A Pairing-Free, One Round Identity Based Authenticated Key Exchange Protocol Secure Against Memory-Scrapers*

Suvradip Chakraborty†, Srinivasan Raghuraman, and C. Pandu Rangan
Dept. of Computer Science and Engineering
Indian Institute of Technology, Madras, India
suvradip@cse.iitm.ac.in, srini131293@gmail.com, prangan55@gmail.com

Abstract

Security of a key exchange protocol is formally established through an abstract game between a challenger and an adversary. In this game the adversary can get various information which are modeled by giving the adversary access to appropriate oracle queries. Empowered with all these information, the adversary will try to break the protocol. This is modeled by a test query which asks the adversary to distinguish between a session key of a *fresh* session from a random session key; properly guessing which correctly leads the adversary to win the game. In this traditional model of security the adversary sees nothing apart from the input/ output relationship of the algorithms. However, in recent past an adversary could obtain several additional information beyond what he gets to learn in these black box models of computation, thanks to the availability of powerful malwares. This data exfiltration due to the attacks of Memory Scraper/Ram-Scraper-type malwares is an emerging threat. In order to realistically capture these advanced classes of threats posed by such malwares we propose a new security model for identity-based authenticated key exchange (ID-AKE) which we call the Identity based Strong Extended Canetti Krawzyck (ID-seCK) model. Our security model captures leakages of intermediate values by appropriate oracle queries given to the adversary. Following this, we propose a round optimal (i.e., single round) ID-AKE protocol for two-party settings. Our design assumes a hybrid system equipped with a bare minimal Trusted Platform Module (TPM) that can *only* perform group exponentiations. One of the major advantages of our construction is that it does not involve any pairing operations, works in prime order group and have a tight security reduction to the Gap Diffie Hellman (GDH) problem under our new ID-seCK model. Our scheme also has the capability to handle active adversaries while most of the previous ID-AKE protocols are secure only against passive adversaries. The security of our protocol is proved in the Random Oracle (RO) model.

Keywords: Authenticated Key Exchange, Identity-based Authenticated Key Exchange (ID-AKE), Intermediate values, ID-seCK model, Ram Scraper

1 Introduction

Key exchange is a fundamental problem in cryptography. Key exchange protocols allow two or more parties to securely communicate over an adversarially controlled network by establishing shared keys between them. *Authenticated key exchange* (AKE) protocols allow both the parties not only to establish a common shared secret key between them, but also allows them to mutually authenticate each other with the assurance that the key is known only to them.

Diffie and Hellman in [1] proposed the first Key Agreement Protocol. This was a public key based system where every party has a pair of keys, the public key and the private key. However the basic

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 7:1 (Mar. 2016), pp. 1-22

*A portion of overlap of the paper is taken from our earlier work and some of the standard definitions and some statements are taken from few other papers, which are all properly cited in this version.

†Corresponding author: Department of Computer Science and Engg., Indian Institute of Technology, Madras, Chennai, Tamil Nadu-600036, India, Tel: +91-9003641807

Diffie Hellman protocol is susceptible to an *active* adversary who can tamper with the messages sent across by the parties and can launch man-in-the-middle attacks. This marked the advent of AKE where certificates signed by a trusted party would be used to authenticate a party. The parties involved are mandated to obtain and verify certificates whenever the parties require to establish a shared key among themselves. This brings in the issue of public key certificate management with regards to revocation and updation resulting in excess overhead for certificate management. These drawbacks were addressed in identity-based cryptography.

The concept of Identity based Cryptography was put forward by Shamir [2] in 1984. In an identity based key agreement (IDKA) protocol, each user is registered under a Private Key Generator (PKG) which is assumed to be a trusted authority. Each user in the system has an identity (which is nothing but a bit string) that uniquely identifies it among other users in the system. This can be for instance his email address, social security number or other attributes which uniquely identifies him/her. An user in the system submits its own identity to the PKG and obtains his/her private key from the PKG. It then communicates with other users in the system and runs a key agreement protocol, which is essentially an interactive protocol and generates a shared secret key as outcome of the protocol. A major advantage of identity based paradigm over public key settings is that any user can communicate with another user in the system only using his/her identity. So this reduces the overhead of certificate management as in public key based systems. Besides it also provides significant advantage in terms of bandwidth savings as there is no need to transfer public key certificates signed by trusted certification authorities while executing the protocol. Hence key agreement protocols in identity based paradigm are more preferred rather than their corresponding public key based counterparts, which motivates further design and analysis of ID-AKE protocols.

The existing security models of key exchange protocols can be broadly classified into two categories: 1. *Black-box models*, which assumes all computations and storage are in trusted servers and no part of it is accessible to the adversary, and 2. *Hybrid models*, which assumes that only the top level secret information would be stored in a trusted server and only computations related to it/them would be done in the trusted server; rest of the intermediate values may be accessible to the adversary. Naturally this hybrid model of computation considers more powerful classes of adversarial attack models and captures the fact that the adversary can get intermediate values generated and stored as the computation progresses. Typically the Long Term Key (LTK) of a party is its long term secret. So it makes sense to keep it more protected by implementing adequate access control mechanisms typically by storing it in trusted modules; while the other session specific secret keys/values may not be given that level of access protection, so they may not be stored in trusted modules. Canetti-Krawzyck (CK) [3] or extended Canetti-Krawczyk (eCK) [4] (both for PKI based systems) are examples of the black-box models while the seCK model [5] is an example of hybrid model (again for PKI based systems). However the security models under which all the previous IDKA protocols were analyzed are not sufficient to safeguard against more sophisticated threats which involves leakage of intermediate computation values to an adversary. The security of many of these protocols breaks down completely when some of the intermediate values that are computed or stored in the untrusted host machine are available to the attacker. This have turned out to be a serious practical threat due to Ram-Scraper or Memory-Scraper-like malwares. It is a piece of data-harvesting malware [6] that collects data from volatile memory. Rather than taking a whole memory snapshot, these often use stealthy techniques, such as, hooking into a payment processing application and selectively dumping data that matches certain patterns, e.g. the regular expression of a credit card format from a specific memory region. The point-of-sales (PoS) terminals have become a juicy attack vector to these malwares. It becomes even severe when the intermediate values are leaked in the course of an execution of a cryptographic algorithm/ protocol enabling the attacker to successfully trespass the internal state of an algorithm and gain valuable information making the protocol vulnerable. In the white-paper made

available by VISA at [7], VISA brings to attention the serious threats posed by memory parsers. For more elaborate discussion on this malware see [6]. Motivated by these threats posed by such malwares, we re-examine the problem of identity based key agreement protocols under these stronger threat models.

2 Related Works

Following the notion of identity based cryptography put forward by Shamir, a number of IDKA protocols were proposed in this paradigm. However most of them involved expensive pairing operations which makes the practical implementation of those protocol(s) somewhat inefficient. In general it is always desirable to have a protocol that involves simple group theoretic operations rather than pairing as it is slightly inefficient to find many pairing-friendly curves. The documentation of MIRACL [8], the cryptographic library, discusses the costs associated with various cryptographic primitives. The values show that the cost associated with point multiplication in the group $\mathbb{G}_1$ of a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is roughly faster than the pairing operation by a factor of 3:8.

Gunther [9] and Saeednia [10] proposed pairing-free IDKA protocols. However both of them lacked formal proof of security. Fiore [11] then proposed a IDKA protocol without pairing which was formally proved in the Canetti-Krawczyk [3] model. Besides it was also much more efficient compared to the protocols of [9] and Saeednia [10] in terms of computational complexity. However the protocol in [11] was analyzed and vulnerabilities were pointed out in [12] and in [13]. Another pairing-free IDKA protocol was proposed by Cao [14]. However Islam *et al* [15] identified the vulnerabilities of the protocol to key-offset attack and known session specific secret information attack. An improvement on this was done by [16]. The previous works on identity based protocols (except [16]) considered security against passive adversaries. An active adversary is much strong; in particular it can tamper with the components exchanged between them and modify them arbitrarily during transit. [16] was formally proven secure in the CK [3] model and the authors claim their protocol is resistant to ephemeral key compromise attacks which is captured by eCK model [4]. In the CK and eCK models, an adversary can obtain session state information via appropriate reveal queries on session state and ephemeral keys respectively. This captures the fact that the adversary can obtain leakages on the session specific values or session state. CK model allows the adversary to access session state information; however it does not explicitly specify the contents of a session state and leave it to the protocol designers to explicitly state it according to the implementation of the protocol. Cremers took advantage of the vagueness in the **Session State Reveal** query of the CK model to reveal the intermediate values and showed that the NAXOS protocol which is shown to be secure in eCK model is in fact insecure in the CK model. Fujioka *et al.* [17] proposed an IDAKE in id-CK$^+$ model which is the ID based analogue of CK$^+$ model using a CCA secure ID based key Encapsulation Mechanism (ID-KEM), a CPA-secure ID-KEM and a secure Key derivation Function (KDF). The security proof is given in standard model. However, *RAM-scrapers* and all memory-scraper-like malware can obtain leakages on intermediate results produced while computing session keys. Since these kind of leakages are not covered by black box security models like the (e)CK model, a protocol shown to be secure in eCK model can easily be shown vulnerable in the face of such leakage and they will fail in authentication. [5] considers a hybrid security model similar to our model called the *strengthened* eCK (seCK) model. This model for key exchange was a hybrid security model which considered leakages on intermediate results in computing session keys and hence easily encompasses the black box models. However like the CK and the eCK models, the seCK model is also tailor made for public key systems. A concept similar to this was introduced in [18] for public key encryption which they referred to as the Glass Box model.

