International Journal of Computer Applications (0975 - 8887)
Volume 176 - No.31, June 2020

Diffie Hellman Stand the Test of Time (Protocol's Limitations, Applications and Functional Divergence)

Avenash Kumar
Department of Computer Science
National University of Computer and Emerging Sciences
Karachi, Pakistan

Sufian Hameed
Department of Computer Science
National University of Computer and Emerging Sciences
Karachi, Pakistan

ABSTRACT

Cryptography plays a vital role in protecting secret information, as secure communication between two parties over the internet is the necessity and cannot be overstated. The Diffie-Hellman Key Exchange (DHKE) protocol is the well-known asymmetric algorithm formulated by its namesakes Whitfield Diffie and Martin Hellman in 1976. It allows two parties to securely exchange shared secret over an insecure communication channel, without using any pre-shared secret. However, protocol's theoretical assumptions and design often associated with some serious security flaws. This motivates cryptographic community to propose different variants of DHKE protocol. The major intend of this research is to examine both empirical and theoretical vulnerabilities of DHKE protocol. Which leads us to determine true rationales behind different variations of DHKE protocol. By reading this manuscript, it is hoped that application security experts will get good understanding of cryptographic primitives. These primitives are important and should be considered when designing or implementing any security protocol such as DHKE.

General Terms

Diffie Hellman, Key Exchange

Keywords

Diffie-Hellman, key exchange, asymmetric cryptography, secure communication, cryptographic standards

1. INTRODUCTION

In 1976, Whitfield Diffie and Martin Hellman publish their landmark paper named "New Directions in cryptography"[1]. In which they have presented secure key exchange protocol. Where two parties can interchange their secret key over insecure communication channel.

The protocol works under the notion of trapdoor function [1] (mathematical function which is easy to compute in one direction but computationally expensive in opposite direction [2]). This is suitable where shared secret keys are treated as session keys which can be further used with symmetric crypto systems (e.g. AES, 3DES, e.t.c) to make sure message authentication as well as message integrity [3]. Though the actual mechanism of the protocol is straightforward and undemanding. However, protocol's implementation is vulnerable over several adversarial attacks. This is due to limited knowledge of programmers regarding basic cryptographic primitives [4].

The main aim of this paper is to provide complete docket of adversarial models on which DHKE protocol's implementation was/is/will vulnerable. The readers of this manuscript will get good understanding of cryptographic primitives. These primitives are important and should be considered when implementing DHKE or any other security protocol.

To generate shared key K, let two parties be Alice and Bob first agree upon parameters p and g. Parameter p refers to any large prime number, where g is the primitive root of modulo p also known as generator of p. There is also a strong restriction in the agreement that g must lie in a range $1 < g < p - 1$, as the two parties communicate over insecure communication channel so there will be a case when protocol doesn't work and encourage a passive attacker to expose all the communicating messages. Example, when g equals to 1 or $p - 1$ shared secret key K on both sides is always 1 [5].

In the presence of active and passive attackers, when the two communicating parties (Alice and Bob) initiate handshake mechanism, the protocol prosecutes as per follows:

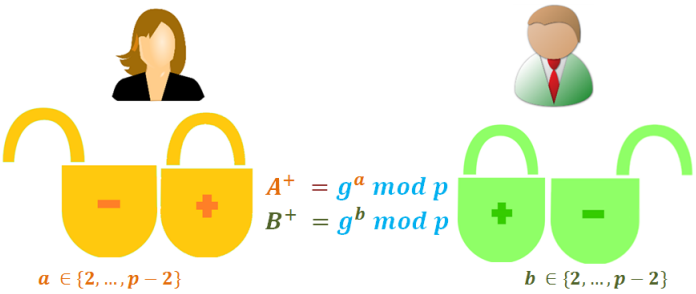

Fig. 1. Public/Private key generation.

(1) Alice and Bob select their private keys, a and b respectively from a random set $\{2, ..., p - 2\}$, as shown in Figure 1.

(2) Alice computes her public key A^+ and sends it to Bob. In the response Bob compute his public key B^+ and sends it to Alice, as shown in Figure 2.

