privateDH: An Enhanced Diffie-Hellman Key-Exchange Protocol using RSA and AES Algorithm

Preprint · May 2021

DOI: 10.13140/RG.2.2.23938.40647

Ripon Patgiri
National Institute of Technology, Silchar

privateDH: An Enhanced Diffie-Hellman Key-Exchange Protocol using RSA and AES Algorithm

Ripon Patgiri, *Senior Member, IEEE*

Abstract—RSA cryptography is an asymmetric communication protocol, and it is facing diverse issues. Recent research works suggest that RSA security has already broken. On the contrary, AES is the most used symmetric-key cryptography protocol, and it is also facing issues. Literature search suggests that there is an issue of cryptanalysis attacks. A shared secret key requires for AES cryptography. The most famous key exchange protocol is Diffie-Hellman; however, it has an issue of the number field sieve discrete log algorithm attacks. Moreover, recent research suggested that Diffie-Hellman is less secure than widely perceived. Moreover, there is another issue of Logjam attack that allows man-in-middle attack in Diffie-Hellman. Thus, we combine RSA, AES, and Diffie-Hellman algorithm to provide security on the key exchange protocol, called privateDH. Our key objective is to provide security to the Diffie-Hellman Algorithm. Therefore, privateDH does not share the data publicly with the intended party. Instead, privateDH encrypts all shareable data in the time of key exchange by encrypting using the AES algorithm. privateDH uses the RSA algorithm and retrieves the public key to avoid a man-in-the-middle attack. Thus, we demonstrate how to provide security to the Diffie-Hellman algorithm to defeat various kinds of attacks.

Index Terms—Cryptography, RSA cryptography, Diffie-Hellman, Symmetric, Asymmetric, Public key, Encryption, Attacks, Security, Networking, Communication.

1 Introduction

Cryptography is the most prominent research challenge to protect the data from adversaries in communication. It can be applied in asymmetric and symmetric communications. RSA cryptography is the most used asymmetric cryptography algorithm [1], [2]. However, symmetric communication requires key exchange protocol and encryption/decryption algorithm. Therefore, the most famous key-exchange protocols are Diffie-Hellman [3], Elliptic-curve Cryptography (ECC) [4], [5], and Elliptic-Curve Diffie-Hellman (ECDH) [6] algorithm. AES is the most used encryption/decryption standard for block cipher [7]. Symmetric key cryptography is the most secure way to exchange data between two parties, and therefore, it requires a key-agreement protocol. If the key-agreement protocol is broken, then the entire communication becomes insecure. Therefore, key-agreement protocol demands security to protect from the attackers.

Recent literature suggests that RSA cryptography has an issue of integer factorization [8], [9]. Also, it faces issues of low-exponential computation [10], [11], [12]. Also, RSA requires Optimal Encryption Asymmetric Padding [13]. Many new attacks have already been reported [14], [15]. The RSA cryptography have already been broken [16], [17], [18], [19]. Therefore, many new techniques have been suggested to secure RSA [20], [21]. Moreover, the public key cryptography requires a trusted third party to distribute the public key, which is a slow process [22]. Similarly, symmetric-key cryptography is prone to many attacks, particularly brute-force, cryptanalysis [23], and fault attacks [24]; however, symmetric key cryptography has diverse applications, for instance, healthcare [25], [26].

The Diffie-Hellman algorithm is a widely used symmetric key exchange protocol. Securing key-agreement protocol is a key challenge because recent research suggests the failure of the Diffie-Hellman algorithm [27], [28]. Adrian *et al.* finds that Diffie-Hellman algorithm allows man-in-middle-attack [27]. Therefore, Diffie-Hellman requires security to protect from the attackers. The root cause is publicly shared keys which makes Diffie-Hellman a failure key-exchange protocol. Thus, our key objective is to provide security to the Diffie-Hellman algorithm. The Diffie-Hellman algorithm becomes private and remains no more public. All the keys are shared secretly by encrypting using AES. Therefore, We provide security to the Diffie-Hellman algorithm by RSA cryptography and AES.

In this article, we provide security to the Diffie-Hellman algorithm. We derive a private Diffie-Hellman algorithm, privateDH for short. The sender requires a public key of the receiver, and therefore, the sender retrieves the receiver's public key from a trusted third party. The sender generates a random key for AES using a true random number generator [29], [30] and sends the generated random key to the receiver by encrypting the receiver's public. The receiver decrypts the key from the received ciphertext using its private key. The receiver initiates the Diffie-Hellman key exchange process. The receiver generates two prime numbers [31] and sends the prime numbers to the sender by encrypting using the received key. Both the receiver and sender encrypt the whole communication using a single key, and the AES

- *Department of Computer Science & Engineering, National Institute of Technology Silchar, Cachar-788010, Assam, India*
E-mail: ripon@cse.nits.ac.in and URL: http://cs.nits.ac.in/rp/

Manuscript received Month XX, 20XX; revised Month XX, 20XX.

algorithm is used to encrypt the communication. Therefore, privateDH maintains secret communication between the sender and the receiver in the entire computation process of the shared secret key. The conventional Diffie-Hellman algorithm shares the numbers over an insecure channel without encryption; however, privateDH encrypts all the number and shares secretly between the sender and receiver.