Another line of research that has emerged in the recent years is called *leakage-resilient* cryptography which aims to resist side channel attacks [19], [20], [21], [22] by allowing the adversary to obtain sig-

nificant fraction of the secret key related information either in Bounded Leakage or Continuous Leakage models [23],[24], [25]. This motivated the development of leakage-resilient key exchange. Moriyama and Okamoto [26] first provided a formalism of leakage resilient key exchange and proposed a two-pass protocol in bounded leakage model. Alawatugoda et. al [27] proposed a generic construction for leakage-resilient key exchange. It can be instantiated either in bounded leakage or continuous leakage model. However their generic construction cannot be instantiated in continuous leakage model due to absence of suitable cryptographic primitives. In an attempt to solve this problem they later introduced Continuous After The Fact Leakage (CAFL) model [28] which is a weaker variant of the ASB model [27]. Although these schemes talk of providing security guarantees even in the presence of side channel attacks, it is to be noted that they do not consider security against exposure of values generated as a result of intermediate computations. So even for a key exchange protocol that is leakage resilient, RAM Scrapers and in general Memory Scraper classes of malwares pose a devastating threat. It is to be noted that in our model, secret keys are stored in the TPM and no part of the secret key is available to the adversary. Thus, ours is very *different* from the leakage models and we will not compare them with our protocol or model.

2.1 Our Contribution

Through this paper, we aim to make contributions along the following lines:

1. **New Security Model.** We propose a new security model for ID-AKE which we call ID-seCK model. Our new security model gives a platform to analyze much stronger and wider classes of attacks where the attacker can obtain intermediate values generated as a result of computation which were not considered in the traditional black box security models for ID-AKE protocols. Our new model is particularly suited for hybrid implementations where only some part of the computation is done in TPM and the rest of it is done in untrusted host. Our security model can be viewed as an identity based analogue of the security model proposed in [5] for public key settings.

2. **New Construction.** We propose a new ID-AKE protocol that is secure in our new ID-seCK security model. Our protocol is the first identity based key exchange protocol that withstands leakages on intermediate computations and satisfies our ID-seCK security definitions. A major advantage of our protocol is that it is *pairing-free*. Beside our protocol is also *round-optimal* in the sense that is a *single* round protocol consisting of one single pass (or message flow) per party. Our ID-AKE protocol is proven secure under the Gap-Diffie Hellman (GDH) assumption in the Random Oracle (RO) model. Our security proof is also tight since it does not involve the use of forking lemma. This makes the practical implementation of our protocol to be efficient since we can work with smaller group sizes.

3. **Resilience to Active Adversaries.** Another attribute of our protocol is that it is also resilient to *active* attacks. Resisting active attacks in a single round is significantly challenging. Most of the previous IDKA protocols were secure against passive attackers. Our proposed protocol overcome this limitation by incorporating appropriate verification mechanisms that would abort the protocol in case of any change in values to be agreed upon. We ensure this by including a term which is a *signature* of the ephemeral keys. This provides enhanced security for minimal extra overhead.

Protocols	Rounds	Security Reduction	Exponentiations	Communication Cost (Elliptic Curve Group)	Communication Cost (Multiplicative Group)
Gunther [9]	2	Not tight	4	$2 \cdot (8 \cdot \mathbb{G}_B) = 2 \cdot (8 \cdot 224) = 3584$	$2 \cdot (8 \cdot \mathbb{G}_B) = 2 \cdot (8 \cdot 2048) = 32768$
Saeednia [10]	1	Not tight	3	$1 \cdot (8 \cdot \mathbb{G}_B) = 1 \cdot (8 \cdot 224) = 1792$	$1 \cdot (8 \cdot \mathbb{G}_B) = 1 \cdot (8 \cdot 2048) = 16384$
Fiore [11]	1	Not tight	2	$1 \cdot (8 \cdot \mathbb{G}_B) = 1 \cdot (8 \cdot 224) = 1792$	$1 \cdot (8 \cdot \mathbb{G}_B) = 1 \cdot (8 \cdot 2048) = 16384$
Proposed protocol	1	Tight	4	$2 \cdot (\mathbb{G}_B) + 1 \cdot (224) = 2 \cdot 224 + 1 \cdot 224 = 672$	$2 \cdot (\mathbb{G}_B) + 1 \cdot (224) = 2 \cdot 2048 + 1 \cdot 224 = 4320$

Table 1 : Comparing Efficiency

Scheme	Reflection Attacks	KCI	Active Adversary	IR leakage Resilience
Gunther [9]	✗	✓	✗	✗
Saeednia [10]	✓	✓	✗	✗
Fiore [11]	✓	✓	✗	✗
Vivek [16]	✓	✓	✓	✗
Proposed protocol	✓	✓	✓	✓

Table 2 : Comparing Security Properties

Tables 1 and 2 shows the comparison of our scheme with the existing schemes in terms of efficiency and security attributes respectively. The terms KCI stands for Key Compromise Impersonation attacks, and IR stands for Intermediate results. Here $\mathbb{G}_B$ denotes the number of bits needed to represent a group element. According to the recommendations in [29] and also specified in [16], we set $\mathbb{G}_B = 224$ bits for operations over elliptic curve groups and $\mathbb{G}_B = 1024$ for operations over multiplicative groups. As shown in [30] the security parameter increases by a factor of 2 if the security reduction uses forking lemma. In any discrete log based system of a prime field $\mathbb{Z}_q$, a factor α increase in the security parameter implies a factor α^3 increase in the size of the modulus q. Thus the size of the modulus increases by a factor of 8.

3 Preliminaries and Assumptions

3.1 Notations.

We denote the security parameter by κ. The set of integers is denoted by $\mathbb{Z}$ and $\mathbb{N}$ denotes the set of natural numbers. $[n]$ denotes the set $\{1,\ldots,n-1\}$ for $n \geq 2$. We denote by $x \in_R X$ the fact that x is chosen uniformly at random from the set of values X. $\mathbb{G}$ denotes a group of order q where $2^{\kappa-1} < q < 2^{\kappa}$ and let g denote the generator of the group $\mathbb{G}$. $\mathbb{Z}_p^*$ denotes the multiplicative group of integers modulo p, where p is a prime and $(p-1)|q$. We use $\mathbb{G}_B$ to denote the size of a group element in terms of the number of bits.

3.2 Preliminaries for our construction.

Here we give a brief overview of two signature schemes namely FXCR-1 and FDCR-1 signatures since these will be required for our construction as in [5].

Definition 3.1. (FXCR-1 Signatures). *Let $B \in \mathbb{Z}_p^*$ be the public key of a party* B *and let* A *be a verifier. Let $H : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function. Let $Y = g^y$, where $y \in_R [p]$ is chosen by* B*, and $s_B = ye + b$, where $e = H(Y, X, m)$. Party* A *also chooses $x \in_R \mathbb{Z}_p^*$ and computes $X = g^x$. It then gives X to*

B *as challenge and keeps x as secret.* B *then takes a message m and the challenge X and signs on m and X as* $Sig_{\mathsf{B}}(m,X) = (Y, X^{s_B})$. A *accepts the pair* (Y,σ_B) *as a valid signature if* $(Y^eB)^x = \sigma_B$.