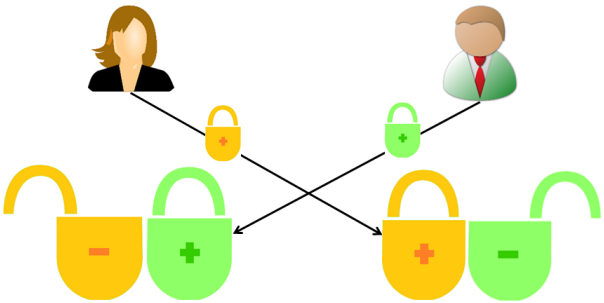

Fig. 2. Key exchange.

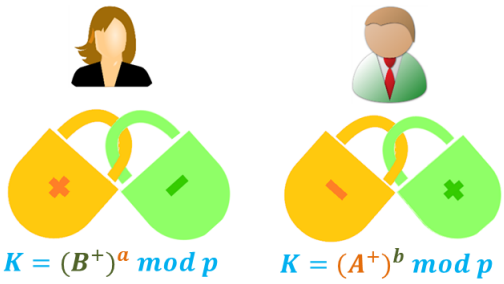

Fig. 3. Secret key generation.

(3) Knowing the fact that Alice already knows the public key of Bob B^+, she can compute shared secret key K, similarly, on the other hand Bob can also compute K with the fact that he already received the public key of Alice A^+, as shown in Figure 3.

Definition 1. *i) The **Computational Diffie Hellman (CDH) problem** states that: Given a generator g, modulo p, public keys A^+ and B^+ in a group, the goal is to compute K.*
*(ii) The **CDH assumption** states that: It is computationally in-feasible to elucidate CDH problem, given the parameter g, p, A^+ and B^+.*
*(iii) The **Discrete Logarithm (DL) problem** states that: Given a generator g and modulo p and public key A^+, the goal is to decompose a from g.*
*(iv) The **DL assumption** states that: It is computationally in-feasible to elucidate DL problem, given the parameters g, p, and A^+.*

The security of DHKE protocol is majorly relying on the fact that, adversary who has access to values which are publicly known such as g, p, A^+, and B^+ cannot calculate K from it. It is also called *Computational Diffie Hellman (CDH)* assumption. This assumption is derived from *Discrete Logarithm (DL)* problem in the way that, if an attacker knows an effective algorithm to break *DL* problem can easily break CDH assumption.

2. VULNERABILITIES

Cryptographic vulnerabilities in network protocols are not unusual. Therefore, such dilemmas always receive serious attention from cryptographic community. These vulnerabilities are sometimes associated with protocol due to the common network attacks, weak mathematical design model or flaws in protocol's implementation. DHKE protocol is widely used method for exchanging keys over insecure network channels. Due to which similar issues are often associated with the protocol, thus, it affects computational texture of the algorithm.

2.1 Network Attacks

2.1.1 Man in the Middle Attack (MiMA). The classical version of DHKE protocol doesn't authenticate participants when the key exchange mechanism takes place, thus, it is difficult to detect man in the middle attack in standard DHKE protocol [6]. In MiMA an active attacker (Eve) can be efficient enough to break DHKE protocol. During the conversation Eve can send, modify or delete messages, which are exchanged between two parties. In order to do that, Eve uses his public key E^+ and interact with Alice and Bob. While Alice and Bob are under the assumption, that they have shared their respective secret key with each other (see Figure 4). This is possible due to lack of authentication primitives in DHKE protocol.

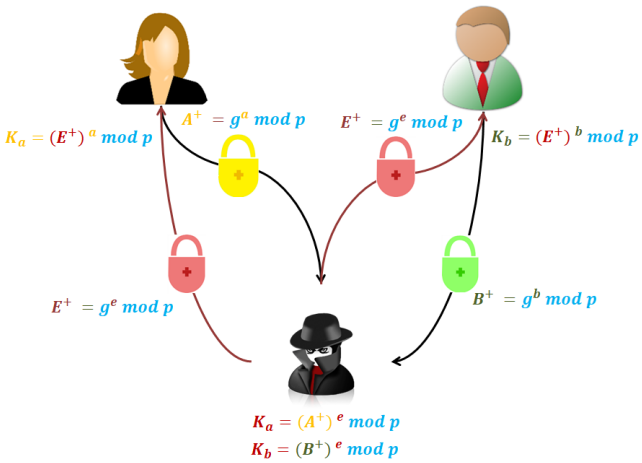

Fig. 4. Man in the Middle Attack (MiMA).