The main contributions of this article are outlined below-

- We derive an enhanced Diffie-Hellman key exchange algorithm to provide a security while exchanging the key.
- We demonstrate how to use RSA and AES in the Diffie-Hellman algorithm.
- Also, we show how to achieve absolute security without communication overhead. But there is a computation overhead; however, it is negligible.
- We also analyze the various possible issues of our proposed solution.

The Diffie-Hellman algorithm is protected from an attacker using RSA and AES cryptography. Therefore, privateDH shows strong resistance against attackers and is able to compute shared secret key securely. In this paper, we demonstrate how to secure the Diffie-Hellman algorithm from attackers. There is no key-agreement protocol to get extra security while computing the shared secret key, to the best of our knowledge.

This article is organized as follows- Section 2 discusses the preliminaries of the proposed system. Section 3 establishes the enhanced Diffie-Hellman algorithm. Section 4 analyzes the proposed system in-depth. Finally, Section 5 concludes the article.

2 BACKGROUND

In this section, we establish preliminaries and discuss the RSA, AES, and Diffie-Hellman algorithm. Table 1 shows the important notations and their definition.

2.1 RSA Cryptography

RSA cryptography is a well-known asymmetric cryptography algorithm that relies on public-private key cryptography [1], [2]. In RSA, the receiver publishes its public key and retains its private key. Anyone can send a message to the receiver by encrypting using the public key. Only the receiver can decrypt the message. The public key is published, and it is a long-term key. Attackers can break the security if the public-private key pair is not renewed. Moreover, the public-private key is time-consuming cryptography, but it is useful in many applications.

RSA algorithm requires two large prime numbers; let the two large prime numbers be P_r and Q_r. The product of these two prime number is made public; let it be $n = P_r \times Q_r$. The P_r and Q_r are kept secret. The RSA algorithm calculates $\lambda(n) = \text{LCM}((P_r-1),\ (Q_r-1))$ which is also kept secret. Let us choose a public key exponent e where $1 < e < \lambda(n)$. Moreover, the e is coprime with $\lambda(n)$. Finally, the RSA algorithm computes a secret key as $d \times e \equiv 1\ (mod\ \lambda(n))$. RSA algorithm publishes $(n,\ e)$ as a public key, and the rest parameters are kept secret.

TABLE 1
Important symbols used in the article and their descriptions.

Notation	Description
$\mathcal{A}$	The Sender
$\mathcal{B}$	The receiver
P_r and Q_r	Two prime numbers for RSA algorithm
e	The exponent of RSA algorithm
$\lambda(n)$	LCM of two prime numbers
d	The private key of RSA algorithm.
m	It is the converted integer form of message
c	It is ciphertext in RSA
mod	Modulus operator
P and g	Two prime numbers for Diffie-Hellman algorithm
a and b	True random number generated
$\mathcal{SK}$	Shared secret key
$\mathcal{K}$	Secret Key for AES
$Enc^{\mathcal{B}_{pub}}$	Encryption process using RSA by the public key of the $\mathcal{B}$
$Enc^{\mathcal{B}_{priv}}$	Decryption process using RSA by the private key of the $\mathcal{B}$
$\zeta^{\mathcal{A}}$	Ciphertext from $\mathcal{A}$
$\zeta^{\mathcal{B}}$	Ciphertext from $\mathcal{B}$
$Enc^{\mathcal{K}}$	Encryption by AES using secret key $\mathcal{K}$
$Dec^{\mathcal{K}}$	Decryption by AES using secret key $\mathcal{K}$
β	Bit length of a key.

2.1.1 Encryption

Let the m be the message to be encrypted, then RSA Algorithm encrypts using Equation (1).

$$m^e \ \equiv \ c\ (mod\ n) \tag{1}$$

2.1.2 Decryption

The sender sends the ciphertext c to the receiver using encrypting by the receiver's public key. The receiver receives the ciphertext c and decrypts using its private key by Equation (2).

$$c^d \ \equiv \ m\ mod\ n) \tag{2}$$

2.2 Advanced Encryption Standard

Advanced Encryption Standard (AES) is a block cipher algorithm for symmetric cryptography [7]. It is known as Rijndael. The round keys are derived from the cipher key using AES key schedule method. Moreover, AES performs an XOR operation in each byte of the state to combine with a byte of the round key, which is known as AddRoundKey. Each byte is replaced with another according to a lookup table in a non-linear substitution step, which is known as SubBytes. The last three rows of the state are shifted cyclically a certain number of steps in a transposition step, which is known as ShiftRows. MixColumns operates on the columns of the state to combine the four bytes in each column in a linear mixing operation. AES performs SubBytes, ShiftRows, MixColumns, and AddRoundKey in rounds 9, 11, and 13. Similarly, it performs SubBytes, ShiftRows, and AddRoundKey in rounds 10, 12, and 14.