Proposition 3.1. (FXCR-1 Security). *Suppose there is an adaptive probabilistic polynomial time (PPT) adversary* $\mathscr{A}$ *that can forge an FXCR-1 signature; then using that adversary we can solve CDH problem. More precisely, given a public key* B^**, a challenge* X^**, and access to signing and hashing oracles, the probability that the adversary comes up with a triple* (m^*, Y^*, σ^*) *as forgery such that the following conditions are satisfied is negligible under the CDH assumption in the RO model:*

1. The signature (Y^*, σ^*) passes the verification algorithm with respect to the public key B^*, and the message-challenge pair (m^*, X^*); and
2. The adversary $\mathscr{A}$ did not explicitly query the signing oracle with the pair (m^*, X^*) to get (Y^*, σ^*)

Definition 3.2. (FDCR-1 Signatures). *Let* $A = g^a$ *and* $B = g^b$ *be the public keys of two parties* A *and* B *respectively and let* m_1 *and* m_2 *denote two messages. Let* $H : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ *be a collision-resistant hash function. Let* $X = g^x, Y = g^y$*, where* $x, y \in_R [p]$ *are chosen by* A *and* B *respectively, and let* $d = H(X,Y,m_1,m_2)$*, and* $e = H(Y,X,m_1,m_2)$*. The dual signatures of* A *and* B *on the messages* m_1 *and* m_2 *are* $DSig_{\mathsf{A},\mathsf{B}}(m_1,m_2,X,Y) = (X^dA)^{ye+b} = (Y^eB)^{xd+a}$.

Proposition 3.2. (FDCR-1 Security). *Suppose there is an adaptive probabilistic polynomial time (PPT) adversary* $\mathscr{A}$ *that can forge an FDCR-1 signature; then using that adversary we can solve CDH problem. More precisely, given given* a^*, A^*, B^*, X^**, a message* m_1^* *and access to hashing and signing oracles, the probability that the adversary comes up with a triple* (m_2^*, Y^*, σ^*) *as forgery such that the following conditions are satisfied is negligible under the CDH assumption in the RO model:*

1. The signature (m_2^*, Y^*, σ^*) passes the verification algorithm with respect to the public key B^*, and the message-challenge pairs (m_1^*, X^*) and
2. The adversary $\mathscr{A}$ did not explicitly query the signing oracle with the pair (m_1', X') to obtain (Y^*, σ^*) such that $X^* = X'$ and $m_1'||m_2' = m_1^*||m_2^*$, where m_2' is obtained by querying the signing oracle on (m_1', X') (here $m_1^*||m_2^*$ denotes the concatenation of m_1^* and m_2^*).

3.3 Complexity Assumption.

In this section we present the complexity assumptions required for our construction as in [16].

Definition 3.3. Computation Diffie-Hellman Problem (CDH) - Given $(g, g^a, g^b) \in_R \mathbb{G}^3$ for unknown $a, b \in_R \mathbb{Z}_q^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with g as a generator and q the order of the group, the CDH problem in $\mathbb{G}$ is to compute g^{ab}.

The advantage of any probabilistic polynomial time algorithm $\mathscr{A}$ in solving the CDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathscr{A}}^{CDH} = Pr\left[\mathscr{A}(g, g^a, g^b) = g^{ab} \mid a, b \in_R \mathbb{Z}_q^*\right]$$

The *CDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathscr{A}$, the advantage $Adv_{\mathscr{A}}^{CDH}$ is negligibly small.

Definition 3.4. Decisional Diffie-Hellman Problem (DDH) - Given $(g, g^a, g^b, h) \in \mathbb{G}^4$ for unknown $a, b \in_R \mathbb{Z}_q^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with g as a generator and q the order of the group, the DDH problem in $\mathbb{G}$ is to check whether $h \stackrel{?}{=} g^{ab}$.

The advantage of any probabilistic polynomial time algorithm $\mathscr{A}$ in solving the DDH problem in $\mathbb{G}$ is defined as

$$Adv_{\mathscr{A}}^{DDH} = |Pr\left[\mathscr{A}(g, g^a, g^b, g^{ab}) = 1\right] - Pr\left[\mathscr{A}(g, g^a, g^b, g^c) = 1\right]| \mid a, b, c \in_R \mathbb{Z}_q^*$$

The *DDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathscr{A}$, the advantage $Adv_{\mathscr{A}}^{DDH}$ is negligibly small.

Definition 3.5. Gap Diffie Hellman Problem (GDH). Given $(g, g^a, g^b) \in_R \mathbb{G}^3$ and access to a Decision Diffie Hellman (DDH) oracle $DDH(\cdot,\cdot,\cdot)$ which on input g^a, g^b and g^c outputs `True` if and only if $c = ab$, the Gap Diffie Hellman problem is to compute $g^{ab} \in \mathbb{G}$.

The advantage of an adversary $\mathscr{A}$ in solving the Gap Diffie Hellman problem is defined as

$$\mathsf{Adv}_{\mathscr{A}}^{GDH} = \Pr\left[\mathscr{A}^{DDH(\cdot,\cdot,\cdot)}(g, g^a, g^b) = g^{ab}\right]$$

The Gap Diffie Hellman *Assumption* holds in $\mathbb{G}$ if for all polynomial time adversaries $\mathscr{A}$, $\mathsf{Adv}_{\mathscr{A}}^{GDH}$ is negligible.

4 Security model

In this section we describe our new security model for Identity based key agreement protocols. Each entity participating in an IDKA protocol is uniquely identified by a identity (bit) string ID_i corresponding to the i^{th} party. The PKG generates the master public key or public parameters and a master secret key. It then makes the master public key as system wide public parameters and keeps master secret key only to itself. The PKG uses its master secret key to generate the private key S_i for each user i. An IDKA protocol comprises of the following algorithms:

1. ***Setup:*** This algorithm is run by the PKG. It takes a security parameter as input and generates public parameters/ master public key of the system and a master secret key. The public parameters are available publicly to all users in the system and the master secret key is kept secret by the PKG.
2. ***Key Generation:*** This algorithm is an *interactive* protocol between the user and the PKG. Each user in the system submits its identity string to the PKG. The PKG verifies the identity of each user via some verification mechanism and uses its master secret key and the public parameters of the system to generate a secret key or private key S_i corresponding to user with identity ID_i. The private key is then transmitted to the user via a secure channel.
3. ***Key Agreement:*** This is also an *interactive* protocol running between two parties with identities ID_A and ID_B having secret keys S_A and S_B respectively. They exchange messages between them and at the end of this protocol both the parties establish a shared secret key. Here any one of the two users could initiate the protocol.

   An instance of the protocol when run at a party is called a *session*.A party can run multiple instances of a protocol simultaneously with different parties in the system. Associated with each session is the *owner* who initiates the session and the other party is called the *peer*. Both the parties exchange some messages or components between them. Each party uses its secret key, local randomness and the components received from the other party to establish a shared key corresponding to that session at the end. All the components exchanged between the parties and their local randomness constitute the *session state* for that particular session. The fundamental requirement of a key exchange protocol is that at the end both the parties involved should compute the *same* session key. If the session is successfully completed, each party outputs its shared session key and also *erases* its session state. Otherwise, both the parties aborts at some point of execution of the protocol and in this case no session key is established between them. A session is uniquely identified by an identifier. For e.g. if ID_A communicates with another party ID_B, it sets the session identifier of that session to be (ID_A, ID_B, out, in), where *out* and *in* denotes the messages sent to ID_B and received from ID_B respectively.

Adversary. The adversary $\mathscr{A}$ is also modeled as a Probabilistic Polynomial Turing Machine (PPTM) which has full control on the communication network over which protocol messages can be altered, deleted, injected, rescheduled or eavesdropped at any time. Here we assume each party is working in a hybrid environment which consists of a standard computing device (that are untrusted) called the *untrusted host* and a *Trusted Computing/Platform Module* (TPM). In the untrusted host all the normal world computations are done and in the trusted computing module or tamper-proof device only the top level secrets like the secret keys of users, .i.e, Long term secret keys (LTK) and session specific secret key (or the ephemeral secret keys) of users may be stored. All the computations that directly involve the secret keys are done in the TPM. For example, while executing the algorithm for generating the session key at a party some steps may not involve direct involvement of the secret key values and some steps

that may involve secret key values. The computation of the first kind will be done in the untrusted host machine or CPU and the resulting values will be available in the untrusted host. The computations of the second kind will be carried out in the TPM. We assume that all the values that are computed or available in the untrusted host machine (UHM) is available to the adversary. Note that what values must be stored in the TPM will depend on the implementation. For example in traditional *black-box* security model i.e. CK or eCK model, all the computations are done in the TPM and hence no intermediate information is available to the adversary as no computation is done on untrusted host. Whereas in our security model only the top level secret informations of a party are kept in the TPM and only those computations that directly involve the secret keys are done in the TPM. After the computations are done in the TPM, the computed values are returned back to the untrusted host. So summarizing, we consider the implementation of our protocol at a party follows one of the two approaches detailed below as in [5].

