Privacy Preserving Two-Server Diffie-Hellman Key Exchange Protocol

Durbadal Chattaraj and Monalisa Sarma*
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
dchattaraj@iitkgp.ac.in,monalisa@iitkgp.ac.in

Debasis Samanta†
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal 721302, India
dsamanta@sit.iitkgp.ernet.in

ABSTRACT

For a secure communication over an insecure channel the Diffie-Hellman key exchange protocol (DHKEP) is treated as the de facto standard. However, it suffers form server-side compromisation, identity compromisation, man-in-the-middle, replay attacks, etc. Also, there are single point of vulnerability (SOV), single point of failure (SOF) and user privacy preservation issues. This work proposes an identity-based two-server DHKEP to address the aforesaid issues and alleviating the attacks. To preserve user identity from outside intruders, a k-anonymity based identity hiding principle has been adopted. Further, to ensure efficient utilization of channel bandwidth, the proposed scheme employs elliptic curve cryptography. The security analysis substantiate that our scheme is provably secure and successfully addressed the above-mentioned issues. The performance study contemplates that the overhead of the protocol is reasonable and comparable with other schemes.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*;

KEYWORDS

Security and privacy, Security services, Privacy-preserving protocols

ACM Reference format:
Durbadal Chattaraj and Monalisa Sarma and Debasis Samanta. 2017. Privacy Preserving Two-Server Diffie-Hellman Key Exchange Protocol. In *Proceedings of SIN '17, Jaipur, IN, India, October 13–15, 2017,* 8 pages.
https://doi.org/10.1145/3136825.3136871

1 INTRODUCTION

The Next-generation network [14] demands secure ubiquitous service delivery over insecure channel. In this synergy, establishment of a secure communication channel between end user and remote service server is necessary. A de facto DHKEP [5] is widely used for this purpose. In this regard, both the intended sender and receiver establish a secret key between themselves based on previously shared common domain parameters (also called public parameters). However, this protocol does not have any mechanism to accomplish a robust mutual authentication between sender and receiver. As a result, an eavesdropper can easily makes a man-in-the-middle attacks by listening the communication traffic of both the intended parties. In addition to this, identity compromisation attacks, impersonation attacks, replay attacks, server-side compromisation, etc. are the major security threats that are not properly addressed [13].

To alleviate these issues, several modifications of DHKEP are reported in the recent literatures [2, 4, 7, 10–13, 17, 19, 20] where, both the intended parties should verify their identities before the shared secret symmetric key is established between themselves. However, without loss of generality, these existing approaches do not properly address the user privacy, anonymity support, protection from malicious insiders (i.e., passive attacks on server-side [3]), server-side compromisation attacks, single point of vulnerability issue [18], etc. It may also be noted that efficient utilization of communication bandwidth is a key factor for message exchanges. The existing DHKEPs do not fully resolve this issue. Therefore, there is a need to propose a viable solution to address these issues.

This paper aims to address the aforesaid limitations of the existing DKKEPs. In this regard, we finalize the following objectives. (1) To overcome existing security threats namely man-in-the-middle (MITM), replay and server-side compromisation attacks. (2) To provide better server-side security, and user privacy with k-anonymity (i.e., the k^{th} user identity cannot be distinguished from at least $(k-1)$ individuals [15] identities that are available in the protocol transcripts) support. (3) To make mutual authentication more robust and avoiding the impersonation or identity compromisation attack. (4) To provide a better solution for single point of vulnerability issue.

Our proposed approach vis-a-vis the above-mentioned objectives are as follows. A PKI-enabled identity-based two-server key exchange protocol has been put forward. Our dual server model is organized in such a way that it is resilient against existing vulnerabilities namely MITM attacks, server-side compromisation attacks, replay attacks, etc. In this two-server model, the server, which is at the front-end is responsible for interfacing with both client and service

*D. Chattaraj and M. Sarma are associated with Subir Chowdhury School of Quality and Reliability, IIT Kharagpur, India, Contact no.: +91-3222-283998.
†D. Samanta is assosiated with Computer Science and Engineering, IIT Kharagpur, India, Contact no.: +91-3222-282334.

SIN '17, October 13–15, 2017, Jaipur, IN, India

ACM ISBN 978-1-4503-5303-8/17/10…$15.00
https://doi.org/10.1145/3136825.3136871

server, and the other server at the back-end accomplishes the mutual authentication task. As the back-end server is hidden from public exposure, it ensures the server-side security. It may be noted that, the back-end server manages all the security credentials for both the client and the service server where as the front-end server only allows to stores necessary messages in encrypted format. As a result, it gives a solution for both single point of vulnerability issue and server-side compromisation attacks. To ensure user privacy from external attackers, a "k-anonymity" based identity hiding strategy has been put forward. In addition to this, as a solution to the impersonation or identity compromisation attacks, we propose an elliptic curve cryptography (ECC) based identity encapsulation scheme. Our proposed ECC based key exchange protocol ensures efficient utilization of channel bandwidth in terms of bits.

Following are the major research contributions, which we have accomplished as a realization of our proposed approach. (1) Our proposed protocol alleviates all the vulnerabilities such as MITM, identity compromisation, server-side compromisation and replay attacks. (2) Existing DHKEPs suffer from the server-side compromisation [10]; that is, once the trusted third party server is compromised; the overall system will be jeopardized. In our proposed protocol, such a situation can be avoided. (3) We have proposed a novel identity-based verification technique by which each entity would be able to substantiate the other entity along with the issuer of the security credentials on which both the entities rely. As a result, it gives a novel solution to preserve both server-side and user privacy. (4) Our proposed scheme provides a novel solution to preserve user privacy achieving k-anonymity.

The organization of the paper is as follows. Section 2 discuss the basic concepts of elliptic curve cryptography and its underlying security assumptions. Section 3 presents our proposed protocol. We discuss security analysis in Section 4. Section 5, we formally verify our proposed protocol using AVISPA tool [1]. We evaluate performance of our proposed scheme in Section 6. Finally, Section 7 concludes the paper.

2 PRELIMINARIES

In this section, we discuss elliptic curve cryptography (ECC) and its two underlying security assumptions namely elliptic curve computational Diffie-Hellman problem (ECCDHP) and elliptic curve discrete logarithm problem (ECDLP).