2.3 Diffie-Hellman Cryptography

Table 2 shows the algorithm of Diffie-Hellman key exchange. Initially, both the sender and receiver share two prime numbers P and g over the public channel. Then, the

TABLE 2
Diffie-Hellman Key Exchange Protocol for three secret key generation.

Sender $\mathcal{A}$	Attacker $\mathcal{E}$	Receiver $\mathcal{B}$
P	P	P
g	g	g
a		b
$A = g^a \ mod \ \mathcal{P}$		$B = g^b \ mod \ P$
B	A, B	A
$\mathcal{SK} = B^a \ mod \ P$		$\mathcal{SK} = A^b \ mod \ P$

$\mathcal{A}$ and $\mathcal{B}$ generate random key a and b using a true-random number generator, respectively. Both the keys a and b are kept secret. The $\mathcal{A}$ computes $A = g^a \ mod \ \mathcal{P}$, and sends the A to $\mathcal{B}$. Similarly, the $\mathcal{B}$ computes $B = g^b \ mod \ \mathcal{P}$ and sends the B to $\mathcal{A}$. The $\mathcal{A}$ and $\mathcal{B}$ can compute the shared secret key $\mathcal{SK} = B^a \ mod \ \mathcal{P}$ and $\mathcal{SK} = A^b \ mod \ \mathcal{P}$, respectively.

3 PRIVATEDH: THE PROPOSED SYSTEM

We propose a private Diffie-Hellman algorithm, privateDH for short, to overcome the existing issues of the Diffie-Hellman algorithm [27]. It is a straightforward enhancement of the existing Diffie-Hellman algorithm to provide absolute security for key exchange. privateDH combines the Diffie-Hellman algorithm with RSA cryptography [1] and AES symmetric cryptography [7]. This section demonstrates how to provide absolute security to the key exchange protocol using the combination of RSA and AES cryptography.

3.1 Assumption

We have a few assumptions to establish the enhanced Diffie-Hellman key-exchange protocol, which are outlined below-

- We assume that the P_r and Q_r of RSA are large prime numbers, and integer factorization takes many years. Also, we assume that the a in the RSA is large, which is approximately equivalent to $2^{16} - 1$.
- We assume the sender and the receiver are valid and intended users.
- The symmetric key cryptography uses AES cryptography algorithm [7].
- privateDH depends on the trusted third party, and the trusted third party is assumed to be valid.

3.2 Description

Let us assume that the sender $\mathcal{A}$ wishes to send a message to the receiver $\mathcal{B}$. Therefore, the $\mathcal{A}$ encrypt sends hello to $\mathcal{B}$ along with the key. The message is encrypted using the public key of the $\mathcal{B}$, and therefore, only the $\mathcal{B}$ can decrypt the message. The receiver $\mathcal{B}$ decrypts the hello message using its private key. Instead of the hello message, the $\mathcal{A}$ generates a random key, encrypt it using the public key of the $\mathcal{B}$, and send the ciphertext to the $\mathcal{B}$. The $\mathcal{B}$ decrypts the key and initiates the Diffie-Hellman key exchange protocol. Thus, the $\mathcal{A}$ and $\mathcal{B}$ share the initial secret key successfully. The $\mathcal{B}$ generates two random prime numbers, encrypt the two prime numbers by AES using the initial shared secret key, and send the ciphertext to the $\mathcal{A}$. The rest of the communications are encrypted using the shared key. The Diffie-Hellman requires two prime numbers P and g to share between $\mathcal{A}$ and $\mathcal{B}$. All communications are encrypted using the shared key. Conventionally, Diffie-Hellman shares all the keys publicly excepts randomly selected numbers and shared secret key. However, we suggest encrypting all the keys using the initial shared key between $\mathcal{A}$ and $\mathcal{B}$. Thus, the Diffie-Hellman algorithm shares all the keys secretly.

3.3 Trusted Third Party

The sender $\mathcal{A}$ requires the public key of $\mathcal{B}$ to send a message. Thus, the $\mathcal{A}$ retrieves the public key of $\mathcal{B}$ from the trusted third party. The trusted third party facilitates the public keys. The trusted third party cannot be malicious; otherwise, privateDH fails.