Approach 1. This approach is similar to the black box models like the CK or eCK security models. Here the LTK of a party is stored in the TPM and the session specific secret values are stored in untrusted host machine. The session key is computed in the TPM and then passed on to the host machine or application for use. This approach is in line with the CK or eCK model since in these models the adversary can obtain leakages on the session state or ephemeral secret keys of a session. Also an adversary can obtain session keys of some sessions and it gains no useful information about the other session keys. This models *known session key* attacks.

Approach 2. In this approach the top level secret keys and intermediate results are stored in the tamper proof device (TPM) and the intermediate results (*IR*) are passed on to the host machine with which it computes the session keys. This approach is motivated by real life constraints and attacks. In practice often the computation of intermediate results are more costly than the ephemeral public key computations. Hence, the implementation efficiency of this approach is very high. Also, an attacker can learn all the intermediate information stored in the unturusted host or passed to the host machine from the TPM by means of implanting malwares in the host machine. Our protocol should guarantee that even in such a scenario the attacker does not get to compromise any *fresh* or *unexposed* sessions.

We now specify the queries that are allowed to the adversary in our security model in both the implementation approaches mentioned above. These queries model all the leakages that may occur in both these scenarios.

In Set 1, the adversary is given access to the following oracle queries as mentioned. These queries capture all the information that an adversary may access in a black box model of security for key exchange like the CK or eCK model.

1. *EphemeralKeyReveal(sid):* This query gives the adversary access to the ephemeral secret keys of parties. In real world the random number generators may be leaky in the sense that their output may not be purely random which gives some idea on the ephemeral key of parties to an adversary. This query models this leakage on ephemeral secret keys of session *sid*.
2. *Corrupt(ID_i):* This query allows an adversary to gain read access to the device's private memory and hence the static secret keys of the party with identity ID_i, thus bypassing the protection of the TPM.
3. *SessionKeyReveal(sid):* This query allows an adversary to learn the session key of a completed session with identifier *sid*. This model *known session key* attacks.
4. *EstablishParty(ID_i):* This query allows an adversary to register static public key on behalf of a party with identity ID_i. A party against which this query is not issued is said to be *honest*, otherwise it is at the complete control of the adversary. Note that unlike *Corrupt* query, this query gives the adversary the ability to register arbitrary public keys in the system.
5. $Test(sid)$: The Test query is made on a completed and fresh session *sid* as defined in Definition 4.2. On a test query a bit $b \in \{0,1\}$ is randomly chosen. If $b = 0$ the true session key of that session *sid* is returned to the adversary $\mathscr{A}$, otherwise a uniformly chosen random value from the distribution of valid session keys is returned to $\mathscr{A}$. Only one query of this form is allowed. After the Test query has been issued, the adversary can adaptively query the oracles like before provided the test session remains fresh.

In Set 2 the following queries are allowed to the adversary. These queries corresponds to hybrid model of computation and considers leakages corresponding to the second approach mentioned above. The definitions of all the queries mentioned below apart from the first query remains unchanged from Set 1.

1. *IntermediateResult(sid):* This query allows the adversary to obtain all the session specific values available and computed locally in the untrusted host. Let AUX denotes the output of *IntermediateValues(sid)* oracle while computing the session keys i.e. $AUX \leftarrow$ *IntermediateValues(sid)*, where *sid* denotes the session identifier of the session run between two parties say ID_A and ID_B. It includes all the intermediate results or auxiliary information generated in computing session keys for the session *sid*. Particularly it includes all the values that are available in the untrusted host and also computed locally in the untrusted host. So the set AUX includes (i) the values computed using the secret keys stored in the TPM and passed on to the host machine and (ii) all other local computations done in the untrusted host machine. So when the adversary makes the *IntermediateValues(sid)* oracle query he gets AUX. Note that when all computations are done in TPM, AUX is empty and then this model is same as the black-box model, where the adversary has no access to the values present in the untrusted host.
2. *Corrupt(ID_i).*
3. *SessionKeyReveal(sid).*
4. *EstablishParty(ID_i).*
5. $Test(sid)$

We now give the definition for a *matching* session and what it means for a session to be *fresh*.

Definition 4.1 (Matching Sessions)**.** Let Π be a protocol and $sid = (ID_i, ID_j, out, in, \zeta)$ and $sid' = (ID_r, ID_s, out', in', \zeta')$ be the identifier of two sessions. Then *sid* and sid' are called matching sessions if:

- $s = i$ and $r = j$
- $in' = out$ and $out' = in$ and
- $\zeta' \neq \zeta$

Definition 4.2. (Session Freshness). Let Π be a protocol, and $sid = (ID_i, ID_j, out, in, \zeta)$ be the identifier of a completed session. The session *sid* is said to be *locally-exposed* if any of the following conditions holds:

- $\mathscr{A}$ issued a *SessionKeyReveal*(sid) on the target session *sid*.
- The party with identifier ID_i follows the first approach and $\mathscr{A}$ issued an *EphemeralKeyReveal*(sid) and *Corrupt(ID_i)* query.
- The party with identifier ID_i follows the second approach and $\mathscr{A}$ issued an *IntermediateResult(sid)* query.

The session *sid* is said to be *exposed* if (a) the session *sid* is locally exposed or its matching session sid' exists and is locally exposed, or (b) its matching session sid' does not exist and $\mathscr{A}$ issued *Corrupt(ID_j)* query.

A session *sid* is said to be *fresh*, if it is not *exposed*.

Definition 4.3. (ID-seCK **security**). A protocol Π is said to be ID-seCK-secure, if no polynomially bounded adversary can distinguish a fresh session key from a random value, chosen from the distribution of session keys, with probability significantly greater than 1/2. $\mathscr{A}$ outputs his guess b' in the test session. An adversary wins the game if he guesses the challenge b correctly, i.e., $b' = b$. The advantage of $\mathscr{A}$ in the ID-seCK game is defined as:

$$\mathsf{Adv}_{\Pi}^{\mathsf{ID\text{-}seCK}}(\mathscr{A}) = \Pr[b' = b] - \frac{1}{2}$$

We define the ID-seCK-security of Π as follows:

1. *If two honest parties complete matching sessions, then, except with negligible probability, they both compute the same session key.* (Correctness)
2. *For any probabilistic polynomial-time adversary* $\mathscr{A}$, $\mathsf{Adv}_{\Pi}^{\mathsf{ID\text{-}seCK}}(\mathscr{A})$ *is negligible.*

5 Our Construction

We now present the details of our ID-AKE protocol.

Setup: The PKG picks $s_1, s_2 \in_R \mathbb{Z}_p^*$ and sets $y_1 = g^{s_1}$ and $y_2 = g^{s_2}$. The master secret key is $\langle s_1, s_2 \rangle$ and the master public key is $\langle y_1, y_2 \rangle$. It also defines the following hash functions: $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G} \to \mathbb{Z}_p^*$ and $H_3 : \{0,1\}^* \to \mathbb{Z}_p^*$. The PKG sets $params = \langle \mathbb{G}^*, g, q, p, y_1, y_2, H_1, H_2, H_3 \rangle$ and $msk = \langle s_1, s_2 \rangle$.

Key Extract: To compute the private key of a user with identity ID_i, it proceeds as follows:

- Choose $x_i \in_R \mathbb{Z}_p^*$, compute $u_{i_1} = g^{x_i}$, set $h_i = H_1(ID_i)$ and compute $v_{i_1} = {h_i}^{x_i}$.
- Choose $r_i \in_R \mathbb{Z}_p^*$ and compute $u_{i_2} = g^{r_i}$ and $v_{i_2} = {h_i}^{r_i}$.
- Compute $c_i = H_1(ID_i, u_{i_1})$, $b_i = H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$ and $e_i = H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$.
- Compute $d_{i_1} = x_i + s_1 \cdot c_i$ and $d_{i_2} = x_i + r_i \cdot b_i + s_2 \cdot e_i$.

Finally it sends $\langle u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2}, d_{i_1}, d_{i_2}, {h_i}^{s_2} \rangle$ to the user with identity ID_i.

After receiving the private key from the PKG the user performs a Key Sanity Check as described below. This check ensures that the private key components are computed correctly by the PKG.