(1) Elliptic Curve. An elliptic curve E over the real field F_p with genus 1 and a special point $\mathcal{O}$ called the point at infinity is defined by an equation of the form $y^2 = x^3 + mx + n \bmod p$, where $p > 3$ is a odd prime and $m, n \in F_p$, and discriminant $\Delta = 4m^3 + 27n^2 \neq 0 \bmod p$. The set $E(F_p)$ consists of all points $(x, y) \in F_p \times F_p$ with a special point $\mathcal{O}$. [18]

The algebraic formulas for the sum of two same or different points on $y^2 = x^3 + mx + n \bmod p$ as follows:

(1) $V + \mathcal{O} = \mathcal{O} + V = V$ for all $V \in E(F_p)$.
(2) If $V = (x, y) \in E(F_k)$, then $(x, y) + (x, -y) = \mathcal{O}$. (The point $(x, -y)$ is denoted by $-V$).
(3) Let $V \neq W$, where $V = (x_1, y_1) \in E(F_p)$ and $W = (x_2, y_2) \in E(F_p)$ with $x_1 \neq x_2$, where $V \neq \pm W$. Then $V + W = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2 \bmod p$ and $y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$. Here, $\lambda = (\frac{y_2 - y_1}{x_2 - x_1})$
(4) If $x_1 = x_2$, but $y_1 \neq y_2$, then $V_1 + V_2 = \mathcal{O}$.
(5) Let $V = (x_1, y_1) \in E(F_p)$ with $y_1 \neq 0$, where $V \neq -V$. Then $2V = (x_3, y_3)$, where $x_3 = \lambda^2 - 2x_1 \bmod p$ and $y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$. Here, $\lambda = (\frac{3x_1^2 + m}{2y_1})$
(6) Let $V = (x_1, y_1) \in E(F_p)$ with $y_1 = 0$. Then $V_1 + V_2 = \mathcal{O}$. ■

(2) ECCDHP Assumption. Given a point V on an elliptic curve over a finite field of large order and two other points $x \cdot V$, $y \cdot V$, find $x \cdot y \cdot V$ is computationally hard.

(3) ECDLP Assumption. Given an elliptic curve E defined over F_k and two points $V, W \in E(F_k)$, find an integer x such that $W = x \cdot V$ is computationally intractable.

3 PROPOSED PROTOCOL

In this section, we present our proposed identity-based two-server Diffie-Hellman key exchange protocol. The system architecture of our proposed protocol is shown in Fig. 1. In our discussion, we frequently refer to some notations and symbols. All notations and symbols with their annotations are listed in Tab. 1.

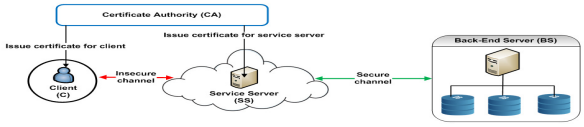

Figure 1: System architecture of proposed protocol.

3.1 System model

Three types of entities are involved in the proposed system namely client (C), service server (SS), back-end server (BS), where SS is the public server in two-server model and BS is the private server. The public server is reachable to everyone whereas the private server works in the background, and it is controlled internally by the administrator only. It may be noted that, there exists a secure channel between SS and BS. C and SS both enroll themselves with BS during registration phase, but the authenticated key exchange task is carried out by both SS and BS, respectively. Further, there exists a certificate authority (CA) which issues a certificate CR_E for each entity E except BS. The CA selects a generator G on the elliptic curve $E(F_p)$ of order k, and also selects two hash functions $\mathcal{H}_1(\cdot)$ and $\mathcal{H}_2(\cdot)$. Further, the entity E can randomly selects $s_E \in \mathbb{Z}_k^*$ as the secret key and computes the corresponding public key as $Q_E = s_E \cdot G$. Entity E needs to keep s_E secret but distributes Q_E using CR_E. Suppose, there exists an application B_A provided by BS with which C can transfer his user identity securely to BS at the time of user registration (this is one time activity). BS then registers the user identity in its secure database and sends a reference

Table 1: Notations and their meanings.

Notation	Description
A_i	An application running on user C's workstation to access service server
BS	Back End Authentication Server
BS_{ID}	BS's public identity
C	Client
CA	Certificate Authority
C_{ID}	Public identity of C
C_{ID}^{A}	Anonymous identity of C where $R_1 \leq C_{ID}^{A} \geq R_2$
C_{id}^{reg}	C's transformed identity
CR_E	An X.509 certificate issued by CA for each entity E
E	An entity like C and SS
$E(F_p)$	Elliptic curve on the field F_p over modulo p
$E(X:[M])$	Message M encrypts using key X
F_k	A field containing the elements $0, 1, \cdots, k-1$
G	A generator point $\in E(F_p)$ of some order (say, k)
hash/$h(\cdot)$	One way hash function such as SHA-1, SHA-2 etc.
i	An intruder
k	A 160 to 256-bit prime
k_bs/$K_{b,ss}$	A shared secret key between SS and BS
L	The size of a group element [19]
l	The size of the hash value [19]
Q_c/Q_c	C's public key
Q_{ss}/Q_ss	SS's public key
Q_i	i's public key
SS	Service Server
text_set_s/SS_{ID}	Public identity of SS
S_{secret}	Secret identity of SS
SS_{tid}	Transformed identity of SS
s_{ss}	SS's private key
s_c	C's private key
text_set_L_1	A text set consists of $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$
text_set_i	i's knowledge about C_{tid}
R_1–R_2	A range of random numbers where $R_1, R_2 \in \mathbb{Z}$, $R_2 > R_1$
r_cb/$r_{c,b}$	A secret reference number shared between C and BS
$\mathbb{U}$	A set of all points on the elliptic curve E_k
$\mathcal{H}_1(\cdot)$	One way cryptographic hash function as $\{0,1\}^* \rightarrow \mathbb{U}$ [8]
$\mathcal{H}_2(\cdot)$	One way cryptographic hash function as $\mathbb{U} \rightarrow \{0,1\}^*$ [8]
text_set_u/C_{tid}	C's transformed user id
$\mathbb{Z}_k$	A set containing the elements $0, 1, \cdots, k-1$
(x/y)	An entity computes x number of exponentiations in real time and y number of exponentiations in offline mode
$\|q\|$	Bit length of q
$a \hookleftarrow b$	The value of a has been taken from b

number to C. C needs to keep the reference number secret with himself. It may be noted that, the enrollments of SS are carried out by both SS's and BS's administrator at the time of system deployment where, a secret identity is shared between themselves. However, at the time of authenticated key exchange, C verifies the legitimacy of both SS and BS, and SS checks the legitimacy of both C and BS without disclosing their secret identities over the insecure channel. In addition to this, a client application A_i is running in client workstation in order to access the SSs.