3.4 Combination of the RSA, AES, and Diffie-Hellman algorithm

TABLE 3
Encryption-decryption using asymmetric and symmetric key to enhance Diffie-Hellman algorithm in the presence of attacker.

Sender $\mathcal{A}$	Attacker $\mathcal{E}$	Receiver $\mathcal{B}$
$\mathcal{K} = \text{KEYGENTRNG}(\,)$ $\zeta_1^{\mathcal{A}} = Enc^{\mathcal{B}_{pub}}(\mathcal{K})$	$\mathcal{B}_{pub}$	
	$\zeta_1^{\mathcal{A}}$	$\mathcal{K} = Dec^{\mathcal{B}_{pub}}(\zeta_1^{\mathcal{A}})$ $P = \text{GENKEYPRIME}(\,)$ $g = \text{GENKEYPRIME}(\,)$ $\zeta_1^{\mathcal{B}} = Enc^{\mathcal{K}}(P)$ $\zeta_2^{\mathcal{B}} = Enc^{\mathcal{K}}(g)$
$P = Dec^{\mathcal{K}}(\zeta_1^{\mathcal{B}})$ $g = Dec^{\mathcal{K}}(\zeta_2^{\mathcal{B}})$ $a = \text{GENKEYTRNG}(\,)$ $A = g^a \ mod \ P$	$\zeta_1^{\mathcal{B}}, \zeta_2^{\mathcal{B}}$	$b = \text{GENKEYTRNG}(\,)$ $B = g^b \ mod \ P$ $\zeta_3^{\mathcal{B}} = Enc^{\mathcal{K}}(B)$
$B = Dec^{\mathcal{K}}(\zeta_3^{\mathcal{B}})$ $\zeta_2^{\mathcal{A}} = Enc^{\mathcal{K}}(A)$		
$\mathcal{SK} = B^a \ mod \ P$	$\zeta_2^{\mathcal{A}}$	$A = Dec^{\mathcal{K}}(\zeta_2^{\mathcal{A}})$ $\mathcal{SK} = A^b \ mod \ P$

Figure 1 demonstrates the combination of the asymmetric and symmetric key exchange with time-frame. Table 3 demonstrates Diffie-Hellman key exchange protocol in the presence of the attackers. In this enhanced version, the parameters are shared between the sender and receiver by encryption. Conventional Diffie-Hellman shares the parameters publicly. Initially, the sender $\mathcal{A}$ generates a random key using a true-random number generator, let the key be $\mathcal{K}$. The generated key $\mathcal{K}$ is encrypted using the public key of $\mathcal{B}$, let the public key be $\mathcal{B}_{pub}$. The $\mathcal{A}$ sends the generated key $\mathcal{K}$ to the $\mathcal{B}$ using Equation (3).

$$\zeta_1^{\mathcal{A}} = Enc\mathcal{B}_{pub}(\mathcal{K}) \tag{3}$$

Table 3 shows that the attacker $\mathcal{E}$ can get the ciphertext $\zeta_1^{\mathcal{A}}$. The $\mathcal{B}$ receives the ciphertext $\zeta_1^{\mathcal{A}}$ and decrypts the message using its private key; let the private key be $\mathcal{B}_{priv}$ and decryption process is shown in Equation (4).

$$\mathcal{K} = Dec^{\mathcal{B}_{priv}}(\zeta_1^{\mathcal{A}}) \tag{4}$$

The $\mathcal{B}$ needs to generate two prime numbers; let the two prime numbers be the P and g. The two prime numbers are encrypted using shared key $\mathcal{K}$ to send to the $\mathcal{A}$ as shown in Equation (5).

$$\begin{aligned} \zeta_1^{\mathcal{B}} &= Enc^{\mathcal{K}}(P) \\ \zeta_2^{\mathcal{B}} &= Enc^{\mathcal{K}}(g) \end{aligned} \tag{5}$$