In order to understand this attack, let Alice wants to send her secret message m to Bob. To do so, she uses any symmetric crypto-system (AES, or 3DES) and encrypt m using K_a, where K_a is a shared secret key between Alice and Eve.

(1) Alice encodes her secret message m using $ENC(K_a, m)$ function and sends it over public network.
(2) Eve intercept her encrypted message, and decrypts it with a key K_a which he already knows.
(3) Eve swaps m with $ENC(K_b, m')$ and sends it to Bob. On the other side Bob receives the message. Considering Alice as a recipient and decrypts with key K_b which he shared with Eve, as shown in Figure 5.

Thus, the protocol is clearly vulnerable against MiMA as the message integrity and as well as participants' authentications are violated.

2.1.2 Logjam Attack. Logjam was a type of an active attack detected in 2015 export ciphers of TLS. It allows attacker to downgrade vulnerable TLS connections to 512-bit export-grade cryptography [7]. It is another variant of FREAK attack but instead of RSA, Logjam affects Diffie-Hellman Ephemeral (DHE) key exchange protocol. The actual cause behind this attack was the major flaw found in TLS. Which allows attacker to read

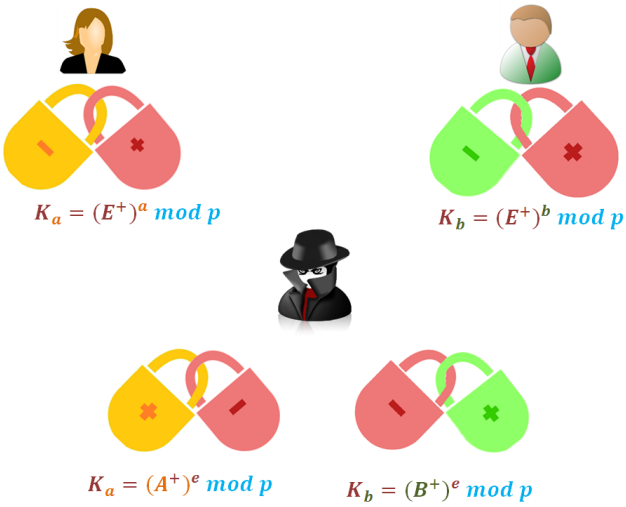

Fig. 5. Alice and Bob shared secret key with Eve.

or inject data into the connection stream. Any server which supports DHE_EXPORT ciphers was influenced with this attack. It was estimated that out of Top 1 million, 8.4% domains become vulnerable [7].

Logjam was not an implementation bug. However, it was direct flaw in the underline architecture of the TLS protocol [8]. In the preprocessing stage of TLS/SSL connection, client forwards list of supported cipher suites to the server, in the response server select one suite for the purpose of handshake mechanism (TLS maintains the list of cipher suite so that, during the process of handshake old systems can still be able to connect with new machines that use new protocols). "Export Suite" is one type of the cipher suites which uses 512 bits public key. During the world war-II US government impose strict export regulations on crypto. The purpose was supposed to spy on foreigners, since 512 bits are easy to factor [9]. The vulnerability comes from the fact that most of the servers were still support "Export Ciphers". In fact, most of the SSL servers use same prime p while exchanging session keys. Cryptanalysis believed this was safe if new keys are exchanged for every connection. However, this thinking was enough to underestimate active adversaries. Because if an attacker invests a lot in computation power in cracking private key of weak DHKE instance (decomposing a from $g^a mod p$), same computations can be reuse for other instances as well [10].

TLS allows servers to select their own parameters in order to execute DHKE protocol. Due to which immense majority of servers employ common prime numbers. Experts who were involved in detecting logjam attack estimates that only two different 512 bits primes were used for 92% of Alexa top 1M domains which support DHE_EXPORT (see Table *2.1.2*) [10].

The attack model of Logjam requires Man in the Middle when the handshake is performed between two communicating parties. By being in middle, attacker modifies *ClientHello* packet(s) to force the server to use an Export Ciphersuite. In the response server construct weak parameters for public key and forwards following messages to complete handshake (see Figure 6).

(1) *ServerHello*: Specifies supported cipher suite with respect to the list that server receives from client. Attacker modifies this message for the purpose of maintaining trust of victim client.