3.2 Registration

Initially, both C and SS need to register themselves with BS. Suppose, C wants to enroll his secret credentials with BS. Then the detail enrollment steps is as follows:

Step 1: C sends his identity (C_{ID}) online securely to BS using B_A and goto Step 2. It may be noted that, this is a one time operation.

Step 2: After receiving C_{ID}, BS takes a system generated reference number (i.e., $r_{c,b}$) and goto Step 3.

Step 3: BS encrypts the C_{ID} using $r_{c,b}$, and applies $h(\cdot)$ on it as $C_{reg}^{id} = h(E(r_{c,b} : [C_{ID}]))$, and goto Step 4.

Step 4: BS creates C's account and updates user table with C_{reg}^{id} and $r_{c,b}$ respectively.

Step 5: BS securely sends the reference number to C and deletes the actual client identity (i.e., C_{ID}) immediately.

It may also note that, SS's registration into BS has been carried out offline before deployment of our proposed protocol. In this enrollment process, a secret identity of SS (i.e., S_{secret}) has been shared with BS securely. Hence, completes both C and SS registration process. After completion of registration, C can access services from SS followed by a mutual authentication and key exchange task as follows.

3.3 Mutual authentication and key exchange

In this section, we discuss mutual authentication and key exchange process of our proposed scheme. A diagram presenting the several communications among different entities involved throughout the authenticated key exchange process is shown in Fig. 2. The detail step in this process is as follows.

Step 1: C enters C_{ID}, $r_{c,b}$ and SS_{ID} into application A_i.

Step 2: After getting C's response, A_i generates a random number $C_2 = C_{ID}^{A}$ and computes $C_{tid} = h(C_1||C_2)$ where $C_1 = E(r_{c,b} : [C_{ID}])$. It also computes a signature pair (i.e., $\{\mathcal{S}_1, \mathcal{S}_2\}$) for the same C_{tid} using clientSigGen[ ]() (see Fig. 3).

Step 3: A_i constructs a message as $m_1 = \{C_1, C_2, SS_{ID}, \mathcal{S}_1, \mathcal{S}_2, CR_C\}$ and sends it to SS.

Step 6: After receiving m_1, SS constructs a message $m_2 = \{C_2, C_1, SS_{ID}, B_{ID}, \mathcal{S}_1\}$ and sends the same to BS.

Step 7: BS checks both registered client identity ($C_1 \stackrel{?}{=} C_{reg}^{id}$) and server identity ($SS_{reg}^{id} \stackrel{?}{=} SS_{ID}$) respectively from its database.

Step 8: If both the identities exist then, BS computes two transformed identities as $C_{tid}^{'} = h(C_{reg}^{id}||C_2)$ and $SS_{tid} = h(SS_{reg}^{id}||S_{secret})$, respectively (see encryptIdnt[ ]() in Fig. 4). BS then encrypts both these identities as $\mathcal{M}_1 = E(K_{b,ss} : [C_{tid}^{'}])$ and $\mathcal{M}_2 = E(r_{c,b} : [SS_{tid}])$, and goto Step 9, else rejects SS's request.

Step 9: BS constructs a message as $m_3 = \{B_{ID}, C_2, SS_{ID}, \mathcal{S}_1, \mathcal{M}_1, \mathcal{M}_2\}$ and sends the same to SS.

Step 10: After getting reply from BS, SS verifies the legitimacy of both C and BS using verifyClientSig()(see Fig. 5) and goto Step 11. In this connection, SS decrypts $\mathcal{M}_1$ and gets the transformed client identity (i.e., $C^{'}_{tid}$). After getting the identity, SS checks the following condition:

$$\begin{aligned} \mathcal{S}_2 \cdot G \stackrel{?}{=} r_c \cdot \mathcal{H}_2(\mathcal{H}_1(C^{'}_{tid})) \cdot G + \\ s_c \cdot \mathcal{H}_2(\mathcal{S}_1) \cdot G \stackrel{?}{=} \mathcal{S}_1 \cdot \mathcal{H}_2(\mathcal{H}_1(C^{'}_{tid})) + \\ Q_c \cdot \mathcal{H}_2(\mathcal{S}_1) \end{aligned} \quad (1)$$

where, $\mathcal{S}_1 = r_c \cdot G$ and $Q_c = s_c \cdot G$. Intuitively, satisfying the condition in Equ. 1 proofs the legitimacy of both BS and C respectively to SS.

Step 11: SS concurrently constructs a session key ($SK = r_{ss} \cdot \mathcal{S}_1 = r_{ss} \cdot r_c \cdot G$ (see verifyClientSig() in Fig. 5 and serverSigGen[ ]() in Fig. 6), and a message $m_4 = \{C_2, SS_{ID}, \mathcal{S}_1, \mathcal{S}_3, \mathcal{S}_4, \mathcal{M}_2, CR_{SS}\}$ (see Fig. 2), and goto Step 12, else rejects C's request.

Step 12: SS sends the message m_4 to C.

Step 13: After receiving m_4, C checks the legitimacy of both SS and BS (see verifyServerSig() in Fig. 7) by applying the same analogy as discussed for SS.

Step 14: If the legitimacy of both the servers (SS and BS) are verified successfully, then C constructs a session key as $SK = r_c \cdot r_{ss} \cdot G$.