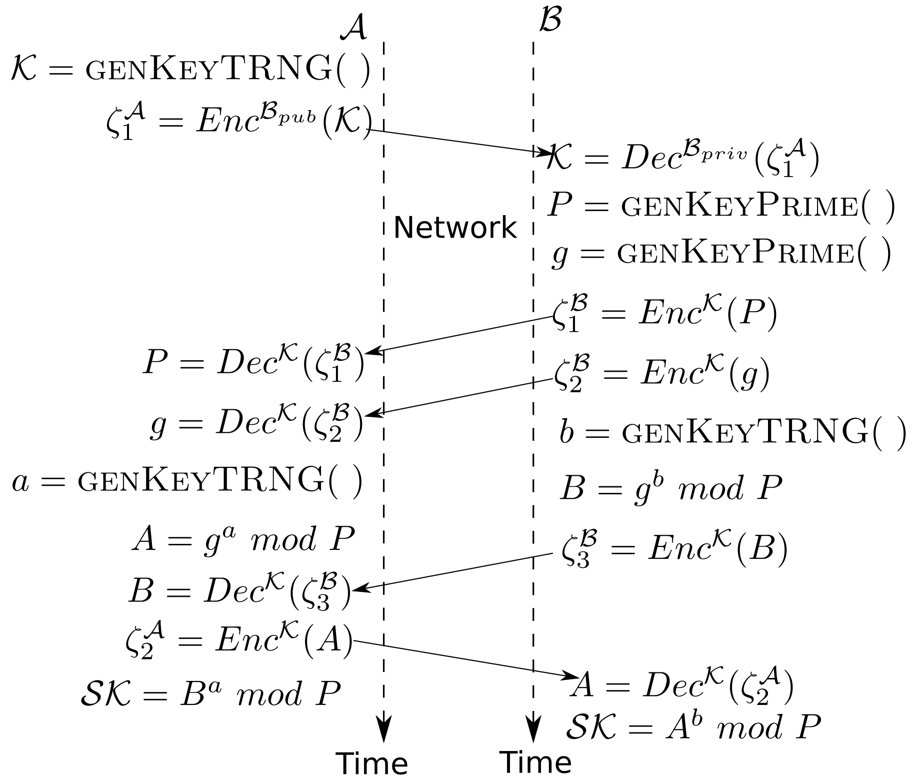
$\mathcal{A}$
$\mathcal{B}$
$\mathcal{K} = \textsc{genKeyTRNG}(\ )$
$\zeta_1^{\mathcal{A}} = Enc^{\mathcal{B}_{pub}}(\mathcal{K})$
$\mathcal{K} = Dec^{\mathcal{B}_{priv}}(\zeta_1^{\mathcal{A}})$
$P = \textsc{genKeyPrime}(\ )$
Network
$g = \textsc{genKeyPrime}(\ )$
$\zeta_1^{\mathcal{B}} = Enc^{\mathcal{K}}(P)$
$P = Dec^{\mathcal{K}}(\zeta_1^{\mathcal{B}})$
$\zeta_2^{\mathcal{B}} = Enc^{\mathcal{K}}(g)$
$g = Dec^{\mathcal{K}}(\zeta_2^{\mathcal{B}})$
$b = \textsc{genKeyTRNG}(\ )$
$a = \textsc{genKeyTRNG}(\ )$
$B = g^b \ mod \ P$
$A = g^a \ mod \ P$
$\zeta_3^{\mathcal{B}} = Enc^{\mathcal{K}}(B)$
$B = Dec^{\mathcal{K}}(\zeta_3^{\mathcal{B}})$
$\zeta_2^{\mathcal{A}} = Enc^{\mathcal{K}}(A)$
$\mathcal{SK} = B^a \ mod \ P$
$A = Dec^{\mathcal{K}}(\zeta_2^{\mathcal{A}})$
$\mathcal{SK} = A^b \ mod \ P$
Time
Time

Fig. 1. Encryption-decryption using asymmetric and symmetric key to enhance Diffie-Hellman algorithm.

Equation (5) uses AES symmetric block cipher algorithm to encrypts the P and g. The $\mathcal{A}$ receives the two ciphertexts and decrypts the two shared prime numbers using Equation (6).

$$\begin{aligned} P &= Enc^{\mathcal{K}}(\zeta_1^{\mathcal{B}}) \\ g &= Enc^{\mathcal{K}}(\zeta_2^{\mathcal{B}}) \end{aligned} \tag{6}$$

Equation (6) uses the AES algorithm to decrypt the ciphertext. Thus, both sender and receiver use the AES algorithm for encryption and decryption using the shared key $\mathcal{K}$ and avoid public-key cryptography. The attacker can get the ciphertexts $\zeta_1^{\mathcal{B}}$ and $\zeta_1^{\mathcal{B}}$. Meanwhile, the $\mathcal{B}$ generates a random number using a true-random number generator, let the number be b. Also, the $\mathcal{B}$ computes B and encrypts it using the shared key shown in Equation (7).

$$\begin{aligned} B &= g^b \ mod \ P \\ \zeta_3^{\mathcal{B}} &= Enc^{\mathcal{K}}(B) \end{aligned} \tag{7}$$

The $\mathcal{B}$ sends the ciphertext $\zeta_3^{\mathcal{B}}$ to $\mathcal{A}$. The attacker $\mathcal{E}$ can get the ciphertext $\zeta_3^{\mathcal{B}}$ from $\mathcal{B}$. In the meantime, the $\mathcal{A}$ computes a random secret number using a true-random number generator; let the random number be a and computes A using Equation Equation (8).