Key Sanity Check: The user computes the values $c_i = H_1(ID_i, u_{i1})$, $b_i = H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$ and $e_i = H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$, and checks the following:

1. $\dfrac{g^{d_{i_1}}}{y_1^{H_1(ID_i, u_{i_1})}} \stackrel{?}{=} u_{i_1}$
2. $\dfrac{g^{d_{i2}}}{u_{i_2}^{H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})} y_2^{H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})}} \stackrel{?}{=} u_{i_1}$
3. $\dfrac{h_i^{d_{i_2}}}{v_{i_2}^{H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})} ({h_i}^{s_2})^{H_1(ID_i, u_{i1}, v_{i_1}, u_{i_2}, v_{i_2})}} \stackrel{?}{=} v_{i_1}$

Test 1 ensures the correctness of d_{i_1} and u_{i_1}. Test 2 ensures the correctness of $d_{i_2}, u_{i_2}, v_{i_1}, v_{i_2}$ and Test 3 ensures the correctness of ${h_i}^{s_2}$. Test 2 and Test 3 ensures the integrity of the exponent x_i in u_{i_1} and v_{i_1} respectively. All these tests can be verified in the honest case trivially.

Key Agreement: Let us assume two users with identifiers ID_i and ID_j with their private keys S_i and S_j respectively engage in the ID-AKE protocol. User ID_i chooses $w_i \in_R \mathbb{Z}_p^*$ as ephemeral secret and computes $W_i = g^{w_i}$. Similarly user ID_j chooses $w_j \in_R \mathbb{Z}_p^*$ as ephemeral secret and computes $W_j = g^{w_j}$. They now proceed with the protocol as described in Table 3. Finally, at the end both the parties establish a shared secret key denoted by Z.

Remark 1. The values of V_i and V_j are session specific. So each user can precompute it earlier and use those values later when they are required by using a look up table. The signature $w_i + d_{i_1} \cdot H_2(g^{w_i})$ on $W_i (= g^{w_i})$ is computed securely so that w_i and d_{i_1} are not leaked to the adversary although at the end the signature is made available to the adversary.

Remark 2. Note that the components F_i, V_i and F_j, V_j are required to be sent only once for the first key establishment between users ID_i and ID_j since these components remains invariant across all sessions. For all subsequent sessions between these two users, it is enough to transfer the components that are freshly generated per session namely the V_i and V_j values.

We note the following.

- **Check 1** is done to ensure correctness of the component F_i and F_j. This in turn checks whether both g and h_i are raised to the same exponent x_i.
- **Check 2** is done to ensure correctness of the component V_i and V_j. This check essentially verifies the signature $w_i + d_{i_1} \cdot H_2(g^{w_i})$ to ensure that an adversary has not tampered with the message in transit.

Party ID_i	**Party** ID_j
Step 1: Send $F_i = \langle u_{i_1}, v_{i_1}, d_{i_2}, b_i, e_i, {h_i}^{s_2},\ ID_i\rangle$, $V_i = \langle w_i + d_{i_1} \cdot H_2(g^{w_i}),\ g^{w_i}\rangle, X_i = g^{d_{i_1}}$ to ID_j.	**Step 1:** Send $F_j = \langle u_{j_1}, v_{j_1}, d_{j_2}, b_j, e_j, {h_j}^{s_2}, ID_j\rangle$, $V_j = \langle w_j + d_{j_1} \cdot H_2(g^{w_j}),\ g^{w_j}\rangle, X_j = g^{d_{j_1}}$ to ID_i.
Step 2: (Correctness Checks) (a) **Check 1:** Compute $u_{j_2} = \left(\frac{g^{d_{j_2}}}{u_{j_1} \cdot {y_2}^{e_j}}\right)^{{b_j}^{-1}}$ Compute $v_{j_2} = \left(\frac{{h_j}^{d_{j_2}}}{v_{j_1} \cdot ({h_j}^{s_2})^{e_j}}\right)^{{b_j}^{-1}}$ Check if $b_j \stackrel{?}{=} H_1(ID_j, u_{j_1}, v_{j_1}, u_{j_2}, v_{j_2})$ $e_j \stackrel{?}{=} H_1(ID_j, u_{j_1}, v_{j_1}, u_{j2}, v_{j_2})$ Proceed if (Check 1 == True). (b) **Check 2:** Compute $c_j = H_1(ID_j, u_{j_1})$. Check if $\left[\frac{g^{\left(w_j + d_{j_1} \cdot H_2\left(g^{w_j}\right)\right)}}{\left(g^{x_j}\right)^{H_2\left(g^{w_j}\right)} \left(y_1\right)^{c_j \cdot H_2\left(g^{w_j}\right)}}\right] \stackrel{?}{=} g^{w_j}$ Proceed if (Check 2 == True). (c) **Check 3:** $u_{j_1} \cdot (y_1)^{c_j} \stackrel{?}{=} X_j$. Proceed if (Check 3 == True).	**Step 2:** (Correctness Checks) (a) **Check 1:** Compute $u_{i_2} = \left(\frac{g^{d_{i_2}}}{u_{i_1} . {y_2}^{e_i}}\right)^{{b_i}^{-1}}$ Compute $v_{i_2} = \left(\frac{{h_i}^{d_{i_2}}}{v_{i_1} . ({h_i}^{s_2})^{e_i}}\right)^{{b_i}^{-1}}$ Check if $b_i \stackrel{?}{=} H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$ $e_i \stackrel{?}{=} H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$ Proceed if (Check 1 == True). (b) **Check 2:** Compute $c_i = H_1(ID_i, u_{i_1})$. Check if $\left[\frac{g^{\left(w_i + d_{i_1} \cdot H_2(g^{w_i})\right)}}{\left(g^{x_i}\right)^{H_2(g^{w_i})} \left(y_1\right)^{c_i \cdot H_2(g^{w_i})}}\right] \stackrel{?}{=} g^{w_i}$ Proceed if (Check 2 == True). (c) **Check 3:** $u_{i_1} \cdot (y_1)^{c_i} \stackrel{?}{=} X_i$. Proceed if (Check 3 == True).
Step 3: (Shared secret key generation) Compute: $d = H_1(W_i, W_j, ID_i, ID_j)$ $e = H_1(W_j, W_i, ID_i, ID_j)$ $s_i = w_i \cdot d + d_{i_1} \mod p$ $\sigma_i = \left(W_j^e \cdot X_j\right)^{s_i}$ $Z = H_3(\sigma_i, ID_i, ID_j, W_i, W_j)$.	**Step 3:** (Shared secret key generation) Compute: $d = H_1(W_i, W_j, ID_i, ID_j)$ $e = H_1(W_j, W_i, ID_i, ID_j)$ $s_j = w_j \cdot e + d_{j_1} \mod p$ $\sigma_j = \left(W_i^d \cdot X_i\right)^{s_j}$ $Z = H_3(\sigma_j, ID_i, ID_j, W_i, W_j)$.

Table 3 : Proposed ID-AKE protocol

- **Check 3** is done to ensure correctness of the component X_i and X_j. Note that the computation of X_i and X_j involves the LTK of user ID_i and ID_j respectively. An adversary can modify the components X_i and X_j while simply relaying the rest of the message as it is namely the values F_i, V_i and F_j, V_j. Check 3 prevents this kind of illegal modification thus guaranteeing the integrity of X_i and X_j.

It is trivial to verify that for valid components, these checks hold good. The lemma below shows the correctness of our protocol.

Lemma 5.1. *The shared secret key computed by both the parties are identical.*

Proof. User i computes

$$\sigma_i = \left(W_j^e \cdot Y\right)^{s_i} = g^{\left(w_j \cdot e + d_{j_1}\right) \cdot \left(w_i \cdot d + d_{i_1}\right)}$$

User j computes

$$\sigma_j = \left(W_i^d \cdot X\right)^{s_j} = g^{\left(w_i \cdot d + d_{i_1}\right) \cdot \left(w_j \cdot e + d_{j_1}\right)}$$

So we have $\sigma_i = \sigma_j$, i.e., the values computed by both the parties are identical. This proves the lemma. □

6 Security Proof

Theorem 6.1. *Under the GDH assumption in $\mathbb{G}$ and the RO model, our ID-AKE protocol is secure in our* ID-seCK *model.*

Proof. The adversary $\mathscr{A}$ can make session activation queries of the form (ID_i, ID_j) which makes user ID_i perform **Step 1** of our protocol, and create a session with identifier $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i\rangle, \star, \mathscr{I})$ in accordance with the protocol. On a session activation query of the form $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i\rangle)$, user D_j performs **Step 2** of our protocol and creates a session with identifier $(ID_j, ID_i, \langle F_j, V_j, W_j, X_j\rangle, \langle F_i, V_i, W_i, X_i\rangle, \mathscr{R})$. The query $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{R})$ makes user ID_i update the session identifier $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i\rangle, \star, \mathscr{I})$ (if any) to $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{I})$ and perform **Step 3** of our protocol. The allowed queries in Set 1 are the following: *EphemeralKeyReveal, Corrupt, SessionKeyReveal, and EstablishParty*. In Set 2, in addition to the queries of Set 1, the adversary is allowed the following queries: *SecretExponentReveal*, to obtain the secret exponent s_i or s_j, and *SessionSignatureReveal* to obtain the session signature σ_i or σ_j.