Table 1. Top 512-bit DH primes mostly used by Top 1M servers of Alexa

Source	Popularity	Prime
Apache	82%	9fdb8b8a004544f0045f173 7d0ba2e0b274cdf1a9f5882 18fb435316a16e374171fd1 9d8d8f37c39bf863fd60e3e 300680a3030c6e4c3757d08 f70e6aa871033
mod_ssl	10%	d4bcd52406f69b35994b88d e5db89682c8157f62d8f336 33ee5772f11f05ab22d6b51 45b9f241e5acc31ff090a4b c71148976f76795094e71e7 903529f5a824b
(Other)	8%	(463 distinct primes)

(2) *ServerKeyExchange*: Consists of server certificate which is composed of 512 bits modulo p, generator g and public key of server. When using export version of Ephemeral Diffie Hellman or normal DHKE the structure and the sending mechanism of the messages was not secure in a way that server only signs the certificates instead of communication which is required during ServerHello messages.

(3) *ServerHelloDone*: Defines that handshake mechanism from Server's is complete.

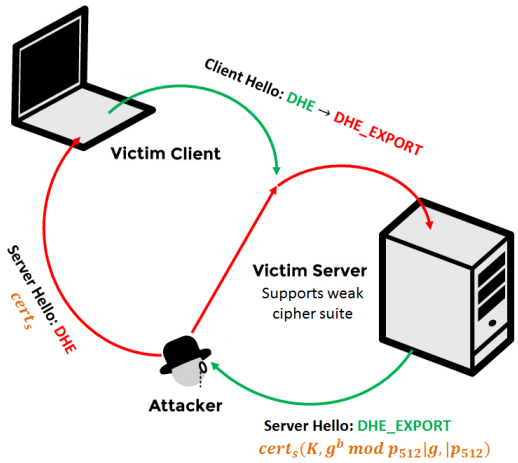

Fig. 6. Logjam Attack.

Researchers who were involved in detecting Logjam vulnerability believes that, even 768 and 1024-bit long prime numbers are risky in use as their processing power, required to crack p is within the range of academic teams and state-level attackers respectively [7].

2.2 Attacks in the Context of Number Theory

Though adversarial model of man in middle attack completely burst the underline security of the system but Eve needs to be an active and commanding adversary in order to cement bridge over two communicating parties. On the other side, attacks which can disturb emphasized mathematical model of DHKE offers luxury of observing network traffic passively. In these types of attacks, attacker acquires access on shared secret key itself.

Table 2. Cases for generating same K on both sides

	$g=1$	$g=p-1$	$g=p+1$
$e \equiv 0 \bmod 2$	1	1	1
$e \equiv 0 \bmod 1$	1	$p-1$	1

2.2.1 Degenerated Attack. Degenerated attacks are associated with DHKE protocol due to its mathematical formulation. If an attacker is efficient enough to launch such attack, protocol would no longer be effective. To do so, generator g should be equal to modulo inverse of $gmodp$. If such a case occurs, shared secret key on both sides would always be either 1 or $p-1$. Fortunately, from an attacker's perspective setting up such situation could be strenuous in a well implemented DHKE protocol, since both participants select generator g from set $\{2, ..., p-2\}$. However as far as an inside attacker is concerned, it is equally likely that an adversary can successfully decipher the ongoing communication of two parties by composing a smart input (see Table *2.2.1*) value for g^e [4].

2.2.2 Simple Exponents Attack. In DHKE when the values of exponents (either a or b) are deterministic the protocol becomes vulnerable against an attack known as Simple Exponents Attack. In such type of attack if an attacker is efficient enough in intercepting one of the private keys of the two participants and he can be able to successfully decipher all the communication of two parties, for example, when $a = 1$ in that case $g = g^a$. Attacker who monitors the network actively will be able to detect and can easily break the protocol. Another case could be attacker compute g^i and compare entire set with g^a or g^b. Thus, it is very difficult to find-out where to draw a boundary line, what set of values should DHKE protocol allows when selecting exponents, because in that case any set of values i cloud break the entire protocol. In either way it seems logical to declare that values of a or b are must not equal to 1.

2.2.3 Simple Substitution Attack. In this type of attack, the main intend of an attacker is to substitute an intelligent value for public keys i.e. A^+, B^+ which leads both parties to compute unity shared secret key on their respective sides. This type of attack is an example of careless implementation which normally doesn't report.