$$\begin{aligned} A &= g^a \ mod \ P \\ \zeta_2^{\mathcal{A}} &= Enc^{\mathcal{K}}(A) \end{aligned} \tag{8}$$

The $\mathcal{A}$ sends the ciphertext $\zeta_2^{\mathcal{A}}$ to $\mathcal{B}$. Similarly, the attacker $\mathcal{E}$ can get the ciphertext $\zeta_2^{\mathcal{A}}$ from $\mathcal{A}$. Simultaneously, the $\mathcal{A}$ and $\mathcal{B}$ receive the ciphertexts $\zeta_3^{\mathcal{B}}$ and $\zeta_2^{\mathcal{A}}$, respectively. The $\mathcal{A}$ decrypts the ciphertext using Equation (9) and compute shared secret key $\mathcal{SK}$.

$$\begin{aligned} B &= Dec^{\mathcal{K}}(\zeta_3^{\mathcal{B}}) \\ \mathcal{SK} &= B^a \ mod \ P \end{aligned} \tag{9}$$

Similarly, the $\mathcal{B}$ decrypts the ciphertext using Equation (10) and compute shared secret key $\mathcal{SK}$.

$$\begin{aligned} A &= Dec^{\mathcal{K}}(\zeta_2^{\mathcal{A}}) \\ \mathcal{SK} &= A^b \ mod \ P \end{aligned} \tag{10}$$

Thus, the sender $\mathcal{A}$ and receiver $\mathcal{B}$ mutual computes shared secret key $\mathcal{SK}$ secretly. Therefore, the Diffie-Hellman does not share any parameters publicly. Now, the $\mathcal{A}$ and $\mathcal{B}$ can share the message by the shared secret key $\mathcal{SK}$ using symmetric key cryptography. The scope of privateDH ends at the beginning of the message sharing between the $\mathcal{A}$ and $\mathcal{B}$. The attack on the symmetric key cryptography (block cipher or stream cipher) for message communication is out of the scope of privateDH.

4 Analysis

privateDH shares a random key initially to compute the shared secret key again. This section analyzes the requirement of the initial shared secret key and the overheads. Moreover, we explore various possible issues.

4.1 Overhead

There is an obvious computation overhead; however, there is no communication overhead. The total number of communication is equal to the conventional Diffie-Hellman key exchange protocol. The computation overhead is clearly visible from Figure 1 and Table 3. The encryption and decryption using public-private keys are not overhead because any conventional secure communication requires establishing the key exchange protocol. In the conventional secure communication setup, the trusted third party distributes the public key. Moreover, the sender retrieves the receiver's public key and sends a secret key by encrypting using the receiver's public key. The receiver sends the two random prime numbers to the sender by encrypting using a secret key instead of an acknowledgment message. Therefore, it reduces one communication overhead than the conventional secure symmetric communication. The computation overhead starts from the encryption and decryption of the g, B, and A as shown in Figure 1. The overheads of the sender are one encryption and two decryption. Similarly, the overheads of the receiver are two encryption and one decryption. However, these overheads are negligible because these are encrypted and decrypted using symmetric key cryptography. Thus, it is much faster than public-key cryptography. Moreover, privateDH is justifiable if security is the top concern.

4.2 Public key

There are diverse issues in the public key encryption, and under this circumstance, the public key may be broken at any time. Therefore, we are assuming that there is a quantum attack that can easily break a public key within eight hours [19]. In this assumption, the attacker can retrieve all the information excepts a, b, and $\mathcal{SK}$, which becomes the conventional Diffie-Hellman algorithm. Let us assume that the public key is not broken. In this assumption, the security becomes tighter than the conventional Diffie-Hellman algorithm. The attacker will not be able to get the secret key generated by the sender. Thus, privateDH ensures security even if the public key is broken. The public key cryptography is slow, and thus, the sender performs encryption only once, and the receiver performs decryption only once.

4.3 Symmetric key

The symmetric key requires a shared secret key to encrypt and decrypt. privateDH shares the secret key $\mathcal{K}$ encrypting using the receiver's public key, and all Diffie-Hellman communications are performed using symmetric-key cryptography. For instance, the receiver shares the prime numbers P and g to the sender by encrypting these prime numbers using $\mathcal{K}$. Moreover, the sender and receiver encrypt using $\mathcal{K}$ for further communications. Let us assume that the attacker is able to break the security of symmetric cryptography by applying cryptanalysis. In this assumption, all parameters of privateDH are compromised, and thus, it becomes the conventional Diffie-Hellman algorithm. Therefore, privateDH ensures security even if the symmetric key cryptography is broken. The attacker will not get a and $\mathcal{SK}$, and thus, privateDH is still secure.

