Forward Anonymity-Preserving Secure Remote Authentication Scheme

Hanwook Lee[1], Junghyun Nam[2], Moonseong Kim[3] and Dongho Won[1]
[1] Department of Computer Engineering, Sungkyunkwan University, Korea
[e-mail: hwlee@security.re.kr, dhwon@security.re.kr]
[2] Department of Computer Engineering, Konkuk University, Korea
[e-mail: jhnam@kku.ac.kr]
[3] Information Management Division, Korean Intellectual Property Office, Korea
[e-mail: moonseong@kipo.go.kr]
*Corresponding author: Dongho Won

Received May 10, 2015; revised October 23, 2015; accepted January 21, 2016; published March 31, 2016

Abstract

Dynamic ID-based authentication solves the ID-theft problem by changing the ID in each session instead of using a fixed ID while performing authenticated key exchanges between communicating parties. User anonymity is expected to be maintained and the exchanged key kept secret even if one of the long-term keys is compromised in the future. However, in the conventional dynamic ID-based authentication scheme, if the server's long-term key is compromised, user anonymity can be broken or the identities of the users can be traced. In addition, these schemes are vulnerable to replay attacks, in which any adversary who captures the authentication message can retransmit it, and eventually cause the legitimate user to be denied service. This paper proposes a novel dynamic ID-based authentication scheme that preserves forward anonymity as well as forward secrecy and obviates replay attacks.

Keywords: Forward anonymity, dynamic ID-based authentication, smart card

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning (2014R1A1A2002775).

http://dx.doi.org/10.3837/tiis.2016.03.019 **ISSN : 1976-7277**

1. Introduction

Authenticated key exchange allows two or more parties to compute shared keys and also ensures their identities are authentic in insecure networks. Conventional authenticated key exchange protocols transfer identity information (ID) in plain text. Consequently, attackers can eavesdrop on communications between parties for information that enables them to forge the login request messages of legitimate users. Das *et al*. proposed a countermeasure in the form of a dynamic ID-based authentication scheme in which the user ID changes in every session based on the one-way hash function [1]. In the proposed scheme, the dynamic ID appears like a random number but in such a manner that legitimate parties can acquire the correct static ID without attackers being able to distinguish it. Although the scheme's ability to provide user anonymity, mutual authentication, and security was subsequently deemed inadequate [2-4], it significantly influenced later studies related to user anonymity protection.

The dynamic ID-based authentication scheme can be classified as a client encryption scheme or a server encryption scheme, depending on the party that creates the dynamic ID. In the client encryption scheme, the client encrypts the static ID using secret information induced either by the server's long-term or public key [1, 3-12]. Conversely, in the server encryption scheme, the server encrypts the static ID using its own long-term key and transfers this dynamic ID, for subsequent use in the authentication process, to the user [13-18]. In each of these schemes, the user requires a device such as a smart card that can store and manage secret information acquired from the server's long-term key, its public key, or a dynamic ID securely created by the server. Smart cards are convenient, portable, and inexpensive. Consequently, they are being widely adopted for two-factor authentication in remote host login, online banking, e-commerce, and e-health systems. Initially, dynamic ID-based authentication schemes adopted hash functions or symmetric key encryptions because of the poor performance of smart cards. However, more recent schemes have adopted complex operations such as public key encryption as a result of advancements in smart card technology. The most outstanding advantage of public key encryption over hash functions and symmetric key encryption is its ability to ensure forward secrecy.

Forward secrecy is a property that ensures that an established session key is not damaged, even if the long-term key used in key establishment is compromised. It is very important to ensure forward secrecy in order to prevent data leaks in the face of long-term key damage as a result of a system failure or leak, whether accidental or deliberate, because past data can be sensitive in the future as well. Various schemes that ensure forward secrecy have been proposed by researchers such as Sun *et al*. [14], Horng *et al*. [5], Wu *et al*. [8], Ma *et al*. [9], Wang *et al*. [10], and Jiang *et al*. [18].

However, because the schemes cited above used static long-term or public keys instead of the server's ephemeral public key to generate a dynamic ID, they had the following two vulnerabilities. First, forward anonymity is not assured. Forward anonymity—first proposed by Diffie *et al*. to explain their station-to-station protocol attributes [19]—is similar to forward secrecy but prevents attackers from exposing any information about the identities of participants. In server encryption schemes and client encryption schemes that use secret information induced from a server's long-term key, the static ID can be acquired from collected communication messages if a server's long-term key is compromised. In client encryption schemes that use a static rather than an ephemeral public key to transfer a dynamic ID, the static ID can be identified immediately or can be traced if the static public key is compromised. As a result, the revealed ID may be used to identify messages from specific

users, or as partial information to regenerate session keys.

Second, the schemes are vulnerable to replay attacks. Because the user ID is transferred using plain text in authentication schemes that do not ensure user anonymity, an eavesdropper can guess its password. This type of scheme defends against online dictionary attacks by limiting the number of attempts, but cannot limit attacks by attackers who try incorrect passwords repeatedly in order to induce denial of service. On the other hand, in schemes that ensure user anonymity, attackers cannot distinguish the static IDs of victims. Thus, replay attacks must be prevented if attackers do not know the static ID or dynamic ID generated by the server. However, attackers who eavesdrop on login request messages can carry out replay attacks by retransmitting the message to the server. This paper proposes a novel dynamic ID-based authentication scheme that solves the above problems by providing forward secrecy and forward anonymity. Further, we prove that the proposed scheme is secure under the computational Diffie-Hellman assumption and in the random oracle and ideal-cipher models.

The remainder of this paper is organized as follows: Section 2 defines the adversary model. Section 3 gives a brief review and cryptanalysis of Horng *et al*.'s, Wu *et al*.'s, and Wang *et al*.'s schemes. Section 4 outlines our proposed scheme. Section 5 and Section 6 conduct a security analysis and a performance analysis of our proposed scheme, respectively. Section 7 concludes this paper.

2. Adversary Model

Hao summarized robust security in the extreme-adversary principle as protection against an extremely powerful adversary who has all capabilities except that of trivially breaking a certain scheme [20]. On the basis of this ultimate definition of protocol security, Wang *et al*. [10] proposed the following six capabilities of the adversary for password-based dynamic ID authentication schemes that utilize smart card as follows: (1) full control of communication channel, (2) off-line enumeration of all possible ID-password pairs, (3) user ID identification, (4) password detection or sensitive information extraction from smart cards, (5) acquisition of the previous session key(s), and (6) acquisition of the server's long-term key(s). They proved the above capabilities both theoretically and practically through the following cases: ID and password can be exposed through malicious smart card readers; ID can be guessed easily when a user selects an easy-to-memorize ID; and as a result, ID has low entropy so attackers can attempt off-line guessing attacks to all possible ID-password pairs within polynomial time. Note that the capability to acquire the server's long-term key among the above six capabilities is limited to evaluation of the server's final failure, called weak forward secrecy [21]. A scheme that provides weak forward security is vulnerable to session interception and user impersonation attacks when a long-term key is compromised.

The scheme proposed by Wang *et al*. [10] is trivially broken when a long-term key and the user registration table in the server are simultaneously compromised. Attackers can acquire either the server's long-term key or user registration table without restrictions under the condition that two events do not occur simultaneously, according to the extreme-adversary principle. However, the scheme uses low precision time, resulting in it being susceptible to user impersonation attacks that can easily be carried out by any attacker who acquires a long-term key.

We propose the following adversary model, in which the adversary model proposed by Wang *et al*. is bolstered with strong forward secrecy characteristics:

1. Adversary $\mathcal{A}$ has full control of the communication channel between communicating parties.
2. Adversary $\mathcal{A}$ can enumerate all possible ID-password pairs within a reasonable amount of time off-line.
3. The (active) adversary $\mathcal{A}$ can determine the victim's identity.
4. Adversary $\mathcal{A}$ may either learn the victim's password or extract secret data from the lost smart card, but cannot achieve both.
5. Adversary $\mathcal{A}$ can learn the previous session key(s).
6. Attacker $\mathcal{A}$ may learn either the server's long-term key or the registration table managed by the server, but cannot achieve both.

Capabilities 1 to 5 are the same as those proposed by Wang *et al*. Capability 6 includes forward secrecy against the active adversary. If the adversary learns both the server's long-term key and the registration table, then the protocol can be trivially broken; therefore, it was excluded from the capabilities.

3. Review of Dynamic ID-based Authentication Schemes

In this section, we briefly review dynamic ID-based authentication schemes that ensure forward secrecy—specifically, those proposed by Horng *et al*. [5], Wu *et al*. [8], and Wang *et al*. [10]—and discuss their vulnerabilities. Each scheme is a client encryption scheme that performs Diffie-Hellman key exchange using the client's ephemeral public key and the server's public key, and then encrypts the static ID using the key exchanged. The procedure comprises several phases—specifically, registration, login, authentication, and password changing. The password changing phase is not dealt with in this paper. The notations used throughout this paper are defined in **Table 1**.

Table 1. Notations

Symbol	Description
U	User
S	Remote server
id	User's ID
pw	User's password
$SC[M]$	Smart card containing M
x	Private key of remote server
y	Public key of remote server
sk_{US}	Established session key
$\oplus$	Bitwise XOR operation
$\|\|$	Concatenation operation
$\mathcal{H}(\cdot)$	One-way hash function
$\mathcal{E}_k(\cdot)$	Symmetric encryption with key k
$\mathcal{D}_k(\cdot)$	Symmetric decryption with key k
$A \rightarrow B: M$	A sends M to B through a public channel
$A \Rightarrow B: M$	A sends M to B through a secure channel

3.1 Review of Horng *et al.*'s Scheme

Horng *et al*.'s scheme uses $\mathcal{H}(s)$ and $g^{\mathcal{H}(s)}$ as the server's private key and corresponding public key, respectively.