Finally, C and SS establish a secure channel between themselves and set up a shared secret symmetric key as $SK = r_c \cdot r_{ss} \cdot G$. It may be noted that, after establishment of session key, C uses the anonymous identity (i.e., k-anonymous) C_2 where as the transformed identity C_1 is being forwarded to SS via a secure channel.

function clientSigGen[ ]$(C_{ID}, r_{c,b})$
{
Input: client identity and C-BS's secret ref. no.
Output: Two nonnegative random numbers as $\mathcal{S}_1$ and $\mathcal{S}_2$
Data: CR_C, s_c
1. Choose a random number C^A_{ID} such that $R_1 \leq C^A_{ID} \geq R_2$
2. Compute $C_{tid} = h(C_1||C_2)$
where $C_1 = E(r_{c,b} : [C_{ID}])$ and $C_2 = C^A_{ID}$
3. Choose a pseudo-random number $r_c \in \mathbb{Z}^*_k$
4. Compute $\mathcal{S}_1 = r_c \cdot G$
5. Compute $\mathcal{S}_2 = r_c \cdot \mathcal{H}_2(\mathcal{H}_1(C_{tid})) + s_c \cdot \mathcal{H}_2(\mathcal{S}_1) \bmod k$
6. return $\mathcal{S}_1$, $\mathcal{S}_2$
}

Figure 3: Client-side signature generation process.

3.3.1 Correctness. In order to verify the legitimacy of both C and SS along with back-end third party server (BS), C/SS must adheres the following two conditions.

Condition 1: For the purpose of verifying C and BS, SS needs to check

$$\mathcal{S}_2 \cdot G \stackrel{?}{=} \mathcal{S}_1 \cdot \mathcal{H}_2(\mathcal{H}_1(C^{'}_{tid})) + Q_c \cdot \mathcal{H}_2(\mathcal{S}_1). \quad (2)$$

To make the checks work, it must holds

$$\mathcal{S}_2 = r_c \cdot \mathcal{H}_2(\mathcal{H}_1(C_{tid})) + s_c \cdot \mathcal{H}_2(r_c \cdot G) \bmod k \quad (3)$$

function encryptIdnt[ ]$(SS_{ID}, C_2, C_1, \mathcal{S}_1)$
{
Input: SS's identity, C's anonymous identity, C's transformed identity, C's signature
Output: Two cipher-text pair as $\mathcal{M}_1$ and $\mathcal{M}_2$
Data: $SS^{id}_{reg} := SS_{ID}, r_{c,b}, K_{b,ss}, S_{secret}, C^{id}_{reg} := h(E(r_{c,b} : [C_{ID}]))$
if($C^{id}_{reg} = C_1$ & $SS^{id}_{reg} = SS_{ID}$ & $\mathcal{S}_1$ is fresh[a])
{
1. Compute $C^{'}_{tid} = h(C^{id}_{reg}||C_2)$
2. Compute $SS^{'}_{tid} = h(SS^{id}_{reg}||S_{secret})$
3. Compute $\mathcal{M}_1 = E(K_{b,ss} : [C^{'}_{tid}])$
4. Compute $\mathcal{M}_2 = E(r_{c,b} : [SS^{'}_{tid}])$
5. return $\mathcal{M}_1, \mathcal{M}_2$
}
else
{
6. return null; (//reject SS request//)
}
}
[a]an unique bit string for a single protocol run

Figure 4: Identity transformation process at BS.

function verifyClientSig $(\mathcal{M}_1, \mathcal{S}_1, \mathcal{S}_2, CR_C)$
{
Input: encrypted C's transformed identity, a pair of C's signatures and C's certificate
Output: return true or false
Data: $K_{b,ss}, SS_{ID}, S_{secret}, SS_{tid} := h(SS_{ID}||S_{secret})$
1. Decrypt $\mathcal{M}_1$ and get $C^{'}_{tid}$
2. Compute $temp := \mathcal{S}_2 \cdot G$
3. Compute $temp_1 := \mathcal{S}_1 \cdot \mathcal{H}_2(\mathcal{H}_1(C^{'}_{tid})) + Q_c \cdot \mathcal{H}_2(\mathcal{S}_1)$
where $\mathcal{S}_1 = r_c \cdot G$ and $Q_c = s_c \cdot G \hookleftarrow CR_C$
if($temp = temp_1$)
{
4. $var[]$=function serverSigGen[ ](SS_{ID}, SS_{tid})
5. Compute $SK := var[0] \cdot \mathcal{S}_1 = var[0] \cdot r_c \cdot G$
where $var[0] = r_{ss}$
6. return true (//accept C and BS//)
}
else
{
7. return false (//reject C and BS//)
}
}

Figure 5: Verification process at Service Server.

function serverSigGen[ ](SS_{ID}, SS_{tid})
{
Input: SS's identity and SS-BS shared secret identity
Output: Two nonnegative random numbers as $\mathcal{S}_3$ and $\mathcal{S}_4$
Data: CR_{SS}, s_{ss}
1. Choose a random number $r_{ss} \in \mathbb{Z}^*_k$
4. Compute $\mathcal{S}_3 = r_{ss} \cdot G$
5. Compute $\mathcal{S}_4 = r_{ss} \cdot \mathcal{H}_2(\mathcal{H}_1(SS_{tid})) + s_{ss} \cdot \mathcal{H}_2(\mathcal{S}_3) \bmod k$
6. return r_{ss}, $\mathcal{S}_3$, $\mathcal{S}_4$
}

Figure 6: SS-side signature generation process.