(1) Eve perceive public keys of Alice and Bob i.e. A^+, B^+ and respectively substitute with unity (1).

(2) When Alice receives Bob's public key and vice versa, both end up with unity shared secret key i.e. $K = 1$.

If applications are not capable of detecting unity public key, in that case protocol is vulnerable to simple substitution attack. The same statement is also applicable for the cases when any of the two exponents (a or b) holds a value which are multiple of either $p-1$. Based on Euler's totient theorem [11]:

$$g^{\varphi(p)} \equiv 1 \, mod \, p, \quad \varphi(p) = p - 1 \tag{1}$$

$$g^{\gamma \cdot \varphi(p)} \equiv 1 \, mod \, p, \quad \gamma \geq 1 \tag{2}$$

It is habitually good practice to confirm condition mentioned below:

$$1 < A^+, B^+ < p - 1$$

3. APPLICATIONS

3.1 Diffie-Hellman in SSL

Security is one the most important ingredient of online business in order to create trusted environment, where consumers must feel sanguine about the prospects of online transactions. Secure Socket Layer (SSL) provides the foundation of that trust by creating secure connection. To proof the assurance for the visitors, web browsers provide visual indications such as green bar or lock icons.
The term "Trusted Environment" refers in the context of ensuring confidentiality, authenticity, and integrity. There are three different version of Diffie-Hellman are currently used in order to provide the trusted path between the two communicating parties [12].

(1) Anonymous Diffie-Hellman
(2) Fixed Diffie-Hellman
(3) Ephemeral Diffie-Hellman

3.1.1 Anonymous Diffie Hellman. Anonymous Diffie-Hellman: Anonymous DHKE doesn't provide any validation or authentication mechanism for the communicating parties. Due to this fact the original version of Diffie-Hellman is also known Anonymous Hellman. By the means of Networking Attacks such 'man in the middle' or Attacks in the context of 'number theory', an active attacker can intercept the messages / transactions of two communicating parties. In order to resist such attacks authenticated version of DHKE protocol should been used. In case if an application uses this version of DHKE protocol, a call to SSL_get_peer_certificate can return NULL. Under the normal circumstances this is the only case where SSL_get_peer_certificate return NULL [12].

3.1.2 Fixed Diffie Hellman. Unlike Anonymous Diffie-Hellman, Fixed Diffie-Hellman key exchange provides authentication primitives by means of server's certificates. These certificates carry public and shared parameters of DHKE protocol which was signed by certificate authority (CA). On the other hand, as far as client side is concerned, clients provide its DHKE parameters either by means of certificate or via key exchange message. In this way under the notion of authenticity and integrity, fixed key is exchanged between client and server [12].

3.1.3 Ephemeral Diffie Hellman. In the context of SSL/TLS Ephemeral Diffie-Hellman (DHE) enables Forward Secrecy (FS), which cites a reliable communicating protocol in which compromised long term keys do not affect past communication of two parties. In other words, DHE defend old sessions against future attacks, which are related to secret keys. This property also applies for different variations of Elliptic Curve such as ECDHE (Elliptic Curve Diffie Hellman Ephemeral) [12].
As far as the authentication is concerned DHE doesn't provide implicit authentication at all, because every time system generates distinct key neither Alice nor Bob are certain about the intended recipient. In order to enable authentication mechanism DHE is often used with RSA, PSK or ECDSA etc [12].
Current Version of SSL/TLS supports lots of cipher suites which assist DHE in order to enable authentication [13], e.g.

(1) Ephemeral Diffie Hellman with RSA (DHE-RSA) key exchange.
(2) Elliptic Curve Ephemeral Diffie Hellman with RSA (ECDHE-RSA) key exchange.
(3) Elliptic Curve Ephemeral Diffie Hellman with ECDSA (ECDHE-ECDSA) key exchange.

(4) Pre Shared Key with Diffie Hellman (DHE-PSK) key exchange.

(5) Pre Shared Key with Elliptic Curve Diffie Hellman (ECDHE-PSK) key exchange.