Registration phase:

1. User U chooses his/her identity id, password pw, and a random number b.
2. $U \Rightarrow S$: $\{id, \mathcal{H}(b||pw)\}$.
3. S computes $W = \mathcal{H}(id||s) \oplus \mathcal{H}(b||pw)$ and $w = g^{\{\mathcal{H}(id||s)\cdot\mathcal{H}(s)\}} \bmod p$, where s is the secret key of S.
4. $S \Rightarrow U$: $\{SC[W, w, \mathcal{H}(\cdot), p, g]\}$.
5. On receiving the smart card from S, U inputs b into his/her smart card.

Login phase:

1. U inserts his/her smart card into the card reader and inputs his/her id and pw.
2. The smart card chooses random numbers a and u, and computes $I = W \oplus \mathcal{H}(b||pw)$, $C = g^{aI} \bmod p$, $R = w^a \bmod p$, $N_u = g^u \bmod p$, and $M_1 = I \oplus N_u$.
3. $U \rightarrow S$: $\{C, \mathcal{E}_R(id, M_1)\}$.

Authentication phase:

1. S computes $R = C^{\mathcal{H}(s)} \bmod p$, decrypts $\mathcal{E}_R(id, M_1)$ to obtain id and M_1, and checks the validity of id. If it is invalid, the login request is rejected. Otherwise, S chooses a random number v, and computes $I = \mathcal{H}(id||s)$, $N_u = I \oplus M_1$, $N_v = g^v \bmod p$, $M_2 = I \oplus N_v$, $sk_{US} = N_u^v \bmod p$, and $M_3 = \mathcal{H}(id||sk_{US}||N_u)$.
2. $S \rightarrow U$: $\{\mathcal{E}_R(M_2, M_3)\}$.
3. On receiving reply message $\mathcal{E}_R(M_2, M_3)$ from S, the smart card decrypts $\mathcal{E}_R(M_2, M_3)$ to obtain M_2 and M_3, computes $N_v = I \oplus M_2$ and $sk_{US} = N_v^u \bmod p$, and checks if $\mathcal{H}(id||sk_{US}||N_u)$ is equal to M_3. If they are not equal, the session is terminated.
4. $U \rightarrow S$: $\{M_4\}$, where $M_4 = \mathcal{H}(N_v||sk_{US})$.
5. On receiving message M_4 from U, S checks if $\mathcal{H}(N_v||sk_{US})$ is equal to M_4. If they are not, the session is terminated.

3.1.1 Cryptanalysis of Horng *et al.*'s scheme

Off-line password guessing attack: An attacker $\mathcal{A}$ acquires $W_{\mathcal{A}}, w_{\mathcal{A}}, b_{\mathcal{A}}$ stored in his smart card that was issued legally for a legitimate user by executing side-channel attacks. Then, $\mathcal{A}$ can get $g^{\mathcal{H}(s)} \bmod p$ via the following calculation:

$$w_{\mathcal{A}}^{(W_{\mathcal{A}} \oplus \mathcal{H}(b_{\mathcal{A}}||pw_{\mathcal{A}}))^{-1}} = (g^{\mathcal{H}(id_{\mathcal{A}}||s)\cdot\mathcal{H}(s)})^{\mathcal{H}(id_{\mathcal{A}}||s)^{-1}} = g^{\mathcal{H}(s)} \pmod p.$$

Now let us assume that $\mathcal{A}$ steals the smart card of victim U and acquires W, w, b stored in U's smart card in the same way. $\mathcal{A}$ can guess a password, $pw,^*$ from the dictionary Password and check that the corresponding $(g^{\mathcal{H}(s)})^{W \oplus \mathcal{H}(b||pw^*)} \bmod p$ is equal to w. If they are, the correct password is pw^*. Otherwise, $\mathcal{A}$ selects another password and repeats this process until the correct password is found.

Online ID verification attack: Attacker $\mathcal{A}$ may need a process to know whether the guessed ID is a valid ID registered on the server prior to attacking user U. The vulnerability of the scheme is that the validity of the ID can be checked online. If an attacker already has $g^{\mathcal{H}(s)}$ as in the off-line password guessing attack, he/she may compute $C' = g^{a'} \bmod p$ and send C' to

the server without knowing I, because the server cannot distinguish between C' and $C = g^{aI} \bmod p$, where a and a' are random numbers. An attack can be carried out as follows:

1. $g^{\mathcal{H}(s)} \bmod p$ is calculated from a legitimate smart card, as in the off-line password guessing attack.
2. Random numbers a' and r are selected, and $C' = g^{a'} \bmod p$ and $R' = (g^{\mathcal{H}(s)})^{a'} \bmod p$ are calculated.
3. With respect to id^*, whose validity check is desired, login request message $\{C', \mathcal{E}_{R'}(id^*, r)\}$ is transferred to S.
4. Since $R = (C')^{\mathcal{H}(s)} \bmod p = g^{a' \cdot \mathcal{H}(s)} \bmod p = R'$, S can decrypt $\mathcal{E}_{R'}(id^*, r)$. Thus, if S transmits a normal response, id^* can be a valid identity.

No forward anonymity: If a long-term master key s or static private key $\mathcal{H}(s)$ of S is compromised, the attacker can calculate $R = C^{\mathcal{H}(s)} \bmod p$ from the login request message $\{C, \mathcal{E}_R(id, M_1)\}$ and decrypt $\mathcal{E}_R(id, M_1)$ to determine id of the user.

Replay attack: Eavesdropper $\mathcal{A}$, who captured the login request message sent by U to S, subsequently retransmits this message to S. S decrypts the message as usual in the authentication phase, and a response message is created in reply to $\mathcal{A}$. However, $\mathcal{A}$ cannot decrypt the response message, hence correct M_4 cannot be calculated, causing authentication failure. $\mathcal{A}$ repeats this process and eventually exceeds the number of online attempts allocated to legitimate user U.

3.2 Review of Wu *et al.*'s Scheme

Wu *et al.*'s scheme is based on an elliptic curve E defined over a prime finite field F_p. Let P be a base point with a large order q, s be a master key, and $(x, Q = x \cdot P)$ be the server's private key and corresponding public key, respectively.

Registration phase:

1. User U chooses his/her identity id.
2. $U \Rightarrow S$: $\{id, cid\}$, where cid is the identity of the smart card.
3. S computes $A = \mathcal{H}_0(x||id||cid) + \mathcal{H}_0(pw_0||id)$, where pw_0 is the initial password. Then, S inserts (id, cid) into the registration table.
4. $S \Rightarrow U$: $\{SC[A, Q, E_p, \mathcal{H}_{i\ (i=0,1,2)}(\cdot)], pw_0\}$.
5. On receiving the smart card from S, U must change the password immediately in the password changing phase.

Precomputation phase: The smart card chooses two random elements r_1, r_2 in $\mathbb{Z}_q^*$, computes $R_1 = r_1 \cdot P$, $R_2 = r_2 \cdot P$, $Z_2 = r_2 \cdot Q$, and $h = \mathcal{H}_1(R_2||Q||Z_2)$, and stores R_1, r_1, R_2, h into its memory for use in the authentication and key agreement phase.

Authentication and key agreement phase:

1. U inserts his/her smart card into the card reader and inputs his/her id and pw.
2. The smart card computes $R_1^* = R_1 + A - \mathcal{H}_0(pw||id)$ and $did = h \oplus (id||cid)$.
3. $U \rightarrow S$: $\{R_1^*, R_2\}$.
4. On receiving message $\{R_1^*, R_2\}$ from U, S extracts $(id||cid) = did \oplus \mathcal{H}_1(R_2||Q||x \cdot R_2)$, and verifies the user's identity using the registration table. Then, S chooses a random

element w in $\mathbb{Z}_q^*$, and computes $R_1 = R_1^* - \mathcal{H}_0(x||id||cid)$, $W = w \cdot P$, $Z_1 = w \cdot R_1$, and $V_1 = \mathcal{H}_2(S||did||W||R_1^*||R_2||Z_1)$.

5. $S \rightarrow U$: $\{W, V_1\}$.
6. On receiving reply message $\{W, V_1\}$ from S, the smart card computes $Z_1 = r_1 \cdot W$, and checks if $\mathcal{H}_2(S||did||W||R_1^*||R_2||Z_1)$ is equal to V_1. If they are, it computes $V_2 = \mathcal{H}_2(did||S||R_1^*||R_2||W||Z_1)$.
7. $U \rightarrow S$: $\{V_2\}$.
8. On receiving V_2 from U, S checks if $\mathcal{H}_2(did||S||R_1^*||R_2||W||Z_1)$ is equal to V_2.
9. Finally, the smart card and server compute the common session key $sk_{US} = \mathcal{H}_2(did||S||R_1^*||W||Q||Z_1)$.

3.2.1 Cryptanalysis of Wu *et al*.'s scheme

Smart card loss attack: Attacker $\mathcal{A}$ steals the smart card of victim U and determines the A and R_1, r_1, R_2, h stored in the registration and precomputation phases, respectively, via side-channel attacks. Then, $\mathcal{A}$ returns the smart card to its original location without the victim's knowledge. Consequently, when U attempts to login to S, $\mathcal{A}$ eavesdrops on the login request message to acquire $\{did, R_1^*, R_2\}$, and guesses U's password via the following process:

1. $\mathcal{A}$ extracts $(id||cid) = did \oplus h$, and computes $C = A - R_1^* + R_1$.
2. $\mathcal{A}$ selects pw^* from the dictionary Password and checks whether $\mathcal{H}_0(pw^*||id)$ is the same as C. If it is, the correct password is pw^*. Otherwise, another password is selected and this process is repeated until the correct password is found.