Suppose there is an adversary $\mathscr{A}$ who can win the game i.e., it can distinguish between a fresh session key and a random session key for the test query with probability significantly bounded away than $\frac{1}{2}$. The only possible ways by which $\mathscr{A}$ can accomplish are by the following attacks:

- **Guess:** $\mathscr{A}$ manages to guess the session key corresponding to the test session correctly.
- **Key Replication:** In this attack $\mathscr{A}$ succeeds in making one more non-matching session different from the test session to compute the same session key as the test session. Then $\mathscr{A}$ can simply issue a session key reveal query on the non-matching session and since both the session keys are identical it can use that session key as a valid forgery for the test session.
- **Forgery:** In this attack $\mathscr{A}$ issues the H_3 digest query on a tuple that evaluates to the same session key as the test session. This represents a successful forgery of the session signature σ by the adversary.

Since we are working in the RO model the first two attacks cannot succeed except with negligible probability. This is because the session specific values are chosen independently and randomly of each session and due to the collision resistance property of the hash function H_3. Thus it suffices the third attack namely the forging probability of the attacker. Let us define an event E as "$\mathscr{A}$ succeeds in forging the signature of a fresh session denoted by $sid_0 = \left(ID_i, ID_j, \langle F_{i_0}, V_{i_0}, W_{i_0}, X_{i_0}\rangle, \langle F_{j_0}, V_{j_0}, W_{j_0}, X_{j_0}\rangle, \varsigma\right)$". The event E is again divided in two sub-cases – E.1: "$\mathscr{A}$ succeeds in forging the signature of a fresh and matching session" and E.2: "$\mathscr{A}$ succeeds in forging the signature of a fresh without matching session". So if suffices to show that neither E.1 nor E.2 can happen with non-negligible probability. E.1 is further analyzed case by case as follows:

- E.1.1: E.1 and both i and j follow the first implementation approach.
- E.1.2: E.1 and both i and j follow the second implementation approach.
- E.1.3: E.1 and i and j follow different implementation approaches.

Suppose that E.1 occurs with non-negligible probability, then at least one of these events happens with non-negligible probability. E.1.1 can be further analyzed by cases as follows:

- E.1.1.1: E.1.1 $\wedge$ $\mathscr{A}$ issues $Corrupt(ID_i)$ and $Corrupt(ID_j)$.
- E.1.1.2: E.1.1: $\wedge$ $\mathscr{A}$ issues an $EphemeralKeyReveal(sid_0)$ and an $EphemeralKeyReveal(sid_0')$ query.

- E.1.1.3: E.1.1: $\wedge$ $\mathscr{A}$ issues $Corrupt(ID_i)$ and an $EphemeralKeyReveal(sid'_0)$ query.
- E.1.1.4: E.1.1: $\wedge$ $\mathscr{A}$ issues $Corrupt(ID_j)$ and an $EphemeralKeyReveal(sid_0)$ query.

Case E.2 can also be further subdivided into cases E.2.1, E.2.2 and E.2.3 in a similar fashion.

Analysis of E.1.1.1. If event E.1.1.1 occurs with non-negligible probability, we can build a polynomial time CDH solver $\mathscr{S}$ using $\mathscr{A}$ with advantage related to the probability with which the event occurs. In particular we will get a CDH solver that succeeds with non-negligible probability. The solver interacts with $\mathscr{A}$ as follows:

Suppose there are n parties $\hat{P}_1, \cdots, \hat{P}_n$ each following a particular implementation approach. $\mathscr{S}$ simulates $\mathscr{A}$'s environment. We only suppose that the number of parties following the first implementation approach is $n_1 \geq 2$. Since $\mathscr{A}$ is polynomial (in $|q|$), we suppose that each party is activated at most m times $(m, n \leq L(|q|))$ for some polynomial L) $\mathscr{S}$ chooses $i, j \in_R \{k \mid \hat{P}_k$ follows the first implementation approach$\}$, and $t \in_R [m+1]$ (with these choices, $\mathscr{S}$ is guessing the test session). The challenger receives as input the GDH problem instance $\langle \mathbb{G}, g, q, p, C = g^a, D = g^b \rangle$ and also access to the Diffie Hellman Oracle $DDH(y_1, \cdot, \cdot)$. We refer to $\hat{P}_i$ as ID_i and $\hat{P}_j$ as ID_j. The challenger simulates the hash oracles in the following way:

H_1 *Oracle* : If the adversary queries the H_1 oracle with ID_i or (ID_i, u_{i_1}) or $(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2})$ as input, the challenger checks whether the hash value of the requested tuple already exists in the hash list L_{h1}. Otherwise chooses $k_i \in_R \mathbb{Z}_p^*$, computes $h_i = g^{k_i}$, adds the tuple $\langle h_i, ID_i, k_i \rangle$ to the L_{h1} list.

Similarly, the H_2 and H_3 oracles are also answered in the same fashion. The challenger looks up its corresponding list L_{h2} to L_{h3} to see if the hash value corresponding to the query is already in the list; else it will choose a value randomly from the domain of the function and answer with a hash value that has an identical distribution to the actual value in the protocol.

Simulation of Corrupt query: When the adversary queries for a private key corresponding to party with identity ID_i, the challenger computes the private key corresponding to that user as shown below. Note that the challenger does not have the master secret key s_1 to generate the private keys of users. So it has to simulate the key extraction as follows:

The challenger first looks up the hash list L_{h1} to check if ID_i was queried before. If it is available in the list, it extracts k_i, h_i from the list and proceeds to the next step. If this is the first query, the challenger chooses $k_i \in_R \mathbb{Z}_p^*$, computes $h_i = g^{k_i}$, adds the tuple $\langle h_i, ID_i, k_i \rangle$ to the L_{h1} list. Then it does the following:

- Choose $c_i, b_i, e_i, x_i', r_i' \in_R {\mathbb{Z}_p}^*$ and sets $u_{i_1} = g^{x_i'}.{y_1}^{-c_i}$.
- Set $H_1(ID_i, u_{i1}) = c_i$ and add $\langle c_i, u_{i1}, ID_i \rangle$ to L_{h1}.
- Set $d_{i_1} = x_i'$, $d_{i_2} = x_i' + r_i' \cdot b_i + s_2 \cdot e_i$ and $u_{i_2} = g^{r_i'} \cdot {y_1}^{c_i \cdot {b_i}^{-1}}$, and compute $v_{i_1} = g^{k_i \cdot x_i'} \cdot {y_1}^{-k_i \cdot c_i}$ and $v_{i_2} = g^{k_i \cdot r_i'} \cdot {y_1}^{k_i \cdot c_i \cdot {b_i}^{-1}}$.
- Set $H_1(ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2}) = b_i$, $H_1(ID_i, u_{i_1}, v_{i_1}, u_{i2}, v_{i2}) = e_i$ and add $\langle b_i, ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2} \rangle$, $\langle e_i, ID_i, u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2} \rangle$ to L_{h1}, and compute ${h_i}^{s_2}$.

It then makes an entry in the list $L_E = \langle u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2}, d_{i_1}, d_{i_2}, ID_i \rangle$ and returns $\langle u_{i_1}, v_{i_1}, u_{i_2}, v_{i_2}, d_{i_1}, d_{i_2}, {h_i}^{s_2} \rangle$ as private key of ID_i.

Note that the private key computed in this way is consistent with the actual private key in the system. It is trivial to check that the private key returned by $\mathscr{S}$ passes the **Key Sanity Check**.

The interaction between the challenger or solver $\mathscr{S}$ and the adversary $\mathscr{A}$ proceeds as follows:

– $\mathscr{S}$ takes as input $C = W_{i_0}$ and $D = W_{j_0} \in \mathbb{G}$. Note that the values of w_{i_0} and w_{j_0} are implicitly set to a and b respectively and they are unknown to the challenger.

– On a session activation query of the form $(\hat{P}_l, \hat{P}_m)$, $\mathscr{S}$ chooses a w_i and create session identifier $sid' = \left(\hat{P}_l, \hat{P}_m, \langle F_i, V_i, W_i, X_i \rangle, \star, \mathscr{I}\right)$ and provide $\mathscr{A}$ with $\left(\hat{P}_l, \hat{P}_m, \langle F_i, V_i, W_i, X_i \rangle\right)$.