4.4 Security measurement

Theorem 1. *privateDH reports a man-in-the-middle attack to the intended user if the attacker breaks the public key.*

Proof. The $\mathcal{A}$ sends ciphertext $\zeta_1^{\mathcal{A}}$ to the $\mathcal{B}$. The sender expects to receive the $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ from $\mathcal{A}$. The attacker $\mathcal{E}$ is able to break the public-key cryptography, and thus, the $\mathcal{E}$ returns $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ to $\mathcal{A}$ before being sent by the $\mathcal{B}$. Because the attacker $\mathcal{E}$ have already the private key of the corresponding public key. The $\mathcal{A}$ proceeds with the returns value of the $\mathcal{E}$. Meanwhile, the $\mathcal{B}$ also sends $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ to $\mathcal{A}$. The $\mathcal{A}$ receives two copies of the same ciphertexts $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ from different sources. In this case, the $\mathcal{A}$ cannot differentiate between the attacker and the intended user. Therefore, the $\mathcal{A}$ unable to process and privateDH fails. Thus, privateDH reports to both the attacker and intended user about the failure of the communication. □

The public key can be broken, but it takes many CPU Core years. Therefore, the attacker can send back the ciphertext $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ to $\mathcal{A}$ with certain time gaps. The $\mathcal{A}$ and $\mathcal{B}$ can complete the communication within that time gap. It might happen that the $\mathcal{A}$ and $\mathcal{E}$ have communicated the message successfully, and then, the $\zeta_1^{\mathcal{B}}$ and $\zeta_2^{\mathcal{B}}$ have arrived from the original source $\mathcal{B}$. In this case, the communications have already over. Therefore, nothing can be undone. However, the $\mathcal{A}$ reports the incident to both the $\mathcal{B}$ and $\mathcal{E}$ such that the $\mathcal{B}$ can rectify the issue. However, it is an assumed incident.

4.5 Attacks

Diverse attacks are carried out to defeat cryptography. Gidney and Ekerå [19] demonstrates that public keys can be broken in eight hours using quantum computing. Therefore, the public key can be compromised, and the attacker can store the precomputed private keys of corresponding public keys, but privateDH still ensures security. Similarly, symmetric-key cryptography is prone to various attacks, namely, brute-force, cryptanalysis [23], dictionary and fault attacks [24]. privateDH can ensure security if the symmetric key is broken using quantum cryptanalysis [23] or any other methods [24].

Let us assume that an attacker is able to get A and B; then the attacker still has not broken the security of privateDH. Therefore, the attacker should attack the shared secret key $\mathcal{SK}$. The probability of getting a shared secret key is $\frac{1}{2^{\beta}}$ using the brute-force method where β is the bit length of the key. It is hard to get the shared secret key.

5 Conclusion

This article demonstrates the private Diffie-Hellman algorithm, privateDH for short, derived from the conventional Diffie-Hellman algorithm. The conventional Diffie-Hellman algorithm shares the numbers publicly in the insecure channel. Therefore, it is possible to attack the Diffie-Hellman algorithm, for instance, Logjam [27]. Therefore, securing key exchange protocol becomes a key challenge. Hence, privateDH provides security to the Diffie-Hellman algorithm by using a combination of RSA and AES. We have demonstrated how the privateDH algorithm uses RSA and AES to secure the Diffie-Hellman algorithm with a few computation overheads. privateDH does not have any communication overhead. Moreover, privateDH shows its strong resistance against any attacks. privateDH can ensure high security even if the attackers compromise the public key. Furthermore, it also ensures high security even if the attacker is able to break symmetric key cryptography. Therefore, privateDH can compute the secret key securely between two parties.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, Feb. 1978.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "Cryptographic communications system and method," Sep. 20 1983, uS Patent 4,405,829.

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

[4] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology — CRYPTO '85 Proceedings*, H. C. Williams, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426.

[5] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[6] R. for Pair-Wise Key Establishment Schemes UsingDiscrete Logarithm Cryptography, "Elaine barker and lily chen and allen roginsky and miles smid," Accessed on January 2021 from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-56ar.pdf, 2007.

[7] "Specification for the advanced encryption standard (aes)," Federal Information Processing Standards Publication 197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] D. Aggarwal and U. Maurer, "Breaking RSA Generically Is Equivalent to Factoring," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6251–6259, Nov. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7523212/

[9] C. P. Schnorr, "Fast factoring integers by svp algorithms," Cryptology ePrint Archive, Report 2021/232, 2021, https://eprint.iacr.org/2021/232.

[10] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-exponent rsa with related messages," ser. EUROCRYPT'96. Berlin, Heidelberg: Springer-Verlag, 1996, p. 1–9.

[11] M. Wiener, "Cryptanalysis of short rsa secret exponents," *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 553–558, 1990.