No forward anonymity: If S's private key x is compromised, the attacker calculates $(id||cid) = did \oplus \mathcal{H}_1(R_2||Q||x \cdot R_2)$ from the login request message $\{did, R_1^*, R_2\}$, thereby finding the id and cid of the user.

Replay attack: Eavesdropper $\mathcal{A}$, who captured login request message $\{did, R_1^*, R_2\}$, which was transferred by U to S, subsequently retransmits this messages to S. Authentication will fail because $\mathcal{A}$ does not know r_1, which will be used to calculate Z_1. However, because S does not store all past messages transmitted by U, it cannot distinguish whether this request is an attack via retransmission or simply an incorrect password input by U. Thus, $\mathcal{A}$ repeats this process and may eventually exceed the number of online attempts allocated to legitimate user U.

3.3 Review of Wang *et al*.'s Scheme

Registration phase:

1. User U chooses his/her id, pw, and a random number b.
2. $U \Rightarrow S$: $\{id, \mathcal{H}_0(b||pw)\}$.
3. On receiving the registration message from U at time T, S computes $N = \mathcal{H}_0(b||pw) \oplus \mathcal{H}_0(x||id||T)$ and $A = \mathcal{H}_0\big((\mathcal{H}_0(id) \oplus \mathcal{H}_0(b||pw)) \bmod n\big)$. Then, S stores (id, T) in the registration table.
4. $S \Rightarrow U$: $\{SC[N, A]\}$.
5. On receiving the smart card from S, U inputs b into his/her smart card.

Login phase:

1. U inserts his/her smart card into the card reader and inputs his/her id and pw.
2. The smart card checks if $\mathcal{H}_0\big((\mathcal{H}_0(id) \oplus \mathcal{H}_0(b||pw))\big) \bmod n\big)$ is equal to A. If they are not, the session is terminated. The smart card chooses a random number u and computes $C_1 = g^u \bmod p$, $Y_1 = y^u \bmod p$, $k = N \oplus \mathcal{H}_0(b||pw) = \mathcal{H}_0(x||id||T)$, $cid = id \oplus \mathcal{H}_0(C_1||Y_1)$, and $M = \mathcal{H}_0(Y_1||k||cid)$.
3. $U \rightarrow S$: $\{C_1, cid, M\}$.

Verification phase:

1. On receiving the login request message $\{C_1, cid, M\}$ from U, S computes $Y_1 = C_1^x \bmod p$ and $id = cid \oplus \mathcal{H}_0(C_1||Y_1)$. If id is invalid, the session is terminated. S computes $k = \mathcal{H}_0(x||id||T)$, and checks if $\mathcal{H}_0(Y_1||k||cid)$ is equal to M. If they are not, the session is terminated. Then, S chooses a random element v and computes $K_S = C_1^v \bmod p$, $C_2 = g^v \bmod p$, and $C_3 = \mathcal{H}_1(id||S||Y_1||C_2||k||K_S)$.
2. $S \rightarrow U$: $\{C_2, C_3\}$.
3. On receiving $\{C_2, C_3\}$ from S, the smart card computes $K_U = C_2^u \bmod p$ and checks that $\mathcal{H}_1(id||S||Y_1||C_2||k||K_S)$ is equal to C_3. If they are not, the session is terminated. Then the smart card computes $C_4 = \mathcal{H}_2(id||S||Y_1||C_2||k||K_U)$.
4. $U \rightarrow S$: $\{C_4\}$.
5. On receiving C_4 from U, S checks that $\mathcal{H}_2(id||S||Y_1||C_2||k||K_S)$ is equal to C_4. If they are not, the session is terminated.
6. Finally, the smart card and server compute the common session key $sk_{US} = \mathcal{H}_3(id||S||Y_1||C_2||k||K_u) = \mathcal{H}_3(id||S||Y_1||C_2||k||K_S)$.

3.3.1 Cryptanalysis of Wang *et al.*'s scheme

No forward anonymity: When S's private key x is compromised, eavesdropper $\mathcal{A}$ can identify $id = cid \oplus \mathcal{H}_0(C_1||\ C_1^x \bmod p)$ using the information collected from the login request message $\{C_1, cid, M\}$.

Replay attack: Eavesdropper $\mathcal{A}$, who captured login request message $\{C_1, cid, M\}$, which was transferred by U to S, subsequently retransmits this messages to S. Authentication will fail because $\mathcal{A}$ does not know random number u, which will be used to calculate C_4 by $\mathcal{A}$. However, S does not store all past messages transmitted by U, so it cannot distinguish whether this request is an attack via retransmission or simply an incorrect password input by U. Thus, $\mathcal{A}$ repeats this process and may eventually exceed the number of online attempts allocated to legitimate user U.

User impersonation attack: Assuming that the precision of time T, generated in the registration phase, is less than or equal to a second, as shown in the Unix time function, its increase is less than 3.2×10^7 per year. As such, if low precision time is used and the server's private key x is compromise, eavesdropper $\mathcal{A}$ can calculate a shared secret k between user and server via the following process using information collected from login request message $\{C_1, cid, M\}$.

$\mathcal{A}$ calculates $Y_1 = (C_1)^x \bmod p$ and $id = cid \oplus\ \mathcal{H}_0(C_1||\ Y_1)$ from the message. Then, the following computation is performed with respect to all time T^*, from service initiation of the server to current time.

1. Compute $k^* = \mathcal{H}_0(x||id||T)$ and $M^* = \mathcal{H}_0(Y_1||k||cid)$.

2. Iterate the calculation until $M^* = M$.

The actual value shared by the server and user is k, so if the value of k is known, the user impersonation attack can be easily accomplished. To interrupt this attack, the time T used in the registration phase should contain a sufficiently long random number along with time information.

4. Our Proposed Scheme

In this section, we outline our proposed dynamic ID-based authentication scheme, in which forward secrecy and forward anonymity are preserved and replay attacks are obviated using a smart card. The scheme comprises three phases: registration phase, authentication phase, and password changing phase.

Registration phase: Our scheme is defined over a finite cyclic group $\mathbb{G}$ of prime order q with g as a generator. Hash functions $\{0,1\}^* \rightarrow \{0,1\}^{\ell_i}$ are denoted by $\mathcal{H}_i$, where $i \in \{0,1,2,3,4,5\}$ and ℓ_i is the output bit length of $\mathcal{H}_i$. Let $(x, y = g^x \bmod p)$ denote server S's private key and its corresponding public key. When user U wishes to register his/her ID to sever S, the following operations are performed:

1. U chooses his/her id.
2. $U \Rightarrow S$: $\{id\}$.
3. On receiving the registration request message from U, S chooses two random numbers r and R, and computes $ID = \mathcal{H}_0(id)$, $V_U = \mathcal{H}_5(x||ID||R)$ and $V_U^\star = \mathcal{E}_{r||pw_0}(V_U)$, where pw_0 is the initial password. Then, S stores (ID, R) in the registration table.
4. $S \Rightarrow U$: $\{SC[r, V_U^\star, y, g, p, q, \mathcal{H}_{i\ (i=0,1,2,3,4)}(\cdot)], pw_0\}$.
5. On receiving the smart card from S, U must change the password immediately in the password changing phase.

Authentication phase:

1. U inserts his/her smart card into the card reader, inputs his/her id and pw, and opens a session with S.
2. S chooses a random element b in $\mathbb{Z}_q^*$, and computes $B = g^b \bmod p$.
3. $S \rightarrow U$: $\{S, B\}$.
4. On receiving message $\{S, B\}$ from S, the smart card chooses a random element a in $\mathbb{Z}_q^*$, computes $A = g^a \bmod p$, $V_U = \mathcal{D}_{r||pw}(V_U^\star)$, $V_S = y^a \bmod p$, $K_U = B^a \bmod p$, $ID = \mathcal{H}_0(id)$, $ID^\star = ID \oplus \mathcal{H}_3(A||B||V_S||K_U)$, and $M_U = \mathcal{H}_1(ID||S||A||B||V_U||V_S||K_U)$.
5. $U \rightarrow S$: $\{ID^\star, A, M_U\}$.
6. On receiving message $\{ID^\star, A, M_U\}$ from U, S computes $V_S = A^x \bmod p$, $K_S = A^b \bmod p$, $ID = ID^\star \oplus \mathcal{H}_3(A||B||V_S||K_S)$, checks the validity of id using the registration table. If it is invalid, the session is terminated. Otherwise, S computes $V_U = \mathcal{H}_5(x||ID||R)$, where R is extracted from the entry corresponding to ID. Then, S checks if $\mathcal{H}_1(ID||S||A||B||V_U||V_S||K_S)$ is equal to M_U. If they are not, the session is terminated. Otherwise, S computes $M_S = \mathcal{H}_2(ID||S||A||B||V_U||V_S||K_S)$.
7. $S \rightarrow U$: $\{M_S\}$.
8. On receiving message M_S from S, U checks if $\mathcal{H}_2(ID||S||A||B||V_U||V_S||K_U)$ is equal to M_S. If they are not, the session is terminated.