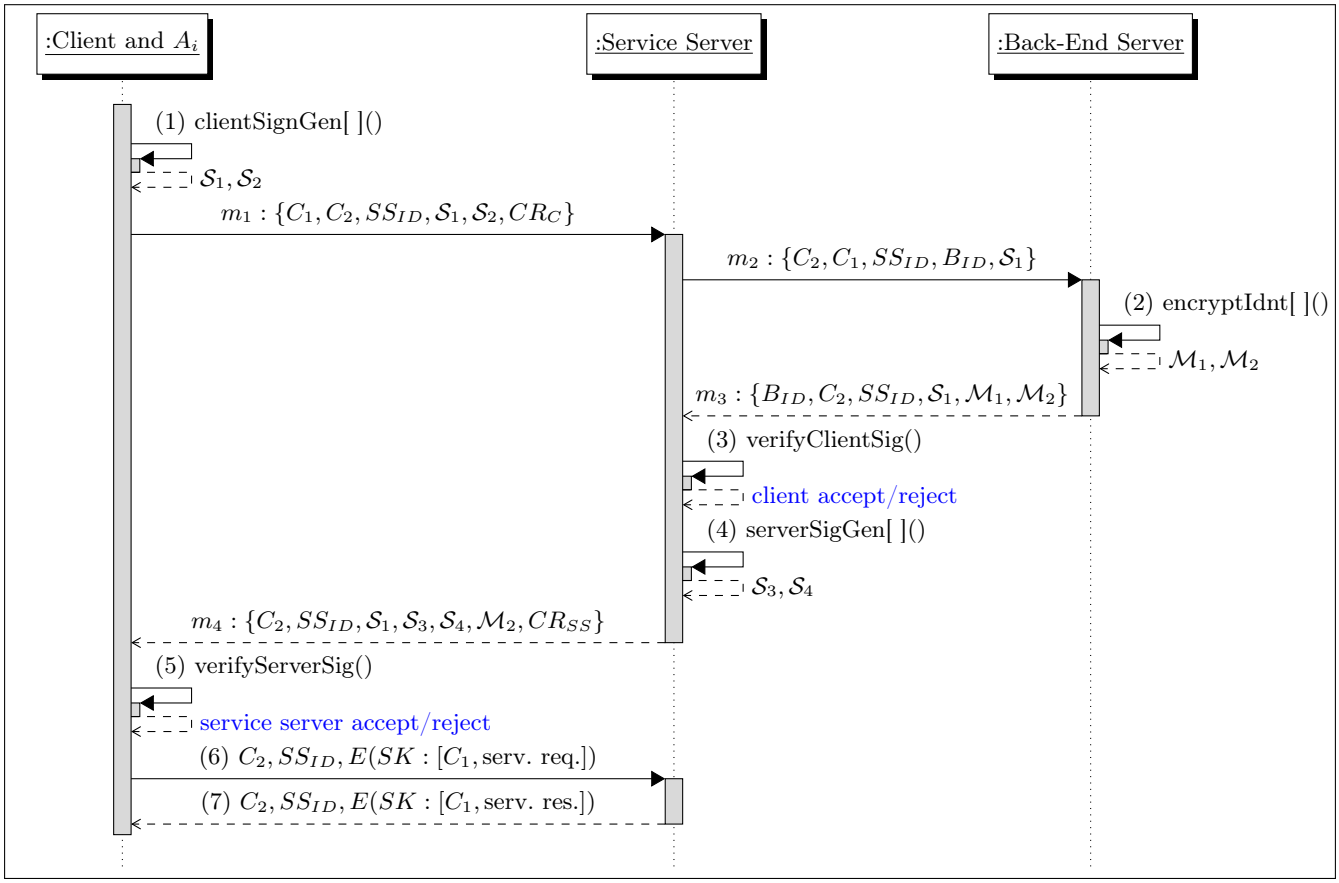

Figure 2: Mutual authentication and key exchange process.

function verifyServerSig $(\mathcal{M}_2, \mathcal{S}_3, \mathcal{S}_4, CR_{SS})$
{
Input: encrypted SS's secret identity, a pair of SS's signatures and SS's certificate
Output: return true or false
Data: $K_{c,b}, C_{ID}, r_c$
1. Decrypt $\mathcal{M}_2$ and get $SS^{'}_{tid}$
2. Compute $temp_2 := \mathcal{S}_4 \cdot G$
3. Compute $temp_3 := \mathcal{S}_3 \cdot \mathcal{H}_2(\mathcal{H}_1(SS^{'}_{tid})) + Q_{ss} \cdot \mathcal{H}_2(\mathcal{S}_3)$
where $\mathcal{S}_3 \hookleftarrow r_{ss} \cdot G$ and $Q_{ss} \hookleftarrow s_{ss} \cdot G \hookleftarrow CR_{SS}$
if($temp_2 = temp_3$)
{
4. Compute $SK := r_c \cdot \mathcal{S}_3 = r_c \cdot r_{ss} \cdot G$
5. return true (//accept SS and BS//)
}
else
{
7. return false (//reject SS and BS//)
}
}

Figure 7: Verification process at client-side.

and

$$C^{'}_{tid} = C_{tid}. \tag{4}$$

Condition 2: For the purpose of checking the legitimacy of both SS and BS, C needs to verify

$$\mathcal{S}_4 \cdot G \stackrel{?}{=} \mathcal{S}_3 \cdot \mathcal{H}_2(\mathcal{H}_1(SS^{'}_{tid})) + Q_{ss} \cdot \mathcal{H}_2(\mathcal{S}_3). \tag{5}$$

To make the verification feasible, it must holds

$$\mathcal{S}_4 = r_{ss} \cdot \mathcal{H}_2(\mathcal{H}_1(SS_{tid})) + s_{ss} \cdot \mathcal{H}_2(r_{ss} \cdot G) \bmod k \tag{6}$$

and

$$SS^{'}_{tid} = SS_{tid}. \tag{7}$$

3.4 Adversary Model

Our adversary model is as follows: C and SS are controlled by an active adversary, and BS is controlled by a passive adversary in terms of different attacks such as man-in-the-middle attacks, replay attacks, server-side compromisation attacks, identity compromisation attacks, etc. The active adversary can behave arbitrarily in order to make the above discussed attacks feasible. Here, we consider the outside intruder as an active adversary. In [18], a passive adversary pursue a honest-but-curious activity, that is, it honestly execute the protocol according to the specification and does not modify any data in server's secure database. But this adversary listens the communication channels to derive the security credentials

between C and SS. Further, the passive adversary also knows all the shared secret keys and the internal state of the server. Intuitively, the implicit knowledge of each entity involve into our proposed security system is as follows:

(1) BS retains four dictionaries namely $D|C^{id}_{reg}|$, $D|S_{secret}|$, $D|r_{c,b}|$ and $D|K_{b,ss}|$ respectively. Assume, a passive adversary that controls BS does not have any access to these dictionaries.
(2) C has a unique reference number (i.e., $r_{c,b}$) and his identity (C_{ID}). An active adversary controlling C has the knowledge about C_{ID} but does not have any idea about $r_{c,b}$.
(3) SS maintains one dictionary (i.e., $D|C_{tid}|$), a shared secret key between SS and BS (i.e., $K_{b,ss}$) and $SS-BS$ shared secret identity (i.e., S_{secret}), and SS's identity (SS_{ID}) respectively. In this situation, an active adversary controlling SS having the knowledge only about the dictionary $D|C_{tid}|$ and SS_{ID}.

