and wait for the announcement of the classical piece.

Many different variations of quantum protocols were invented over the years, and a lot of them incorrectly implemented this information separation, making the protocol insecure. Protocol after protocol, it seemed that there was a direct, beneficial relationship between using dense coding and correct information separation on a quantum level. This resulted in security professionals taking dense coding as a preferred method for constructing a secure quantum security protocol whether it was a QSDC or DSQC based protocol.

Recently, Shukla, Banerjee, & Pathak acknowledged this decade-old, false assumption in an article by stating, "keeping this in mind several authors have designed inefficient (non-maximally efficient) protocols of DSQC and QSDC using W states and have considered their protocols efficient." The authors go on to explain that they designed a DSQC and QSDC protocol lacking such dense coding, and it is more secure and efficient than any past protocol of its type. According to their

research, Shukal, Banerjee, & Pathak's protocols are roughly 17% more efficient than any past implementation [11].

C. Suggested Actions

Any modern version of DSQC or QSDC should run Shukal, Banerjee, & Pathak's version, for our research shows their models and methods provide the highest degree of security regarding quantum security protocols.

However, even though this new evolution of QSDC and DSQC protocols has been one of the most influential and upgrading moments in the history of quantum security protocols, an efficient, secure, and operational quantum security protocol is far from being released [11]. Regardless, there is still a great amount of potential bound within quantum security protocols. Developers, analysts, and scholars should invest and contribute more time, effort, and research in this area, for unlocking the potential in these protocols could have a huge, positive impact on the field of cyber security. Overall, awareness of quantum security protocols is necessary for all these actions.

VI. Summary of our Suggestions

Pretty Good Privacy (PGP):

- Always run the latest version of PGP or manipulate PGP's key management service to allow senders and receivers using different versions of PGP to successfully communicate with each other.
- Replace SHA1 with DOMIH.
- Encrypt each message before and after compression occurs.

Three Party Authentication Key Exchange (3PAKE):

- Implement Farasha & Attari's model as the standard version of 3PAKE.

Transport Layer Security 1.2 (TLS 1.2):

- Replace MD5 with DOMIH.
- Implement Asadzadeh-Kaliahi, Pavandeh, & Ghaznavi-Ghoushchi's new Handshake protocol and trust models to remove MITM (man-in-the-middle) and malicious server attacks respectively.

Quantum Security Protocols:

- Establish Shukal, Banerjee, and Pathak's models for QSDC and DSQC as the standard version of their corresponding quantum security protocols.
- Encourage awareness and support of quantum security protocols so that more research, developing, and effort will be focused on unlocking their potential.

VII. Conclusion

The idea and goal in making and releasing innovative and updated security protocols is to secure information, web applications, operating systems, etc. from nefarious hackers, especially from recent exploitations. However, designers of such security protocols must understand that it is simply a matter of time before their newly-released protocols are proven inefficient or unsecure. Even the most popular and renown security protocols have significant vulnerabilities.

Although PGP is quite efficient and effective as a security protocol, issues such as compatibility, hashing algorithms, and encryption cause PGP to be hypothetically vulnerable to hackers. Every time 3PAKE is updated and re-released, it is not long before half-a-dozen exploits and errors are discovered in the protocol. For example, TLS is the most popular security protocol on the Internet, and there are dozens of security loopholes within its program, making it vulnerable to remote timing attack, truncation attack and cipher suite rollback, denial of service, and change cipher spec dropping [1,13,16].

Security protocol designers must know that it is not a matter of "if" their protocol is exploited but rather a matter of "when" [2]. Regardless, cyber security is a massively growing field, especially concerning security protocols where engineers continuously design, release, and update new security protocols. This process must continually repeat in order to keep individuals as safe as they can be from cyber-crime.

References

[1] M. Asadzadeh-Kaliahi, A. Pavandeh, and M. B. Ghaznavi-Ghoushchi, "TSSL: Improving SSL/TLS protocol by trust model," Security & Communication Networks, vol 8(9), p. 1659-1671, 2015, in press.

[2] D. Basin and C. Cremers, "Know Your Enemy: Compromising Adversaries in Protocol Analysis," ACM Transactions On Information & System Security (TISSEC), vol. 17(2), Doi:10.1145/2658996, in press.

[3] M. Farash and M. Attari, "An efficient client-client password-based authentication scheme with provable security", Journal of Supercomputing, vol. 70(2), p. 1002-1022, 2014, Doi:10.1007/s11227-014-1273-z, in press.

[4] M. S. Karanade, "Chapter 5: Mobile security," World Press, 2015, retrieved from https://mskarande.files.wordpress.com/2015/12/chapter-5-mobile-security.pdf.

[5] Y. Ke, Z. Hong, and K. Lu, "VParC: A compression scheme for numeric data in column-oriented databases," International Jornal of Information Technology (IAJIT), vol. 13(1), p. 1-11, 2016, in press.

[6] N. Kishore and B. Kapoor, "Attacks on and advances in secure hash algorithms," International Journal of Computer Science, vol 43(3), p. 25-34, 2016, in press.

[7] B. Mennink and B. Preneel, "On the impact of known-key attacks on hash functions," Lecture Notes in Computer Science, vol 9453, pp. 59-84, 2015, Springer Berlin Heidelberg, in press.

[8] Replay attacks, n.d., retrieved from the Microsoft Website https://msdn.microsoft.com/en-us/library/aa738652(v=vs.110).aspx

[9] I. Sharpe, Hacking: Basic security, penetration testing, and how to hack. San Bernardino, CA: CreateSpace Independent Publishing Platform, 2015.

[10] W. Shuhua, C. Kefei, and Z. Yuefei, "Enhancements of a three-party password-based authenticated key exchange protocol," International Arab Journal of information technology (IAJIT), vol 10(3), p. 215-221, 2013, in press.

[11] C. Shukla, A. Banerjee, and A Pathak, "Improved protocols of secure quantum communication using W states," International Journal of theoretical physics, vol 52(6), p. 1914-1924, 2013, Doi:10.1007/s10773-012-1311-7, in press.

[12] R. M. Stair and G. W. Reynolds, Principles of information systems: A managerial approach. Boston, Massachusetts: Course Technology, Cengage Learning, 2014.

[13] Z. Tan, "A communication and computation-efficient three-party authenticated key agreement protocol," Security & Communication Networks, vol 6(7), p. 854-863, 2013, Doi:10.1002/sec.622, in press.

[14] S. Turner, "Transport layer security," IEE Internet Computing, vol. 18(6), 60-63, 2014, Doi:10.1109/MIC.2014.126, in press.

[15] M. W., White, Data communications and computer networks: A business user's approach. Boston, Massachusetts: Course Technology, Cengage Learning, 2013.

[16] M. E., Whitman and H. J. Mattord, Principles of information security. India: Course Technology, Cengage Learning, 2015.

[17] M. Yanping, Z. Hailin, X. Hongtao, and S. Qingtan, "Fast search with data-oriented multi-index hashing for multimedia data," KSII Transactions on Internet & Information Systems, 9(7), p. 2599-2613, 2015, Doi:10.3837/tiis.2015.07.015, in press.