9. Finally, the smart card and the server compute the common session key $sk_{US} = \mathcal{H}_3(ID||S||A||B||V_U||V_S||K_U) = \mathcal{H}_3(ID||S||A||B||V_U||V_S||K_S)$

Password changing phase: In our scheme, the user can change his/her password either interactively or non-interactively. Although the non-interactive procedure is simple, it may render the smart card unusable if the user enters a wrong password by mistake or an adversary intentionally inputs an arbitrary password after gaining temporary access to the smart card. Thus, we suggest the following interactive password change procedure:

1. U inserts his/her smart card into the card reader and inputs his/her id, old password pw, and new password pw'.
2. The smart card and the server perform the authentication phase with id and pw. If the smart card shares the session key successfully, it chooses a random number r', computes $V_U' = \mathcal{E}_{r'||pw'}(\mathcal{D}_{r||pw}(V_U^\star))$, and updates the values of $V_U^\star$ and r stored in its memory to V_U' and r', respectively.

Client U		**Server** S
Registration Phase:		x: S's private key
Select id		$y(=g^x)$: S's public key
	$\{id\}$ ⟹	
		$ID = \mathcal{H}_0(id)$
		Select two random elements r and R
		$V_U = \mathcal{H}_5(x\|\|ID\|\|R)$
		$V_U^\star = \mathcal{E}_{r\|\|pw_0}(V_U)$
		Store (ID, R) in registration table
		Store $(r, V_U^\star, y, g, p, q, \mathcal{H}_{i\ (i=0,1,2,3,4)}(\cdot))$ in smart card
	⟸ {smart card,pw_0}	
Authentication Phase:		
Input id, pw		
$a \in_R \mathbb{Z}_q^*$		$b \in_R \mathbb{Z}_q^*$
$A = g^a, V_S = y^a, K_U = B^a, ID = \mathcal{H}_0(id)$		$B = g^b$
	⟵ $\{S,B\}$	
$ID^\star = ID \oplus \mathcal{H}_3(A\|\|B\|\|V_S\|\|K_U)$		
$V_U = \mathcal{D}_{r\|\|pw}(V_U^\star)$		$V_S = A^x, K_S = A^b$
$M_U = \mathcal{H}_1(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_U)$		
	⟶ $\{ID^\star,A,M_U\}$	
		$ID = ID^\star \oplus \mathcal{H}_3(A\|\|B\|\|V_S\|\|K_S)$
		Abort if ID is invalid
		Retrieve R from registration table
		$V_U = \mathcal{H}_5(x\|\|ID\|\|R)$
		Abort if $M_U \neq \mathcal{H}_1(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_S)$
		$M_S = \mathcal{H}_2(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_S)$
	⟵ $\{M_S\}$	
Abort if $M_S \neq \mathcal{H}_2(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_U)$		
$sk_U = \mathcal{H}_0(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_U)$		$sk_S = \mathcal{H}_0(ID\|\|S\|\|A\|\|B\|\|V_U\|\|V_S\|\|K_S)$

Fig. 1. The proposed scheme

5. Security Analysis

In this section, we describe a formal security model for dynamic ID-based authentication schemes using smart cards, and show that our scheme is secure under the computational Diffie-Hellman assumption and in the random oracle and ideal-cipher models.

5.1 Formal Security Model

We adopt the security model defined by Wang *et al*. [10], which is based on previous work by Bellare *et al*. [22] and Bresson *et al*. [23], to define the special security requirements for password authentication schemes using smart cards.

Players. We denote a server S and a client U that can participate in the authenticated key exchange protocol P. Each of them may have several instances, called oracles, involved in distinct, possibly concurrent, executions of P. We denote client instances (resp. server instances) by U^i (resp. S^j), or by I when we consider any kind of instance.

Queries. The adversary, $\mathcal{A}$, interacts with the participants and tries to break the privacy of the key or the authentication of the players. To this aim, several queries are available to $\mathcal{A}$.

- $\mathsf{Execute}(U^i, S^j)$: This query models passive attacks, where the adversary eavesdrops on honest executions of P between U^i and S^j.
- $\mathsf{Reveal}(I)$: This query models the misuse of the session key by instance I. The query is only available to $\mathcal{A}$ if the attacked instance actually holds a session key and it releases sk to $\mathcal{A}$.
- $\mathsf{Send}(I, m)$: This query models $\mathcal{A}$ sending a message to instance I. $\mathcal{A}$ receives the response I generates in processing the message m according to the protocol P in return. In our scheme, a query $\mathsf{Send}(S^j, \mathsf{Start})$ initializes the key exchange algorithm, and thus the adversary receives the flow the server should send to the client.
- $\mathsf{Corrupt}(I, a)$: This query models the corruption capability of the adversary. $\mathcal{A}$ can indeed steal/break either one of the two authentication factors of clients, but not both. A similar process goes on in the private key or the registration table of the server.
  - If $a = 1$, it outputs $U's$ password, pw.
  - If $a = 2$, it outputs the parameter $V_U^\star$ stored in the smart card.
  - If $a = 3$, it outputs all pairs of (ID, R) in the registration table.
  - If $a = 4$, it outputs the private key x of S.

Semantic Security. The privacy of the session key is modeled by the game $\mathrm{Game}^{\mathrm{ake}}(\mathcal{A}, P)$, in which one more query is available to the adversary: $\mathsf{Test}(I)$. The $\mathsf{Test}(I)$-query can be asked at most once by adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the session key is not obviously known to $\mathcal{A}$. This query is answered as follows: one flips a coin b and forward sk (the value $\mathsf{Reveal}(I)$ outputs) if $b = 1$, or a random value if $b = 0$. In playing this game, the goal of the adversary is to guess the bit b involved in the $\mathsf{Test}(I)$-query, by outputting this guess b'. We denote AKE Advantage as the probability that $\mathcal{A}$ correctly guesses the value b. More precisely, we define it by $\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$. Protocol P is said to be AKE-secure if such a probability is negligible in the security parameter.

Authentication. Another goal of the adversary is to impersonate the client or the server. We denote by $\mathsf{Succ}_P^{\mathrm{auth}}(\mathcal{A})$ the probability that $\mathcal{A}$ successfully impersonates an instance of U in

the execution of P. Protocol P is said to be Auth-secure if such a probability is negligible in the security parameter.

Computational Diffie-Hellman Assumption. A (t, ε)-$\mathsf{CDH}_{g,\mathbb{G}}$ attacker, in a finite cyclic group $\mathbb{G}$ of prime order q with g as a generator, is a probabilistic machine Δ running in time t such that its success probability, $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(\Delta)$, given random elements g^x and g^y to output g^{xy}, is greater than ε. As usual, we denote by $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t)$ the maximal success probability over every adversaries running within time t. The CDH-Assumption states that $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t) \leq \varepsilon$ for any t/ε is not overly large.

5.2 Security Proof

Semantic Security. The following theorem shows that the proposed scheme securely distributes session keys under the reasonable and well-defined intractability assumptions.

Theorem 1. *Let P be the above protocol and* Password *be a finite dictionary of size N equipped with a uniform distribution. Let $\mathcal{A}$ be an adversary against the AKE security of P within a time bound t, with less than q_s interactions with the parties and q_p passive eavesdroppings, and asking q_h hash-queries and q_e encryption/decryption-queries. Then, we have*

$$\mathrm{Adv}^{\mathrm{ake}}_{\mathcal{P}}(\mathcal{A}) \leq \frac{q_s}{N} + 4q_h\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G}}(t') + \frac{(q_s+q_p)^2}{q-1} + \frac{(q_h+4q_s+q_p+q_e)^2}{2^{\ell}},$$

where $t' \leq t + (q_s + q_p + 1)\cdot\tau_{\mathbb{G}}$ with ℓ denoting $min\{\ell_1, \ell_2, \ell_3, \ell_4\}$ and $\tau_{\mathbb{G}}$ denoting the computational time for an exponentiation in $\mathbb{G}$.

Proof. We prove this theorem through a sequence of games beginning with the real protocol and ending up with a game where adversary $\mathcal{A}$'s advantage is zero. The details of the proof can be found in Appendix A.1. □

Authentication. The following theorem shows that the proposed scheme further ensures mutual authentication, in the sense that neither a server instance nor a client instance will accept an authenticator that has not been actually sent by the related instance with probability significantly greater than q_s/N.

Theorem 2. *Let P be the above protocol and* Password *be a finite dictionary of size N equipped with a uniform distribution. Let $\mathcal{A}$ be an adversary against the mutual authentication within a time bound t, with less than q_s interactions with the parties and q_p passive eavesdropping, and asking q_h hash-queries and q_e encryption/decryption-queries. Then, we have*

$$\mathrm{Adv}^{\mathrm{auth}}_{\mathcal{P}}(\mathcal{A}) \leq \frac{q_s}{2N} + q_h\mathsf{Succ}^{\mathsf{cdh}}_{\mathbb{G}}(t') + \frac{(q_s+q_p)^2}{2(q-1)} + \frac{(q_h+4q_s+q_p+q_e)^2}{2^{\ell+1}},$$

where $t' \leq t + (q_s + q_p + 1)\cdot\tau_{\mathbb{G}}$ with ℓ denoting $min\{\ell_1, \ell_2, \ell_3, \ell_4\}$ and $\tau_{\mathbb{G}}$ denoting the computational time for an exponentiation in $\mathbb{G}$.

Proof. The proof of this theorem is similar to the previous one. The details of the proof can be found in Appendix A.2. □

5.3 Resistance to Possible Attacks

User anonymity and forward anonymity: Even if attacker $\mathcal{A}$ eavesdrops on a message $\{B, ID^*, A, M_U, M_S\}$ between user and server during the authentication phase, and moreover, even if $\mathcal{A}$ knows the server's private key x, $\mathcal{A}$ has to know random number a, chosen by the user, or random number b, chosen by the server in order to acquire ID from ID^*. Otherwise, $\mathcal{A}$ has to solve a discrete logarithm problem to acquire ID from ID^* under the condition that the attacker does not know a or b. Thus, the proposed scheme provides user anonymity, user untraceability and forward anonymity.