We evaluate an informal security analysis of our proposed protocol as follows:

4 SECURITY ANALYSIS

In this section, we present an informal security analysis of our proposed protocol based on the aforesaid adversary model and two security assumptions (see Section 2) in elliptic curve cryptography (ECC) [16] as follows:

Claim 1: *Resilient against man-in-the-middle attacks as an active adversary controlling no servers.*

Proof: Suppose an adversary $\mathcal{A}dv$ traps the message $\langle m_1 = \{C_1, C_2, SS_{ID}, \mathcal{S}_1, \mathcal{S}_2, CR_C\}\rangle$ sent by C to the SS. It is noted that

$$\mathcal{S}_1 = r_c \cdot G, r_c \in \mathbb{Z}^*_k \tag{8}$$

$$\mathcal{S}_2 = r_c \cdot \mathcal{H}_2(\mathcal{H}_1(C_{tid})) + s_c \cdot \mathcal{H}_2(\mathcal{S}_1) \bmod k. \tag{9}$$

It is computationally hard to compute r_c in equ. 8 (or s_c in equ. 9) from $\mathcal{S}_1$ (or Q_c) due to ECDLP. Further, $\mathcal{A}dv$ can not alter either $\mathcal{S}_1$ or $\mathcal{S}_2$ or both. Suppose, $\mathcal{A}dv$ alters $\mathcal{S}_2$ by $\mathcal{S}_2^{'}$. Then to satisfy the verification condition in equ. 9, $\mathcal{A}dv$ has to alter from $\mathcal{S}_1$ to $\mathcal{S}_1^{'}$. But, due to ECDLP it is hard to alter $\mathcal{S}_2$ by $\mathcal{S}_2^{'}$ and then satisfy the condition in equ. 9. Analogously, it is also hard to alter $\mathcal{S}_1$ by $\mathcal{S}_1^{'}$ and then find $\mathcal{S}_2^{'}$ so that $\mathcal{S}_2^{'} \cdot G = \mathcal{S}_1 \cdot \mathcal{H}_2(\mathcal{H}_1(C_{tid})) + Q_c \cdot \mathcal{H}_2(\mathcal{S}_1)$ holds. Similarly, it holds for message m_4 also. Further, suppose the adversary $\mathcal{A}dv$ wants to find the session key (i.e., SK) from either the value of $\mathcal{S}_1 = r_c \cdot G$ or $\mathcal{S}_3 = r_{ss} \cdot G$ or both by applying man-in-the-middle attack over the communication channel, and guessing the value of r_c and r_{ss}, then $\mathcal{A}dv$ would again end up with the computationally intractable or hard problem (i.e., ECCDHP assumption) to compute $SK = r_c \cdot \mathcal{S}_3 = r_{ss} \cdot \mathcal{S}_1 = r_c \cdot r_{ss} \cdot G$. We have ignored man-in-the-middle attacks feasibility for messages m_2 and m_3 as our intension is to establish a session key between C and SS. Hence, we can conclude that our scheme is resilient to man-in-the-middle attacks. □

Claim 2: *Resilient against replay attacks as an active adversary controlling no servers.*

Proof: In a particular session, C chooses a pseudo-random number r_c and it behaves like a nonce or a fresh value. We can easily infer from the four communication messages (i.e., m_1, m_2, m_3 and m_4 respectively) that the $\mathcal{S}_1$ value is exchanged among three parties and it is derived from r_c. As r_c is fresh for a single protocol run thus we can conclude that $\mathcal{S}_1$ is also fresh and vis-a-vis all the messages are also fresh. It ensures that our protocol is safe from replay attacks. □

Claim 3: *Resilient against identity compromisation attacks from an active adversary by controlling SS.*

Proof: In our scheme, SS is allowed to store only hashed and encrypted client identities (i.e., $h(C_1||C_2)$ where $C_1 = E(r_{c,b} : [C_{ID}])$ and $C_2 = C^A_{ID}$) instead of C's original identities. Therefore, if a malicious insider or a passive adversary ($\mathcal{A}dv$) controls SS, it gets the dictionary of client transformed identities (i.e., $D|C_{tid}|$) and analyses it offline, he can not trace out the original identity of a particular client. It may be noted that, in information theoretic sense, getting C_1 from the hash value of both C_1 and C_2 is computationally hard. Suppose, $\mathcal{A}dv$ gets both C_1 and C_2 from protocol transcript and try to guess the value of C_{ID} then, $\mathcal{A}dv$ needs to guess the value of $r_{c,b}$. But, it is also computationally hard to gain the knowledge of $r_{c,b}$ from a large key space. Therefore, we can say that our scheme is robust against SS-side identity compromisation attacks. This completes the proof. □

Claim 4: *Resilient against identity compromisation attacks from a passive adversary by controlling BS.*

Proof: According to our proposed adversary model (refer Section 3.4), we can observe that BS maintains four dictionaries namely $D|C^{id}_{reg}|$, $D|S_{secret}|$, $D|r_{c,b}|$ and $D|K_{b,ss}|$. Further, we assume that a passive adversary ($\mathcal{A}dv$) that controls BS does not have any access to those dictionaries. In this regard, $\mathcal{A}dv$ can only listens the communication channel between SS and BS. As a result, $\mathcal{A}dv$ will only get the messages m_2 and m_3. From these messages, $\mathcal{A}dv$ wants to gain the knowledge about C_{ID} then it needs to know $r_{c,b}$. As $\mathcal{A}dv$ does not have any access of BS's secret database, it is computationally hard to find $r_{c,b}$. Therefore, we can infer that our proposed scheme is resilient to BS-side client compromisation attacks. □

Claim 5: *Resilient against single point of vulnerability and single point of failure.*

Proof: In our proposed dual server model, BS is not directly reachable to the client/external adversary and it manages the crucial security credentials for both C and SS respectively. Although, SSs' are the front-end interface for malicious external adversaries, it is not possible to gain any knowledge about the registered secret identities (i.e., C^{id}_{reg} and S_{secret}) except the hashed identities' of both C and SS. Thus, it voids the single point of vulnerability issue. In addition to this, distribution of secret databases, periodical back-up and auditing of the security credentials from back-end is easy task rather than doing all these stuffs along with client and/or external intruder interfacing at front-end server (i.e., single server based approach). This way our solution may avoid the

single point of failure. □

Claim 6: *Resilient against disclosure of client's future session keys by server compromisation attacks on SS.*