– On a session activation query of the form $\left(\hat{P}_m, \hat{P}_l, \langle F_j, V_j, W_j, X_j \rangle\right)$, $\mathscr{S}$ chooses a w_i and create session identifier $sid' = \left(\hat{P}_l, \hat{P}_m, \langle F_i, V_i, W_i, X_i \rangle, \langle F_j, V_j, W_j, X_j \rangle, \mathscr{R}\right)$ and provide $\mathscr{A}$ with $\left(\hat{P}_l, \hat{P}_m, \langle F_i, V_i, W_i, X_i \rangle\right)$ and completes the session $\left(\hat{P}_l, \hat{P}_m, \langle F_i, V_i, W_i, X_i \rangle, \langle F_j, V_j, W_j, X_j \rangle \mathscr{R}\right)$.

– On the session activation query $\left(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i.X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle\right)$, $\mathscr{S}$ updates the session identifier $\left(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \star, \mathscr{I}\right)$ (if any) to $sid = \left(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{I}\right)$. If the sid' session exists and is already completed, $\mathscr{S}$ sets the sid session key to that of sid'. Else, if a digest query was previously issued on some $\psi = \left(\sigma, \hat{P_l}, \hat{P_m}, W_i, W_j\right)$ (in this case d and e are already defined) and if σ is the sid session signature ($\mathscr{S}$ can compute the session signature), $\mathscr{S}$ sets the session key to $H_3(\psi)$. Else $\mathscr{S}$ chooses $Z \in_R \{0,1\}^\kappa$ and set the sid session key to Z and updates L_{h3}.

– If $\mathscr{A}$ issues a *Corrupt*, an *EphemeralKeyReveal*, a *SessionKeyReveal*, or an *EstablishParty* query at a party following the first implementation approach, $\mathscr{S}$ answers faithfully.

• If $\mathscr{A}$ issues a *Corrupt*, a *SessionKeyReveal*, a *SecretExponentReveal*, a *SessionSignatureReveal*, or an *Establish-Party* query at a party following the second implementation approach, $\mathscr{S}$ answers faithfully.

– At the activation of the t^{th} session at ID_i, $\mathscr{S}$ aborts if the peer of ID_i is not ID_j; otherwise, it gives $(ID_i, ID_j, F_i, V_i, W_{i_0}, X_i)$ as input to $\mathscr{A}$ (recall that the solver takes as input W_{i_0} and W_{j_0}).

– When $\mathscr{A}$ activates the session matching the t^{th} session at ID_i, $\mathscr{S}$ provides $\mathscr{A}$ with $(ID_j, ID_i, F_j, V_j, W_{j_0}, X_j)$.

• $\mathscr{S}$ aborts if any of the following happens:

  – $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  – $\mathscr{A}$ issues a *SessionKeyReveal(sid)* or $SessionKeyReveal(sid')$ query, where sid' is the matching session of sid.
  – $\mathscr{A}$ issues a *EphemeralKeyReveal(sid)* or $EphemeralKeyReveal(sid')$.
  – $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.

• If $\mathscr{A}$ halts with a guess σ_0, $\mathscr{S}$ outputs the solution to the CDH problem as follows:

$\left(\sigma_0\left(W_{i0}^{d_0} \cdot X_i\right)^{-d_{j_1}} W_{j0}^{-d_{i_1} \cdot e_0}\right)^{(d_0 \cdot e_0)^{-1}}$ as $CDH(C,D)$. Otherwise $\mathscr{S}$ aborts.

The simulation remains perfect, except with negligible probability. The probability that the solver $\mathscr{S}$ guesses the test session correctly is $\left(n^2 m\right)^{-1}$. If $\mathscr{A}$ succeeds under this simulation, and $\mathscr{S}$ guesses correctly the test session, $\mathscr{S}$ outputs $CDH(W_{i_0}, W_{j_0})$. Hence if $\mathscr{A}$ succeeds with non–negligible probability in E.1.1.1, $\mathscr{S}$ outputs with non–negligible probability $CDH(W_{i_0}, W_{j_0})$, contradicting the GDH assumption.

Analysis of E.1.1.2. If event E.1.1.2 occurs with non-negligible probability, we can build a polynomial time CDH solver $\mathscr{S}$ using $\mathscr{A}$ with advantage related to the probability with which the event occurs. In particular we will get a CDH solver that succeeds with non-negligible probability. We modify the interaction of event E.1.1.1 as follows:

• $\mathscr{S}$ takes as input $C = X_i$ and $D = X_j \in \mathbb{G}$. So it doesn't know the values of d_{i_1} and d_{j_1}.

• On a session activation query of the form $(\hat{P_m}, \hat{P_l}, \langle F_j, V_j, W_j, X_j\rangle)$, with $\hat{P_l} = ID_i$ or ID_j, $\mathscr{S}$ chooses a $w_i \in_R \mathbb{Z}_p^*$ and create session identifier $sid' = (\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{R})$ and provide $\mathscr{A}$ with $(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle)$

• On the session activation query $(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle)$, with $\hat{P_l} = ID_i$ or ID_j, $\mathscr{S}$ updates the session identifier $(\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \star, \mathscr{I})$ (if any) to $sid = (\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{I})$. If the sid' session exists and is already completed, $\mathscr{S}$ sets the sid session key to that of sid'. Else, if a digest query was previously issued on some $\psi = (\sigma, \hat{P_l}, \hat{P_m}, W_i, W_j)$ (in this case d and e are already defined) and if $\sigma = CDH(W_i^d \cdot X_i, W_j^e \cdot X_j)$ (using the *DDH* oracle), set the sid session key to $H_3(\psi)$. Else, a random element from the session key space is chosen and assigned to Z ,i.e., $Z \in_R \{0,1\}^\kappa$ where Z is the session key of the session sid. If no value was previously assigned to $h_1 = H_1(W_i, W_j, \hat{P_l}, \hat{P_m})$ (respectively $h_1' = H_1(W_j, W_i =, \hat{P_l}, \hat{P_m})$, choose $d \in_R \mathbb{Z}_p^*$ and set $h_1 = d$ (respectively $h_1' = d$).

• When $\mathscr{A}$ makes a hash query on some $\psi = (\sigma, \hat{P_l}, \hat{P_m}, W_i, W_j)$ with with $\hat{P_l} = ID_i$ or ID_j or with $\hat{P_m} = ID_i$ or ID_j, $\mathscr{S}$ checks whether a value Z already exists for the queried ψ, then it returns Z as the completed session key, else *(i)* if there is an already completed session with identifier $sid = (\hat{P_l}, \hat{P_m}, \langle F_i, V_i, W_i, X_i\rangle, \langle F_j, V_j, W_j, X_j\rangle, \mathscr{I})$ or sid', and $\sigma = CDH(W_i^d \cdot X_i, W_j^e \cdot X_j)$, then $\mathscr{S}$ returns the completed session key, else *(ii)* Choose $Z \in_R \{0,1\}^\kappa$ and set Z as the session key of sid and provides $\mathscr{A}$ with Z; if no value was previously assigned to $h_1 = H_1(W_i, W_j, \hat{P_l}, \hat{P_m})$ (respectively $h_1' = H_1(W_j, W_i, \hat{P_l}, \hat{P_m})$, choose $d \in_R \mathbb{Z}_p^*$ and set $h_1 = d$ (respectively $h_1' = d$).

- At the activation of the t^{th} session at ID_i, $\mathscr{S}$ aborts if the peer of ID_i is not ID_j; otherwise, it chooses $w_i \in_R \mathbb{Z}_p^*$ and provides $\mathscr{A}$ with $(ID_i, ID_j, F_i, V_i, W_{i_0}, X_i)$.
- When $\mathscr{A}$ activates the session matching the t^{th} session at ID_i, $\mathscr{S}$ chooses $w_B \in_R \mathbb{Z}_p^*$ provides $\mathscr{A}$ with $(ID_j, ID_i, F_j, V_j, W_{j_0}, X_j)$.
- When $\mathscr{A}$ issues an *EphemeralKeyReveal* query on the t^{th} session at ID_i or its matching session $\mathscr{S}$ answers faithfully.
- $\mathscr{S}$ aborts if any of the following happens:
  - $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  - $\mathscr{A}$ issues a *SessionKeyReveal(sid)* or *SessionKeyReveal*(sid') query, where sid' is the matching session of sid.
  - $\mathscr{A}$ issues a $Corrupt(ID_i)$ or $Corrupt(ID_j)$ query.
  - $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.
- When $\mathscr{A}$ comes up with a guess σ_0 for the test session, $\mathscr{S}$ outputs the solution to the CDH problem $CDH(C,D)$ from $\sigma_0, w_{i_0}, w_{j_0}, d_0$ and e_0.