Forward secrecy: Even if an adversary $\mathcal{A}$ eavesdrops on a message $\{B, ID^*, A, M_U, M_S\}$ between user and server during the authentication phase, and even if $\mathcal{A}$ knows the server's private key x and the confidential information V_U, $\mathcal{A}$ has to know random number a, chosen by the user, or random number b, chosen by the server, in order to acquire sk. Otherwise, $\mathcal{A}$ has to solve a discrete logarithm problem to acquire sk. Thus, the proposed scheme provides forward secrecy.

Smart card loss attack (user impersonation attack): M_U is protected by the secured one-way hash function so that server S can detect this easily if the attacker cannot calculate a valid M_U, even when the attacker has managed to tamper with a legitimate user's authentication message. Moreover, even if confidential information V_U^*, stored in the smart card is exposed, the correct V_U cannot be acquired if the attacker does not know the password. Consequently, a valid M_U cannot be calculated. Thus, the proposed scheme is secure against smart card loss and user impersonation attacks.

Replay attack: Unlike other schemes in which service denial attacks can succeed by exceeding the allocated number of online attempts using retransmission of captured login request messages, the proposed scheme uses the server's ephemeral public key, which is changed in every session, for ID^* conversion so that retransmission of captured messages cannot increase the number of attempts because it cannot identify the user of the particular session. Thus, replay attacks will fail.

Key compromise impersonation attacks: Let us assume that attacker $\mathcal{A}$ has stolen the server's private key x. However, $\mathcal{A}$ does not know random number R for user U, which is stored in the registration table, so $\mathcal{A}$ cannot calculate $\mathcal{H}_5(x||ID||R)$ correctly and U cannot be impersonated.

6. Performance Analysis

In **Table 2**, we provide a comparative summary of relevant dynamic ID-based forward security preserving authentication schemes, including our proposed scheme. The schemes are compared in terms of computation, communication, and security.

We assume that the identities are 32-bit long random numbers and that the outputs of the one-way hash function are 128-bit long. Further, we assume that public key parameters p, y, and g are 1024-bit, q is 128-bit long, and the elliptic curve point is 160-bit long. Let T_E, T_S, T_H, and T_M denote the time complexity for exponential operation, symmetric encryption/decryption, hash function evaluation, and scalar-point multiplication, respectively.

During the authentication phase in our scheme, the total computation cost of the user and

server is $6T_E + 1T_S + 10T_H$ and the communication cost is $2464 (= 32 + 3 \times 128 + 2 \times 1024)$ bits. Our scheme is more efficient than Horng *et al.*'s and is no less efficient than Wang *et al.*'s. Furthermore, it provides forward anonymity and resists both smart card loss attacks and replay attacks, whereas Horng *et al.*'s, Wu *et al.*'s, and Wang *et al.*'s schemes are susceptible to such attacks.

Table 2. Comparative summary of relevant forward security preserving authentication schemes

Scheme	Computation cost	Comm. cost	Security					
			SKS	FS	UA	FA	RSCL	RR
Our scheme	$6T_E + 1T_S + 10T_H$	2464 bits	Yes	Yes	Yes	Yes	Yes	Yes
Horng *et al.* [5]	$7T_E + 4T_S + 8T_H$	2432 bits	Yes	Yes	Yes	No	No	No
Wu *et al.* [8]	$4T_M + 8T_H$	864 bits	Yes	Yes	Yes	No	No	No
Wang *et al.* [10]	$6T_E + 14T_H$	2560 bits	Yes	Yes	Yes	No	Yes	No

SKS: session key security; FS: forward secrecy; UA: user anonymity; FA: forward anonymity; RSCL: Resistance to smart card loss attacks; RR: Resistance to replay attacks

7. Conclusion

In this paper, we demonstrated that three dynamic ID-based authentication schemes that preserve forward secrecy cannot guarantee forward anonymity and can be vulnerable to replay attacks because they use static long-term keys instead of the server's ephemeral keys to generate dynamic IDs. We then presented an adversary model that incorporates forward secrecy against active attackers according to the extreme-adversary principle and proposed a dynamic ID-based authentication scheme in which forward secrecy and forward anonymity are preserved and replay attacks can be obviated through retransmission of the authentication messages. We subsequently proved that our scheme is provably secure under the computational Diffie-Hellman assumption and in the random oracle and ideal-cipher models.

References

[1] M. L. Das, A. Saxena and V. P. Gulati, "A dynamic ID-based remote user authentication scheme," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 629-631, 2004. Article (CrossRef Link).

[2] A. K. Awasthi and S. Lal, "Security analysis of a dynamic ID-based remote user authentication scheme," *IACR Cryptology ePrint Archive*, 2004. Article (CrossRef Link).

[3] I. Liao, C. Lee and M. Hwang, "Security enhancement for a dynamic ID-based remote user authentication scheme," in *Proc. of Int. Conf. on Next Generation Web Services Practices*, 2005. Article (CrossRef Link).

[4] H. Chien and C. Chen, "A remote authentication scheme preserving user anonymity," in *Proc. of 19th Int. Conf. on Advanced Information Networking and Applications*, pp.245-248, 2005. Article (CrossRef Link).

[5] W. Horng, C. Lee and J. Peng, "A secure remote authentication scheme preserving user anonymity with non-tamper resistant smart cards," *WSEAS Transactions on Information Science and Applications*, vol. 7, no. 5, pp. 619-628, 2010. Article (CrossRef Link).

[6] J. Tsai, T. Wu and K. Tsai, "New dynamic ID authentication scheme using smart cards," *International Journal of Communication Systems*, vol. 23, no. 12 pp. 1449-1462, 2010. Article (CrossRef Link).

[7] M. K. Khan, S. Kim and K. Alghathbar, "Cryptanalysis and security enhancement of a `more efficient & secure dynamic ID-based remote user authentication scheme'," *Computer Communications*, vol. 34, no. 3. pp. 305-309, 2011. Article (CrossRef Link).
[8] S. Wu, Y. Zhu and Q. Pu, "Robust smart-cards-based user authentication scheme with user anonymity," *Security and Communication Networks*, vol. 5, no. 2, pp. 236-248, 2012. Article (CrossRef Link).
[9] C. Ma, D. Wang and Q. Zhang, "Cryptanalysis and improvement of Sood et al.'s dynamic ID-based authentication scheme," *Distributed Computing and Internet Technology*, pp. 141-152, 2012. Article (CrossRef Link).
[10] D. Wang, C. Ma, P. Wang and Z. Chen, "Robust smart card based password authentication scheme against smart card security breach," *IACR Cryptology ePrint Archive*, 2012. Article (CrossRef Link).
[11] C. Liu and C. Ma, "An efficient and provable secure pake scheme with robust anonymity," *Information Computing and Applications*, 2012. Article (CrossRef Link).
[12] T. Cao and J. Zhai, "Improved dynamic ID-based authentication scheme for telecare medical information systems," *Journal of medical systems*, vol. 37, no. 2, pp. 1-7, 2013. Article (CrossRef Link).
[13] W. Juang, S. Chen and H. Liaw, "Robust and efficient password-authenticated key agreement using smart cards," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 6, pp. 2551-2556, 2008. Article (CrossRef Link).
[14] D. Sun, J. Huai, J. Sun, J. Li, J. Zhang and Z. Feng, "Improvements of Juang's password-authenticated key agreement scheme using smart cards," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 2284-2291, 2009. Article (CrossRef Link).
[15] X. Li, W. Qiu, D. Zheng, K. Chen and J. Li, "Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 2, pp. 793-800, 2010. Article (CrossRef Link).
[16] C. Chang, H. Le and C. Chang, "Novel untraceable authenticated key agreement protocol suitable for mobile communication," *Wireless personal communications*, vol. 71, no. 1, pp. 425-437, 2013. Article (CrossRef Link).
[17] Q. Jiang, J. Ma, Z. Ma and G. Li, "A privacy enhanced authentication scheme for telecare medical information systems," *Journal of medical systems*, vol. 37, no. 1, pp. 1-8, 2013. Article (CrossRef Link).
[18] Q. Jiang, J. Ma, G. Li and L. Yang, "Robust two-factor authentication and key agreement preserving user privacy," *IJ Network Security* vol. 16, no. 3, pp. 229-240, 2014. Article (CrossRef Link).
[19] W. Diffie, P. C. Van Oorschot and M.J. Wiener, "Authentication and authenticated key exchanges," *Designs, codes and cryptography*, vol. 2, no. 2, pp. 107-125, 1992. Article (CrossRef Link).
[20] F. Hao, "On robust key agreement based on public key authentication," *Financial Cryptography and Data Security*, pp. 383-390, 2010. Article (CrossRef Link).
[21] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," *Advances in Cryptology-CRYPTO 2005*, pp. 546-566, 2005. Article (CrossRef Link).
[22] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," *Advances in Cryptology-Eurocrypt 2000*, pp. 139-155, 2000. Article (CrossRef Link).
[23] E. Bresson, O. Chevassut and D. Pointcheval, "Security proofs for an efficient password-based key exchange," in *Proc. of 10th ACM conf. on Computer and communications security*, pp. 241-250, 2003. Article (CrossRef Link).

A. Proof of Theorems

A.1 Proof of Theorem 1

Proof. In this proof, for simplicity, we do not consider forward secrecy. We incrementally define a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_8$. For each $\mathbf{G}_n$ $(n = 0, 1, \ldots, 8)$, we define the following events:

- S_n occurs if $\mathcal{A}$ correctly guesses the bit b involved in the Test-query.
- $\mathsf{Encrypt}_n$ occurs if $\mathcal{A}$ submits data it has encrypted by itself using the password.
- Auth_n occurs if $\mathcal{A}$ submits an authenticator $Auth$ that is accepted by the server and that has been built by the adversary itself.