Proof: Intuitively, to address the compromisation of client's future session keys considering server compromisation attacks on SS by an active adversary, we plan our scheme in such a way that in each session an entity would be able to substantiate the other entity along with the issuer of the security credentials by means of pseudo-random number generation and digital signature verification. Suppose, SS wants to check the legitimacy of C by verifying both the condition in equ. 2 and equ. 4 respectively. From equ. 2, we can easily infer that to satisfy the condition SS needs three parameters: (1) $K_{b,s}$: to decrypt $\mathcal{M}_1$ and find the C's transformed identity, (2) Q_c and (3) $\mathcal{S}_1$: to verify C's digital signature. Now consider a situation where SS is fully compromised by an active adversary $\mathcal{A}dv$ and he gets $D|C^{'}_{tid}|$, S_{secret}, $SS-BS$'s shared secret key ($K_{b,ss}$), etc. Getting all these knowledge is not enough for the $\mathcal{A}dv$ to compromise the C's future session key due to the pseudo-random number (r_c) generation at client-side and the computational hardness property of ECDLP assumption. This completes the proof. □

Claim 7: *Preserve C's privacy by hiding his original identity from external intruders utilizing "k-anonymity".*

Proof: In our proposed protocol, the client application A_i assigns a non unique random number between R_1 and R_2 to C as C^A_{ID}. In this regards, after the settlement of the session key between C and SS, C use to access his intended services from SS using C^A_{ID} (see the message exchanges (6) and (7) in Fig. 2). Suppose, there exists a set of clients (assume, k number of clients having similar identities) and they are concurrently asking services from SS. In this synergy, an adversary $\mathcal{A}dv$ can not distinguish the k^{th} client from at-least $(k-1)$ number of clients. Therefore, it creates more confusion to the external adversary to guess which identity is used for which client. Hence, we can conclude that our proposed approach preserves user privacy. □

5 FORMAL ANALYSIS WITH AVISPA

In order to do a formal analyses our proposed protocol, we utilize AVISPA (Automated Validation of Internet Security Protocols and Applications) tool [1]. This tool is providing a high-level protocol specification language (HLPSL), which is the *de facto* language recommended for AVISPA, to codify a formal security protocol. We code our authentication message exchanges in HLPSL. More precisely, we have specified three different roles: a client (c), a service server (ss) and a back-end authentication server (b). However, by taking an instance of these basic roles, we have initiated a composite role called "session" to configure a single protocol run. In order to specify the security objectives in HLPSL, AVISPA equips some standard commands, for example, "secrecy_of" (use to define the secrecy of keys), "authentication_on" (use to specify strong authentication on challenges), "weak_authentication_on" (use to represent weak authentication on timestamp) etc. We model the security goals for our proposed protocol as follows.

(1) *Origin authentication:* It is done by incorporating the "authentication_on" to every security goal in every data exchange.
(2) *Replay attack prevention:* It is achieved by establishing of strong origin authentication and assessing fresh message exchange between two parties using the nonce as $\mathcal{S}_1$.
(3) *Data confidentiality:* It is ensured by interpreting "secrecy_of" to the secret keys (i.e., $K_{b,ss}$ and $r_{c,b}$) used in the protocol.
(4) *Data integrity:* It is assured by applying "secrecy_of" to the secret one time distinct values (i.e., $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_4$) exchanged by the protocol.

In order to validate our proposed protocol, we have considered four different scenarios to find out the attack traces in AVISPA. All the scenarios and session configurations are presented in Tab. 2 and illustrated as follows.

Table 2: Different test cases execution in AVISPA.

Scenarios	Session configurations
Scenario 1	session(b,c,ss,r_cb,k_bs,text_set_L_1,hash,text_u,text_s,Q_c,Q_ss)
Scenario 2	session(b,i,ss,r_cb,k_bs,text_set_L_1,hash,text_u_i,text_s,Q_i,Q_ss)
Scenario 3	session(b,c,i,r_cb,k_bs,text_set_L_1,hash,text_u_i,text_s,Q_c,Q_i)
Scenario 4	session(i,c,ss,r_cb,k_bs,text_set_L_1,hash,text_u_i,text_s,Q_c,Q_ss)

- **Scenario 1:** It shows a single protocol session implementation with all the legitimate agents (i.e., b, c and ss) involved in the authentication.
- **Scenario 2, 3 and 4:** It considers a situation, where an adversary (i) is trying to impersonate the client (c), the service server (ss) and the back end authentication server (b).

After execution of all the above scenarios in AVISPA, it does not reveals any attack traces. Thus, we can conclude that our proposed protocol is safe.

6 PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our proposed protocol. We consider three major metrics for our evaluation as follows:

(1) **Computational time:** In information theoretic sense, exponentiations governs individual entity's computational overhead. In this synergy, we estimate the number of exponentiations as the evaluation of execution time and outline the performance results in Tab. 3. Consider C with A_i, for instance; it needs to compute a total of 6 exponentiations (see equ. 5 and clientSigGen[ ]()), that is, for the calculation of $\mathcal{S}_1$, $r_c\mathcal{H}_2(\mathcal{H}_1(C_{tid}))$, $Q_c \cdot \mathcal{H}_2(r_c \cdot G)$, $\mathcal{S}_4 \cdot G$, $Q_{ss} \cdot \mathcal{H}_2(r_{ss} \cdot G)$ and $\mathcal{S}_3 \cdot \mathcal{H}_2(\mathcal{H}_1(SS^{'}_{tid}))$ respectively (also refer [16] for point addition and doubling), and 3 exponentiations (i.e., calculation of $\mathcal{S}_1$, $r_c\mathcal{H}_2(\mathcal{H}_1(C_{tid}))$ and $Q_c \cdot \mathcal{H}_2(r_c \cdot G)$) out of them can be performed offline using [6] [9]. We represent the value as "x/y" notation. Similarly we can find the total number of exponentiations for SS and BS respectively. Here, for ease of calculations, we have ignored symmetric key encryptions/decryptions and hash operations for our analyses.

Table 3: Performance of our proposed protocol.