[12] J. Hastad, "N using rsa with low exponent in a public key network," in *Advances in Cryptology — CRYPTO '85 Proceedings*, H. C. Williams, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 403–408.

[13] A. M. Ahmadian and M. Amirmazlaghani, "A novel secret image sharing with steganography scheme utilizing Optimal Asymmetric Encryption Padding and Information Dispersal Algorithms," *Signal Processing: Image Communication*, vol. 74, pp. 78–88, May 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0923596518307938

[14] A. Nitaj, M. R. K. Ariffin, D. I. Nassr, and H. M. Bahig, "New attacks on the rsa cryptosystem," in *International conference on cryptology in Africa*. Springer, 2014, pp. 178–198.

[15] M. J. Hinek, M. K. Low, and E. Teske, *On Some Attacks on Multi-prime RSA*. Springer Berlin Heidelberg, 2003, p. 385–404. [Online]. Available: http://dx.doi.org/10.1007/3-540-36492-7_25

[16] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, *Factorization of a 768-Bit RSA Modulus*. Springer Berlin Heidelberg, 2010, p. 333–350.

[17] E. Thomé, "[Cado-nfs-discuss] 795-bit factoring and discrete logarithms," Dec 2019, [Online; Accessed on February 2021]. [Online]. Available: https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2019-December/001139.html

[18] P. Zimmermann, "[Cado-nfs-discuss] Factorization of RSA-250," Feb 2020, [Online; accessed on Mars 2021]. [Online]. Available: https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2020-February/001166.html

[19] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, Apr. 2021. [Online]. Available: https://doi.org/10.22331/q-2021-04-15-433

[20] O. F. A. Wahab, A. A. M. Khalaf, A. I. Hussein, and H. F. A. Hamed, "Hiding data using efficient combination of rsa cryptography, and compression steganography techniques," *IEEE Access*, vol. 9, pp. 31 805–31 815, 2021.

[21] S. B. Das, S. K. Mishra, and A. K. Sahu, "A New Modified Version of Standard RSA Cryptography Algorithm," in *Smart Computing Paradigms: New Progresses and Challenges*. Singapore: Springer, Dec 2019, pp. 281–287.

[22] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 1, pp. 71–81, 2021.

[23] S. Jaques and J. M. Schanck, "Quantum cryptanalysis in the ram model: Claw-finding attacks on sike," in *Advances in Cryptology – CRYPTO 2019*, A. Boldyreva and D. Micciancio, Eds. Cham: Springer International Publishing, 2019, pp. 32–61.

[24] A. Baksi, S. Bhasin, J. Breier, D. Jap, and D. Saha, "Fault attacks in symmetric key cryptosystems," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1267, 2020. [Online]. Available: https://eprint.iacr.org/2020/1267

[25] C. Guo, X. Chen, Y. Jie, Z. Fu, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Transactions on Services Computing*, vol. 13, no. 6, pp. 1034–1044, 2020.

[26] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: A survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 978–996, 2018.

[27] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann, "Imperfect forward secrecy: How diffie-hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 5–17. [Online]. Available: https://doi.org/10.1145/2810103.2813707

[28] J. Shen, T. Miao, J.-F. Lai, X. Chen, J. Li, and S. Yu, "Ims: An identity-based many-to-many subscription scheme with efficient key management for wireless broadcast systems," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[29] H. Jiang, D. Belkin, S. E. Savel'ev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, M. Rao, M. Barnell, Q. Wu, J. J. Yang, and Q. Xia, "A novel true random number generator based on a stochastic diffusive memristor," *Nat. Commun.*, vol. 8, no. 882, pp. 1–9, Oct 2017.

[30] İ. Koyuncu, M. Tuna, İ. Pehlivan, C. B. Fidan, and M. Alçın, "Design, FPGA implementation and statistical analysis of chaos-ring based dual entropy core true random number generator," *Analog Integr. Circ. Sig. Process.*, vol. 102, no. 2, pp. 445–456, Feb 2020.

[31] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES Is in P," *Ann. Of Math.*, vol. 160, no. 2, pp. 781–793, Sep 2004. [Online]. Available: http://www.jstor.org/stable/3597229

Ripon Patgiri (ripon@cse.nits.ac.in) is currently working as an Assistant Professor in the Department of Computer Science & Engineering, National Institute of Technology Silchar, Assam-788010, India. He receives his Master Degree and PhD degree from Indian Institute of Technology Guwahati and National Institute of Technology Silchar, respectively. His research interests are Bloom Filter, Network Security, Privacy, and Networking. He has published several research papers in reputed journals, conferences and books. Also, he has published several edited books. He is a senior member of IEEE, member of EAI, associate member of IETE, and life member of ACCS. He serves as a General Chair, Program Chair and Organizing Chair of several conferences. URL: http://cs.nits.ac.in/rp