Game $\mathbf{G}_0$: This is the real protocol in the random oracle and ideal-cipher models. Several oracles are thus available to the adversary: six hash oracles ($\mathcal{H}_i$ (with $i \in \{0, 1, 2, 3, 4, 5\}$) and all the instances U^i and S^j (in order to cover concurrent executions). By definition,

$$\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(\mathcal{A}) = 2\Pr[\mathsf{S}_0] - 1. \tag{1}$$

In the games below, we further assume that when the game aborts or stops with no answer b' outputted by adversary $\mathcal{A}$, we choose this bit b' at random, which in turn defines the actual value of the event, S_k. Moreover, if the adversary has not finished playing the game after q_s Send-queries or lasts for more than time t, we stop the game (and choose a random bit b'), where q_s and t are predetermined upper bounds.

Game $\mathbf{G}_1$: In this game, we simulate the hash oracles, $\mathcal{H}_i$, and five additional hash functions $\mathcal{H}_i'$(with $i \in \{0, 1, 2, 3, 4\}$ that will appear in Game $\mathbf{G}_6$) and the encryption/decryption oracles, as usual by maintaining a hash list $\Lambda_{\mathcal{H}}$ (and another list $\Lambda_{\mathcal{A}}$ containing the hash-queries asked by the adversary itself) (see **Fig. 2**) and an encryption list $\Lambda_{\mathcal{E}}$ (and another list $\Lambda_{\mathcal{D}}$ containing the encryption/decryption-queries asked by the adversary itself). We also simulate all the instances, as the real players would, for the Send-queries and for the Execute, Reveal and Test-queries (see **Fig. 3**). From this simulation, it is clear that the game is perfectly indistinguishable from the real attack.

$$\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0]. \tag{2}$$

Game $\mathbf{G}_2$: In this game, we avoid collisions among the hash-queries asked by the adversary to $\mathcal{H}_i$ (with $i \in \{1, 2, 3, 4\}$) , among the password and the ciphertexts, and among the Send-queries' output. We play the game in such a manner that no collision is found by the adversary for $\mathcal{H}_i$, at most one password corresponds to each plaintext-ciphertext pair, and abort if two instances of the server or client have used the same random values. This will help us later on to prove Lemma 1, the key step in proving Theorem 1. We use the following rules:

- **Rule $\mathcal{H}^{(2)}$** – Choose a random element $r \in \{0, 1\}^{\ell_i}$. If $i \in \{1, 2, 3, 4\}$, this query is directly asked by the adversary, and $(i, *, r) \in \Lambda_{\mathcal{A}}$, then we abort the game.

Then, for any r, $\#\{(i, *, r) \in \Lambda_{\mathcal{A}}\} \leq 1$. However, this rule may cause the game to abort with probability bounded by $\frac{q_h^2}{2^{\ell+1}}$.

For a hash-query $\mathcal{H}_i(q)$ (resp. with $\mathcal{H}_i'(q)$), such that a record (i, q, r) appears in $\Lambda_{\mathcal{H}}$ (resp. $\Lambda_{\mathcal{H}'}$), the answer is r. Otherwise, answer r is defined according to the following rule:

► **Rule $\mathcal{H}^{(1)}$** – Choose a random element $r \in \{0,1\}^{\ell_i}$.

The record (i, q, r) is added to $\Lambda_{\mathcal{H}}$. (resp. $\Lambda_{\mathcal{H}'}$). If the query is directly asked by the adversary, (i, q, r) is added to $\Lambda_{\mathcal{A}}$.

For an Encryption-query $\mathcal{E}_k(Z)$, such that a record $(k, Z, *, Z^\star)$ appears in $\Lambda_{\mathcal{E}}$, the answer is $Z^\star$.
Otherwise, answer $Z^\star$ is defined according to the following rule:

► **Rule $\mathcal{E}^{(1)}$** – Choose a random element $Z^\star \in \{0,1\}^{\ell_4}$.

The record $(k, Z, \mathcal{E}, Z^\star)$ is added to $\Lambda_{\mathcal{E}}$. If the query is directly asked by the adversary, $(k, Z, \mathcal{E}, Z^\star)$ is added to $\Lambda_{\mathcal{D}}$.

For a Decryption-query $\mathcal{D}_k(Z)$, such that a record $(k, Z, *, Z^\star)$ appears in $\Lambda_{\mathcal{E}}$, the answer is Z.
Otherwise, answer Z is defined according to the following rule:

► **Rule $\mathcal{D}^{(1)}$** – Choose a random element $Z^\star \in \{0,1\}^{\ell_4}$.

The record $(k, Z, \mathcal{D}, Z^\star)$ is added to $\Lambda_{\mathcal{E}}$. If the query is directly asked by the adversary, $(k, Z, \mathcal{D}, Z^\star)$ is added to $\Lambda_{\mathcal{D}}$.

Fig. 2. Simulation of the random oracles and encryption/decryption oracles

We answer the Send-queries to the server as follows:

– A Send(S^j, Start)-query is processed according to the following rule:

► **Rule S1**$^{(1)}$ – Choose a random element $\varphi \in \mathbb{Z}_q^*$ and compute $B = g^{\varphi}$.

Then, the query is answered with (S, B), and the server instance goes to an expected state.

– If instance S^j is in an expecting state, a query Send$(S^j, (ID^\star, A, M_U)$ is processed according to the following rule:

► **Rule S2**$^{(1)}$ – Compute $V_S = A^x$, $K_S = A^{\varphi}$, $ID = ID^\star \oplus \mathcal{H}_3(A||B||V_S||K_S)$, and $V_U = \mathcal{H}_4(x||ID||R)$.

► **Rule S3**$^{(1)}$ – If $M_U \neq \mathcal{H}_1(ID||S||A||B||V_U||V_S||K_S)$, the instance terminates without accepting.

► **Rule S4**$^{(1)}$ – Do nothing.

► **Rule S5**$^{(1)}$ – Compute authenticator $M_S = \mathcal{H}_2(ID||S||A||B||V_U||V_S||K_S)$ and session key $sk_S = \mathcal{H}_0(ID||S||A||B||V_U||V_S||K_S)$.

Finally, the query is answered with M_S, the instance accepts and terminates.

Our simulation also adds $((S, B), (ID^\star, A, M_U), M_S)$ to Λ_{Ψ}.

We answer the Send-queries to the client as follows:

– A Send$(U^i, (S, B))$-query is processed according to the following rule:

► **Rule U1**$^{(1)}$ – Choose a random element $\theta \in \mathbb{Z}_q^*$ and compute $A = g^{\theta}$.

► **Rule U2**$^{(1)}$ – Compute $V_U = \mathcal{D}_{pw}(V_U^\star)$, $V_S = y^{\theta}$, $K_U = B^{\theta}$, $ID^\star = ID \oplus \mathcal{H}_3(A||B||V_S||K_U)$, and the authenticator $M_U = \mathcal{H}_1(ID||S||A||B||V_U||V_S||K_U)$.

Then, the query is answered with $(ID^\star, A, M_U)$, and the client instance goes to an expected state.

Our simulation also adds $((S, B), (ID^\star, A, M_U), \perp)$ to Λ_{Ψ}.

– If instance U^i is in an expecting state, a query Send(U^i, M_S) is processed by computing the session key and producing an authenticator. We apply the following rules:

► **Rule U3**$^{(1)}$ – If $M_S \neq \mathcal{H}_2(ID||S||A||B||V_U||V_S||K_U)$, the instance terminates without accepting.

► **Rule U4**$^{(1)}$ – Do nothing.

► **Rule U5**$^{(1)}$ – Compute session key $sk_U = \mathcal{H}_0(ID||S||A||B||V_U||V_S||K_U)$.

The instance terminates with accepting.

An Execute(U^i, S^j)-query is processed successively using the simulations of the Send-queries: $(S, B) \leftarrow$ Send(S^j, Start), $(ID^\star, A, M_U) \leftarrow$ Send$(U^i, (S, B))$ and $M_S \leftarrow$ Send$(S^j, (ID^\star, A, M_U))$, and outputting the transcript $((S, B), (ID^\star, A, M_U), M_S)$.

A Reveal(I) -query returns the session key computed by the instance I (if the latter has accepted).

A Test(I)-query first gets sk from Reveal(I), and flips a coin b. If $b = 1$, we return the value of the session key sk. Otherwise, we return a random value drawn from $\{0,1\}^{\ell_0}$.

Fig. 3. Simulation of the queries

► **Rule $\mathcal{E}^{(2)}$** – Choose a random element $Z^{\star} \in \{0,1\}^{\ell_4}$. If $(*, Z, *, Z^{\star}) \in \Lambda_{\mathcal{E}}$, we abort the game.
► **Rule $\mathcal{D}^{(2)}$** – Choose a random element $Z \in \{0,1\}^{\ell_4}$. If $(*, Z, *, Z^{\star}) \in \Lambda_{\mathcal{E}}$, we abort the game.

Then, for any pair $(Z, Z^{\star})$, $\#\{(*, Z, *, Z^{\star}) \in \Lambda_{\mathcal{E}}\} \leq 1$. However, this rule may cause the game to abort with probability bounded by $\frac{q_{\mathcal{E}}^2}{2^{\ell_4+1}}$, where $q_{\mathcal{E}}$ is the size of $\Lambda_{\mathcal{E}}$.