3.2 Diffie-Hellman in SSH

Secure Shell (SSH) is a network level protocol which is responsible securing remote login and other network services over the un-protected communication channel. Key exchange is the major component of SSH, where multiple parties decide to agree upon shared key in order to communicate over insecure medium.
The protocol executes in three in different steps which are:

(1) "Hello" Phase: This a phase is also called Handshaking in which two parties exchange their identity. Various number of algorithms are involved which are responsible for providing supported DHKE key groups.

(2) In the 2nd stage steps involved in DHKE protocol are executed in order to generate shared key on both sides.

(3) In the final stage application key composed via Message digest, Session ID and Share secret key.

3.3 Diffie-Hellman in IPSec

IPSec (Internet Protocol Security) is initiated by Internet Engineering Task Force (IETF); it is an extension of the Internet Protocol (IP) in network layer of OSI model. It is used to provide secure communication medium by means of authentication and encrypting IP packets.
Unlike SSL and SSH, which are responsible for securing traffic, which is originated via application, IPSec is application independent, it is designated to work with all type of IP traffic. This approach provides a level of abstraction and transparency in which neither application nor the user needs to grasp anything regarding encryption.
Like SSL and SSH, IPSec also uses public key encryption and DHKE for entity authentication and shared secrets respectively. However symmetric algorithms are used to generate cipher text of bulk data. Before IPSec begins the encryption mechanism, some major details are exchanged between the two parties. The information is exchanged via IKE (Internet Key Exchange) protocol (defined in RFC 2409).
IKE accomplishes the task of key exchange in into two phases, Phase 1: It convince two communicating parties to agree dynamically on security parameters. In 2nd phase via shared secret key encryption is performed on exchanged information which governs the encryption parameters for the actual data.

4. CONCLUSION

This research aims to present hypothetical assumptions and designing flaws of DHKE protocol. In doing so, we have discussed the limitations in different variants of DHKE protocol which are supposed to be secure compare to the classical DHKE protocol. It is anticipated that this research will help protocol designers of upcoming generations when designing any cryptographic protocol. We also hoped that similar type of research will conduct for other security protocols, which could be a big step to assure reliability in cryptographic protocol with ingredients of real world.

5. REFERENCES

[1] W. Diffie, and M. Hellman, "New directions in cryptography," in *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[2] A. May, "New RSA vulnerabilities using lattice reduction methods," PhD. dissertation, *University of Paderborn,* 2003.

[3] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *Cryptology?Eurocrypt 2000,* Springer Berlin/Heidelberg, 2000.

[4] J. F. Raymond, and A. Stiglic, "Security Issues in the Diffe-Hellman Key Agreement Protocol," in *IEEE Trans. Information Theory,* vol. 22, Jan. 2002.

[5] D. M. Burton, "An Introduction," in *The History of Mathematics,* 7th ed., McGraw-Hill, 2011.

[6] C. M. Chen, L. Xu, W. Fang, and T. Y. Wu, "A Three-Party Password Authenticated Key Exchange Protocol Resistant to Stolen Smart Card Attacks," in *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the 12th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kaohsiung, Taiwan, Nov 21-23, 2016,* vol. 1, pp. 331–336.

[7] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman,... and B. VanderSloot, "Imperfect forward secrecy: How Diffie-Hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security,* Oct. 2015 pp. 5–17.

[8] J. A. Halderman, and V. Teague, "The new south wales ivote system: Security failures and verification flaws in a live online election," in *International Conference on E-Voting and Identity, September 2015,* Springer International Publishing pp. 35–33.

[9] P. Siriwardena, "Designing Security for APIs," in *Advanced API Security,* Apress, Berkeley, CA, 2020, pp. 33–67.

[10] N. Samarasinghe, and M. Mannan, "Another look at TLS ecosystems in networked devices vs. Web servers," in *Computers and Security*, January 2019 vol 80, pp. 1–3.

[11] L. Euler, "Theoremata arithmetica nova methodo demonstrata," in *Novi Commentarii academiae scientiarum Petropolitanae,* vol.8, pp.74–104, 1763.

[12] OpenSSLWiki, "Information for D̈iffie Hellman," 2015. [Online]. Available: `https://wiki.openssl.org/index.php?title=Diffie_Hellman&action=info`. Accessed on: May 12, 2020.

[13] S. Gallenmller, D. Schffmann, D. Scholz, F. Geyer, and G. Carle, "DTLS Performance-How Expensive is Security?," in *arXiv preprint arXiv:1904.11423*, April 2019.