Participants	Yang et al. [18]	JWX Protocol [12]	Yi et al. [19]	DHKEP [5]	Our scheme
C	comm. $4L + 2l$ comp. 4/2 rounds 4	comm. $6L + 2l$ comp.6 rounds 3	comm. $3L + 4l$ comp. 4 rounds 3	comm. $1L$ comp. 2 rounds 2	comm. $\|\mathcal{J}\| + \|\mathcal{L}\|$ comp. 6/3 rounds 2
SS	comm. $8L + 3l$ comp. 4/1 rounds 8	comm. $11L + 3l$ comp. 8 rounds 6	comm. $6L + 3l$ comp. 5 rounds 4	comm. $1L$ comp. 2 rounds 2	comm. $3\|\mathcal{J}\| + \|\mathcal{L}\| + \|\mathcal{M}_2\|$ comp. 6/5 rounds 4
BS	comm. $4L + 1l$ comp. 3/1 rounds 4	comm. $5L + 1l$ comp. 4 rounds 3	comm. $6L + 3l$ comp. 5 rounds 4	comm. 0 comp. 0 rounds 0	comm. $\|\mathcal{J}\| + \|\mathcal{M}_1\| + \|\mathcal{M}_2\|$ comp. 0 rounds 2

(2) **Communication overhead:** Since, $|\mathcal{S}_1|$ is equal to $|\mathcal{S}_3|$ and $|\mathcal{S}_2|$ is equal to $|\mathcal{S}_4|$, so we can not distinguish among these four parameters. Thus, for ease of comparison, we fix it as $\mathcal{J} = |\mathcal{S}_1| = |\mathcal{S}_3|$ and $\mathcal{L} = |\mathcal{S}_2| = |\mathcal{S}_4|$. In addition to this, we have grossly ignored the bandwidth for both principal identities (i.e., C_1, C_2, SS_{ID}, BS_{ID}, etc.) and the public certificates (i.e., CR_C, CR_{SS}, etc.) in this aspect of evaluation.
(3) **Communication rounds:** A single round comprises a one-way transmission of messages.
Tab. 3 shows that our proposed protocol is quite efficient in terms of both communication and computation. Thus, we may integrate our protocol for wireless devices such as smart phone, personal digital assistant (PDA), etc. also.

7 CONCLUSION

A robust key exchange protocol for accessing remote services has been proposed in this work. Our protocol successfully alleviates the several issues in the existing protocols. Our security analysis reveals that our scheme is resilient to various security attacks. In addition to this, our formal verification substantiate that our proposed protocol is provably secure. Performance analysis also validates that our protocol is efficient in terms of both computation and communication overhead vis-a-vis ease to apply for wireless applications. More significantly, a large scale service servers' can be integrated with a single authorization center hosting our protocol.

REFERENCES

[1] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, et al. 2005. The AVISPA tool for the automated validation of internet security protocols and applications. In *International Conference on Computer Aided Verification*. Springer, 281–285.

[2] Victor Boyko, Philip MacKenzie, and Sarvar Patel. 2000. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology, Eurocrypt 2000*. Springer, 156–171.

[3] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. 2003. Security proofs for an efficient password-based key exchange. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*. ACM, 241–250.

[4] Liqun Chen, Zhaohui Cheng, and Nigel P Smart. 2007. Identity-based key agreement protocols from pairings. *International Journal of Information Security* 6, 4 (2007), 213–241.

[5] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *Transactions on Information Theory* 22, 6 (1976), 644–654.

[6] Youssou Faye, Herve Guyennet, Ibrahima Niang, and Yanbo Shou. 2013. Fast scalar multiplication on elliptic curve cryptography in selected intervals suitable for wireless sensor networks. In *Cyberspace Safety and Security*. Springer, 171–182.

[7] Dario Fiore and Rosario Gennaro. 2010. Identity-based key exchange protocols without pairings. In *Transactions on computational science X*. Springer, 42–77.

[8] Pierre-Alain Fouque, Antoine Joux, and Mehdi Tibouchi. 2013. Injective encodings to elliptic curves. In *Australasian Conference on Information Security and Privacy*. Springer, 203–218.

[9] Robert Gallant, Robert Lambert, and Scott Vanstone. 2001. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology, CRYPTO 2001*. Springer, 190–200.

[10] Hua Guo, Yi Mu, XY Zhang, and ZJ Li. 2010. Enhanced McCullagh-Barreto identity-based key exchange protocols with master key forward security. *International Journal of Security and Networks* 5, 2-3 (2010), 173–187.

[11] Hai Huang and Zhenfu Cao. 2009. An ID-based authenticated key exchange protocol based on bilinear Diffie-Hellman problem. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. ACM, 333–342.

[12] Haimin Jin, Duncan S Wong, and Yinlong Xu. 2007. An efficient password-only two-server authenticated key exchange system. In *Proceeding of Ninth International Conference of Information and Communication Security*. Springer, 44âĂŤ56.

[13] Hugo Krawczyk. 2003. SIGMA: The âĂŸSIGn-and-MAcâĂŹapproach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Annual International Cryptology Conference*. Springer, 400–425.

[14] F Pereniguez, R Marin-Lopez, Georgios Kambourakis, Stefanos Gritzalis, and AF Gomez. 2011. PrivaKERB: A user privacy framework for Kerberos. *Computers & Security* 30, 6 (2011), 446–463.

[15] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[16] NIST white paper. 1999. Recommended Elliptic Curves for Federal Government Use. http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf. (1999). [Online; Last accessed: 2nd April, 2017].

[17] Tsu-Yang Wu and Yuh-Min Tseng. 2009. An ID-based mutual authentication and key exchange protocol for low-power mobile devices. *Comput. J.* 53, 7 (2009), 1062–1070.

[18] Yanjiang Yang, Robert H Deng, and Feng Bao. 2006. A practical password-based two-server authentication and key exchange system. *Transactions on Dependable and Secure Computing* 3, 2 (2006), 105–114.

[19] Xun Yi, San Ling, and Huaxiong Wang. 2013. Efficient two-server password-only authenticated key exchange. *Transactions on Parallel and Distributed systems* 24, 9 (2013), 1773–1782.

[20] Xun Yi, Fang-Yu Rao, Zahir Tari, Feng Hao, Elisa Bertino, Ibrahim Khalil, and Albert Y Zomaya. 2016. ID2S password-authenticated key exchange protocols. *Transactions on Computers* 65, 12 (2016), 3687–3701.