► **Rule S1$^{(2)}$** – Choose a random element $\varphi \in \mathbb{Z}_q^*$ and compute $B = g^{\varphi}$. If $(*, B) \in \Lambda_S$, we abort the game. Otherwise, we add the record (j, B) to Λ_S. Variable Λ_S keeps track of the messages sent by server S.
► **Rule U1$^{(2)}$** – Choose a random element $\theta \in \mathbb{Z}_q^*$ and compute $A = g^{\theta}$. If $(*, A) \in \Lambda_U$, we abort the game. Otherwise, we add the record (i, A) to Λ_U. Variable Λ_U keeps track of the messages sent by client U.

Then, no collision occurs among the A outputted by the client instances and among the B outputted by the server instances. However, this rule may cause the game to abort with probability bounded by $\frac{(q_s+q_p)^2}{2(q-1)}$.

Games $\mathbf{G}_2$ and $\mathbf{G}_1$ are perfectly indistinguishable unless one of the above rules causes the game to abort:

$$|\Pr[\mathrm{S}_2] - \Pr[\mathrm{S}_1]| \leq \frac{q_h^2}{2^{\ell+1}} + \frac{q_{\mathcal{E}}^2}{2^{\ell_4+1}} + \frac{(q_s+q_p)^2}{2(q-1)}. \quad (3)$$

Game $\mathbf{G}_3$: We define game $\mathbf{G}_3$ by aborting the game wherein the adversary may guess the correct authenticator M_U or M_S (that is, without asking the corresponding hash-query $\mathcal{H}_1$ or $\mathcal{H}_2$). We achieve this objective by modifying the manner in which the participants process the queries:

► **Rule S4$^{(3)}$** – If $\big((S, B), (ID^{\star}, A, M_U), *\big) \notin \Lambda_{\Psi}$ and $(1, ID||S||A||B||V_U||V_S||K_S, M_U) \notin \Lambda_{\mathcal{A}}$, then we abort the game.
► **Rule U4$^{(3)}$** – If $\big((S, B), (ID^{\star}, A, M_U), M_S\big) \notin \Lambda_{\Psi}$ and $(2, ID||S||A||B||V_U||V_S||K_U, M_S) \notin \Lambda_{\mathcal{A}}$, then we abort the game.

This rule ensures that all accepted authenticators, M_U and M_S, are from either the simulator, or an adversary that has correctly computed V_U, and asked queried oracle $\mathcal{H}_1$ or $\mathcal{H}_2$. Games $\mathbf{G}_3$ and $\mathbf{G}_2$ are perfectly indistinguishable unless the server rejects a valid authenticator. Because neither A nor B appeared in a previous session (since game $\mathbf{G}_2$), this happens only if the authenticator had been correctly guessed by the adversary without asking $\mathcal{H}_1(ID||S||A||B||V_U||V_S||K_S)$ or $\mathcal{H}_2(ID||S||A||B||V_U||V_S||K_U)$:

$$|\Pr[\mathrm{S}_3] - \Pr[\mathrm{S}_2]| \leq \frac{q_s}{2^{\min\{\ell_1, \ell_2\}}}. \quad (4)$$

Game $\mathbf{G}_4$: We define game $\mathbf{G}_4$ by aborting the executions wherein the adversary guessed the correct parameter, V_U, without asking the corresponding query. We achieve this objective by modifying the manner in which the participants process the queries:

► **Rule S4$^{(4)}$** – If $((S, B), (ID^{\star}, A, M_U), *) \notin \Lambda_{\Psi}$ and either $(1, ID||S||A||B||V_U||*, M_U) \notin \Lambda_{\mathcal{A}}$ or $((*, *, V_U, *) \notin \Lambda_{\mathcal{D}}$ and $(4, * ||ID|| *, V_U) \notin \Lambda_{\mathcal{A}})$, then we abort the game.
► **Rule U4$^{(4)}$** – If $((S, B), (ID^{\star}, A, M_U), M_S) \notin \Lambda_{\Psi}$ and either $(2, ID||S||A||B||V_U||*, M_S) \notin \Lambda_{\mathcal{A}}$ or

$((*,*,V_U,*) \notin \Lambda_{\mathcal{D}}$ and $(4,* \,||ID||\, *,V_U) \notin \Lambda_{\mathcal{A}})$, then we abort the game.

Games $\mathbf{G}_4$ and $\mathbf{G}_3$ are perfectly indistinguishable unless $(1, ID||S||A||B||V_U||V_S||K_S, M_U) \notin \Lambda_{\mathcal{A}}$, $(1, ID||S||A||B||V_U||*, M_U) \notin \Lambda_{\mathcal{A}}$, $(2, ID||S||A||B||V_U||V_S||K_U, M_S) \notin \Lambda_{\mathcal{A}}$, or $(2, ID||S||A||B||V_U||*, M_S) \notin \Lambda_{\mathcal{A}}$. Because neither A nor B appeared in a previous session (since game $\mathbf{G}_2$), this happens only if parameter V_U had been correctly guessed by the adversary without asking $\mathcal{D}_{pw}(V_U^*)$ or $\mathcal{H}_4(x||ID||R)$:

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| \leq \frac{q_S}{2^{\ell_4}}. \tag{5}$$

Game $\mathbf{G}_5$: We define game $\mathbf{G}_5$ by aborting the executions wherein the adversary may have correctly computed parameter V_U, used it to build a valid authenticator M_U or M_S, and impersonate as a client or server. We achieve this objective by modifying the manner in which the participants process the queries:

- **Rule S4**$^{(5)}$ – If $((S,B),(ID^*,A,M_U),*) \notin \Lambda_\Psi$ and either $(1, ID||S||A||B||V_U||*, M_U) \notin \Lambda_{\mathcal{A}}$ or $\left((*,*,V_U,*) \notin \Lambda_{\mathcal{D}} \text{ and } \left(4,* \,||ID||\, *,V_U\right) \notin \Lambda_{\mathcal{A}}\right)$, then we abort the game. Further, if $((S,B),(ID^*,A,M_U),*) \notin \Lambda_\Psi$, we define the event Auth_5 to be true, and abort the game.
- **Rule U4**$^{(5)}$ – If $((S,B),(ID^*,A,M_U),M_S) \notin \Lambda_\Psi$ and either $(2, ID||S||A||B||V_U||*, M_S) \notin \Lambda_{\mathcal{A}}$ or $\left((*,*,V_U,*) \notin \Lambda_{\mathcal{D}} \text{ and } \left(4,* \,||ID||\, *,V_U\right) \notin \Lambda_{\mathcal{A}}\right)$, then we abort the game. Further, if $((S,B),(ID^*,A,M_U),M_S) \notin \Lambda_\Psi$, we define event Auth_5 to be true, and abort the game.

This rule ensures that all accepted authenticators, M_U and M_S, are from the simulator. Games $\mathbf{G}_5$ and $\mathbf{G}_4$ are perfectly indistinguishable unless $(*,*,V_U,*) \notin \Lambda_{\mathcal{D}}$ or $(4,* \,||ID||\, *,V_U) \notin \Lambda_{\mathcal{A}}$, which leads to event Auth_5' being true:

$$|\Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_4]| \leq \Pr[\mathsf{Auth}_5']. \tag{6}$$

Game $\mathbf{G}_6$: In this game, we replace random oracle $\mathcal{H}_i$ (with $i \in \{0,1,2,3\}$) with private oracles $\mathcal{H}_i'$. Because we no longer need to compute the values K_U, K_S, V_U, and V_S, we can also simplify the manner in which the participants process the queries:

- **Rule S2**$^{(6)}$ – Compute $ID = ID^* \oplus \mathcal{H}_3'(A||B)$.
- **Rule S3**$^{(6)}$ – If $M_U \neq \mathcal{H}_1'(ID^*||S||A||B)$, the instance terminates without accepting.
- **Rule S5**$^{(6)}$ – Compute authenticator $M_S = \mathcal{H}_2'(ID^*||S||A||B)$ and session key $sk_S = \mathcal{H}_0'(ID^*||S||A||B)$.
- **Rule U2**$^{(6)}$ – Compute $ID^* = ID \oplus \mathcal{H}_3'(A||B)$ and authenticator $M_U = \mathcal{H}_1'(ID^*||S||A||B)$.
- **Rule U3**$^{(6)}$ – If $M_S \neq \mathcal{H}_2'(ID^*||S||A||B)$, the instance terminates without accepting.
- **Rule U5**$^{(6)}$ – Compute session key $sk_U = \mathcal{H}_0'(ID^*||S||A||B)$.

Games $\mathbf{G}_6$ and $\mathbf{G}_5$ are indistinguishable unless the following event, AskH_6, occurs: $\mathcal{A}$ queries the hash functions $\mathcal{H}_1$ or $\mathcal{H}_2$ on $ID||S||A||B||V_U||V_S||K_U$ or on $ID||S||A||B||V_U||V_S||K_S$ that is on $ID||S||A||B||V_U||V_S||\mathsf{CDH}(A,B)$:

$$|\Pr[\mathsf{Auth}_6'] - \Pr[\mathsf{Auth}_5']| \leq \Pr[\mathsf{AskH}_6], \qquad |\Pr[\mathsf{S}_6] - \Pr[\mathsf{S}_5]| \leq \Pr[\mathsf{AskH}_6]. \tag{7}$$

Lemma 1. *The probabilities of the events S_6 and $Auth_6'$ in game G6 can be upper-bounded by the following values:*

$$\Pr[\mathsf{S}_6] = \frac{1}{2}, \qquad \Pr[\mathsf{Auth}_6'] \leq \frac{3q_s}{2^{\min\{\ell_1,\ell_2\}}} + \frac{q_s}{2N}. \tag{8}$$

Proof. In game $\mathbf{G}_6$, the session keys are computed with private hash oracles unknown to $\mathcal{A}$, and thus $\Pr[\mathsf{S}_6] = \frac{1}{2}$. Let us denote by $R(S)$ the set of M_U received by a server instance, and by $R(U)$ the set of M_S received by a client instance. Because we have avoided cases where $\mathcal{A}$ correctly guessed V_U, $\mathcal{A}$ can correctly compute V_U using $\mathsf{Corrupt}(U^i, 1)$, $\mathsf{Corrupt}(U^i, 2)$, $\mathsf{Corrupt}(U^i, 3)$, or $\mathsf{Corrupt}(S^j, 4)$ -queries with probability denoted by $\Pr[\mathsf{Auth}_6\mathsf{Corr}_1']$, $\Pr[\mathsf{Auth}_6\mathsf{Corr}_2']$, $\Pr[\mathsf{Auth}_6\mathsf{Corr}_3']$, and $\Pr[\mathsf{Auth}_6\mathsf{Corr}_4']$, respectively. From an information theoretical point of view, since we have avoided collisions in game $\mathbf{G}_2$,

$$\begin{aligned}
&\Pr[\mathsf{Auth}_6'] = \Pr[\mathsf{Auth}_6\mathsf{Corr}_1'] + \Pr[\mathsf{Auth}_6\mathsf{Corr}_2'] + \Pr[\mathsf{Auth}_6\mathsf{Corr}_3'] + \Pr[\mathsf{Auth}_6\mathsf{Corr}_4'],\\
&\Pr[\mathsf{Auth}_6\mathsf{Corr}_1'] \leq \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}},\\
&\Pr[\mathsf{Auth}_6\mathsf{Corr}_2'] \leq \Pr_{pw}\left[\exists M_U \in R(S), V \leftarrow \mathcal{D}_{pw}(V^*), (1, ID||S||A||B||V_U||*, M_U) \notin \Lambda_{\mathcal{A}}\right]\\
&\qquad\qquad + \Pr_{pw}\left[\exists M_S \in R(U), V \leftarrow \mathcal{D}_{pw}(V^*), (2, ID||S||A||B||V_U||*, M_S) \notin \Lambda_{\mathcal{A}}\right]\\
&\qquad\qquad \leq \frac{\#R(S)+\#R(U)}{N},\\
&\Pr[\mathsf{Auth}_6\mathsf{Corr}_3'] \leq \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}},\\
&\Pr[\mathsf{Auth}_6\mathsf{Corr}_4'] \leq \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}}.
\end{aligned}$$

By definition of the set $\#R(S)$ and $\#R(U)$, since M_U is received in the second query to the server, and M_S is received in the second query to the client, the cardinalities are both upper-bounded by $q_s/2$. □

Game $\mathbf{G}_7$: In this game, we simulate the executions using the random self-reducibility of the Diffie-Hellman problem, given one CDH instance (P, Q). We do not need to know the values of θ and φ, since the values K_U and K_S are no longer needed to compute the authenticators and the session keys:

- **Rule S1**$^{(7)}$ − Choose a random element $\beta \in \mathbb{Z}_q^*$. Compute $B = g^\beta$ and add record (β, B) to Λ_B.
- **Rule U1**$^{(7)}$ − Choose a random element $\alpha \in \mathbb{Z}_q^*$. Compute $A = g^\alpha$ and add record (α, A) to Λ_A.

$$\Pr[\mathsf{AskH}_7] = \Pr[\mathsf{AskH}_6]. \tag{9}$$

Remember that AskH_7 signifies adversary $\mathcal{A}$ had queried random oracles $\mathcal{H}_0$, $\mathcal{H}_1$, or $\mathcal{H}_2$ on $ID||S||A||B||V_U||V_S||\mathsf{CDH}(A, B)$. By choosing randomly in the Λ_A-list we can obtain the Diffie-Hellman secret value with probability $1/q_h$. This is a triple $(A, B, \mathsf{CDH}(A, B))$. We can then simply look in lists Λ_A and Λ_B to find values α and β such that $A = g^\alpha$ and $B = g^\beta$:

$$\mathsf{CDH}(A, B) = \mathsf{CDH}\left(P^\alpha, Q^\beta\right) = \mathsf{CDH}(P, Q)^{\alpha\beta}.$$

Thus:

$$\Pr[\mathsf{AskH}_7] \leq q_h \mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(t'), \tag{10}$$

where $t' \leq t + (q_s + q_p + 1) \cdot \tau_{\mathbb{G}}$.

Conclusion of the proof: The proof is completed by summing all the relations. Simply note that $q_{\mathcal{E}}$ is the size of $\Lambda_{\mathcal{E}}$, which contains all the encryption/decryption-queries directly asked by the adversary, and all the decryption-queries made by our simulation: at most one per Send-query (direct or through Execute-queries), which gives $q_{\mathcal{E}} \leq q_s + q_p + q_e$. From Equations (1)-(7),

$$\begin{aligned}|\Pr[\mathsf{S}_6] - \Pr[\mathsf{S}_0]| &\leq \frac{q_h^2}{2^{\ell+1}} + \frac{q_{\mathcal{E}}^2}{2^{\ell_4+1}} + \frac{(q_s+q_p)^2}{2(q-1)} + \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}} + \frac{q_s}{2^{\ell_4}} + \Pr[\mathsf{Auth}_5'] + \Pr[\mathsf{AskH}_6] \\ &\leq \frac{q_h^2+(q_s+q_p+q_e)^2+4q_s}{2^{\ell+1}} + \frac{(q_s+q_p)^2}{2(q-1)} + \Pr[\mathsf{Auth}_6'] + 2\Pr[\mathsf{AskH}_6].\end{aligned}$$

From Equations (8)-(10), we get

$$\Pr[\mathsf{Auth}_6'] \leq \frac{3q_s}{2^{\ell}} + \frac{q_s}{2N}, \quad \Pr[\mathsf{AskH}_6] \leq q_h \mathsf{Succ}_{\mathbb{G}}^{\mathsf{cdh}}(t'), \tag{10}$$

which concludes the proof. □

A.2 Proof of Theorem 2

Proof. We can use the same proof presented in the previous section here, because

$$\mathrm{Adv}_{\mathcal{P}}^{\mathrm{auth}}(\mathcal{A}) = \Pr[\mathsf{Auth}_0],$$

and see that in game $\mathbf{G}_5$, $\Pr[\mathsf{Auth}_5] = 0$, and Equations (2) to (7) extend to

$$\begin{aligned}\Pr[\mathsf{Auth}_1] &= \Pr[\mathsf{Auth}_0], \\ |\Pr[\mathsf{Auth}_2] - \Pr[\mathsf{Auth}_1]| &\leq \frac{q_h^2}{2^{\ell+1}} + \frac{q_{\mathcal{E}}^2}{2^{\ell_4+1}} + \frac{(q_s+q_p)^2}{2(q-1)}, \\ |\Pr[\mathsf{Auth}_3] - \Pr[\mathsf{Auth}_2]| &\leq \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}}, \\ |\Pr[\mathsf{Auth}_4] - \Pr[\mathsf{Auth}_3]| &\leq \frac{q_s}{2^{\ell_4}}, \\ |\Pr[\mathsf{Auth}_5] - \Pr[\mathsf{Auth}_4]| &\leq \Pr[\mathsf{Auth}_5'].\end{aligned}$$

Then, we get

$$\mathrm{Adv}_{\mathcal{P}}^{\mathrm{auth}}(\mathcal{A}) \leq \frac{q_h^2}{2^{\ell+1}} + \frac{q_{\mathcal{E}}^2}{2^{\ell_4+1}} + \frac{(q_s+q_p)^2}{2(q-1)} + \frac{q_s}{2^{\min\{\ell_1,\ell_2\}}} + \frac{q_s}{2^{\ell_4}} + \Pr[\mathsf{Auth}_6'] + \Pr[\mathsf{AskH}_6'],$$

which concludes the proof, using Equation (11). □

Hanwook Lee received the B.E. degree in Computer Engineering from Kyungpook National University, Korea, in 1996 and the M.S. degree from POSTECH, Korea, in 1998. Since 1998 he has been a researcher at Korea Financial Telecommunications and Clearings Institute. He is currently a Ph.D. student at Sungkyunkwan University, Korea. His research interests include cryptography and information security.

Junghyun Nam received the B.E. degree in Information Engineering from Sungkyunkwan University, Korea, in 1997. He received his M.S. degree in Computer Science from University of Louisiana, Lafayette, in 2002, and the Ph.D. degree in Computer Engineering from Sungkyunkwan University, Korea, in 2006. He is now a professor in Konkuk University, Korea. His research interests include cryptography and computer security.

Moonseong Kim received the M.S. degree in Mathematics, August 2002 and the Ph.D. degree in Electrical and Computer Engineering, February 2007 both from Sungkyunkwan University, Korea. He was a research professor at Sungkyunkwan University in 2007. From December 2007 to October 2009, he was a visiting scholar in ECE and CSE, Michigan State University, USA. Since October 2009, he has been a patent examiner in Information and Communication Examination Bureau, Korean Intellectual Property Office (KIPO), Korea. His research interests include wired/wireless networking, sensor networking, mobile computing, network security protocols, and simulations/numerical analysis.

Dongho Won received BSC, MSC and PhD in Electronic Engineering from Sungkyunkwan University, South Korea. After working in Electronics and Telecommunication Research Institute for two years, he joined Sungkyunkwan University, where he is currently a leader professor at Information and Communication Engineering. He also served as a President of Korea Institute of Information Security and Cryptography. His research interests are cryptology and information security.