The simulation remains perfect, except with negligible probability. The probability that $\mathscr{A}$ guesses the test session correctly is $\left(n^2 m\right)^{-1}$. The probability that $\mathscr{A}$ succeeds in the test session is $\left(n^2 m\right)^{-1} \cdot \Pr(\text{E.1.1.2})$. If the event E.1.1.2 happens with non-negligible probability, then the probability that $\mathscr{A}$ succeeds in the test session is also non-negligible and hence the probability that the solver $\mathscr{S}$ outputs $CDH(C,D)$ is also non-negligible. However this cannot happen since it directly contradicts the GDH assumption.

Analysis of E.1.1.3 and E.1.1.4: Events E.1.1.3 and E.1.1.4 are symmetrical to each other. So it suffices to analyze only event E.1.1.3. If event E.1.1.2 occurs with non-negligible probability, we can build a polynomial time CDH solver $\mathscr{S}$ using $\mathscr{A}$ with advantage related to the probability with which the event occurs. In particular we will get a CDH solver that succeeds with non-negligible probability. The solver interacts with $\mathscr{A}$ as follows:

- $\mathscr{S}$ takes as input $C = W_{i0}$ and $D = X_j \in \mathbb{G}$. Note that the value of $d_{i_{1_0}}$ is implicitly set to ${x_{i_0}}'$. So $\mathscr{S}$ sets $X_{i_0} = g^{{x_{i_0}}'}$.
- On a session activation query of the form $(ID_i, ID_j, \langle F_i, V_i, W_i, X_i \rangle)$, $\mathscr{S}$ does the following:
  - Choose $d_{j_{1_0}} \in_R \mathbb{Z}_p^*$ and compute $X_j = g^{d_{j_{1_0}}}$.
  - Create a session with identifier $sid' = (ID_j, ID_i, \langle F_j, V_j, W_j, X_j \rangle, \langle F_i, V_i, W_i, X_i \rangle, \mathscr{R})$ and provide $\mathscr{A}$ with $(ID_j, ID_i, \langle F_j, V_j, X_j \rangle)$.
  - Choose $Z \in_R \{0,1\}^\kappa, d, e \in_R \mathbb{Z}_p^*$ and set the session key to Z.
- On the query $(ID_j, ID_i, \langle F_j, V_j, W_j, X_j \rangle, \langle F_i, V_i, W_i, X_i \rangle)$, $\mathscr{S}$ does the following:
  - Update the session identifier $(ID_j, ID_i, \langle F_j, V_j, W_j, X_j \rangle, \star, \mathscr{I})$ (if any) to $(ID_j, ID_i, \langle F_j, V_j, W_j, X_j \rangle, \langle F_i, V_i, W_i, X_i \rangle, \mathscr{I})$.
  - If the session key of the session sid' is already assigned a value, set the sid session key to that of sid'. Else, if a digest query was previously issued on some $\psi = (\sigma, ID_j, ID_i, W_j, W_i)$ (in this case d and e are already defined) and if $\sigma = CDH(W_i^d X_i, W_j^e X_j)$ (using the DDH oracle), set the sid session key to $H_3(\psi)$. Else, choose $Z \in_R \{0,1\}^\kappa$ and set the sid session key to Z; if no value was previously assigned to $h_1 = H_1(W_j, W_i, ID_i, ID_j)$ (respectively $h_1' = H_1(W_i, W_j, ID_i, ID_j)$), choose $d \in_R \mathbb{Z}_p^*$ and set $h_1 = d$ (respectively $h_1' = d$).
- When $\mathscr{A}$ makes a hash query on some $\psi = (\sigma, P_i, P_j, W_i, W_j)$ with $P_i = ID_j$ or $P_j = ID_i$, $\mathscr{S}$ checks whether this has been queried before, then it returns the same value. Else it follows the steps as mentioned ahead.
- At the activation of the t^{th} session at ID_i, $\mathscr{S}$ aborts if the peer of ID_i is not ID_j; otherwise, it provides $\mathscr{A}$ with $(ID_i, ID_j, F_i, V_i, W_{i0}, X_j)$ (recall that the solver takes as input W_{i0} and X_j).

- When $\mathscr{A}$ activates the session matching the t-th session at ID_i, $\mathscr{S}$ chooses $d_{j_{1_0}} \in_R \mathbb{Z}_p^*$, and provides $\mathscr{A}$ with $\left(ID_j, ID_i, \langle F_j, V_j, W_{j_0}, X_j \rangle\right)$.
- If $\mathscr{A}$ issues an $EphemeralKeyReveal$ query on the session matching the t-th session at ID_i, $\mathscr{S}$ answers faithfully.
- $\mathscr{S}$ aborts if any of the following happens:
  - $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  - $\mathscr{A}$ issues a *SessionKeyReveal(sid)* or $SessionKeyReveal(sid')$ query, where sid' is the matching session of sid.
  - $\mathscr{A}$ issues a $Corrupt(ID_j)$ query.
  - $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.
  - $\mathscr{A}$ issues an $EphemeralKeyReveal(sid)$
- If $\mathscr{A}$ halts with a guess σ_0, $\mathscr{S}$ produces $\left(\sigma_0\left(W_{i0}^{d_0} \cdot X_i\right)^{-d_{j_{1_0}} \cdot e_0} X_j^{-d_{i_1}}\right)^{e_0^{-1}}$ as $CDH(C,D)$.

The simulation remains perfect, except with negligible probability. The probability that $\mathscr{A}$ guesses the test session correctly is $\left(n^2 m\right)^{-1}$. The probability that $\mathscr{A}$ succeeds in the test session is $\left(n^2 m\right)^{-1} \cdot$ Pr(E.1.1.2). If the event E.1.1.2 happens with non-negligible probability, then the probability that $\mathscr{A}$ succeeds in the test session is also non-negligible and hence the probability that the solver $\mathscr{S}$ outputs $CDH(W_{i0}, X_j)$ is also non-negligible. However this cannot happen since it directly contradicts the GDH assumption.

So based on the analysis of the events E.1.1.1, E.1.1.2, E.1.1.3,E.1.1.4 we conclude that event E.1.1 cannot occur (except with negligible probability) since all the sub events also cannot happen except with negligible probability.

Analysis of E.1.2. Here both ID_i and ID_j follows the second implementation approach. If event E.1.2 occurs with non-negligible probability, we can build a polynomial time CDH solver $\mathscr{S}$ using $\mathscr{A}$ that succeeds with non-negligible probability. The strongest queries that can be issued on ID_i and ID_j, the test session and its matching session are *Corrupt* queries on both ID_i and ID_j. The simulation is almost similar to case E.1.1.1, with the following modifications:

- $\mathscr{S}$ takes as input $C = W_{i_0}$ and $D = W_{j_0} \in \mathbb{G}$ as before.
- $\mathscr{A}$'s environment is simulated in exactly the same way but $\mathscr{S}$ chooses $i, j \in_R \{k \mid \hat{P}_k$ follows the second implementation approach$\}$ (we assume $n - n_1 \geq 2$ and refer to $\hat{P}_i$ as ID_i and $\hat{P}_j$ as ID_j.
- $\mathscr{S}$ aborts if any of the following happens:
  - $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  - $\mathscr{A}$ issues a *SessionKeyReveal(sid)* or $SessionKeyReveal(sid')$ query, where sid' is the matching session of sid.
  - $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.
  - $\mathscr{A}$ issues a $SessionSignatureReveal(sid)$ or $SessionSignatureReveal(sid')$ query.
  - $\mathscr{A}$ issues a $SecretExponentReveal(sid)$ or $SecretExponentReveal(sid')$ query.

The simulation remains perfect, except with negligible probability. If $\mathscr{A}$ succeeds in the test session and $\mathscr{S}$ guesses correctly the test session which happens with probability $\left((n-n_1)^2 m\right)^{-1} \cdot$ Pr(E.1.2), $\mathscr{S}$ outputs $CDH(C,D)$.

Analysis of E.1.3. Here ID_i and ID_j follows different implementation approaches. Without loss of generality we assume, ID_i follows the first implementation approach. The strongest queries that can be issued by $\mathscr{A}$ are *(i)* $Corrupt(ID_i)$ and $Corrupt(ID_j)$, and *(ii)* An $EphemeralKeyReveal(sid)$ query where sid is the test session and a $Corrupt(ID_j)$ query. So it suffices to consider the following events:

- E.1.3.1: E.1.3 $\wedge$ $\mathscr{A}$ issues $Corrupt(ID_i)$ and $Corrupt(ID_j)$ query.
- E.1.3.2: E.1.3 $\wedge$ $\mathscr{A}$ issues $EphemeralKeyReveal(sid)$ where sid is the test session and a $Corrupt(ID_j)$ query.

For the simulation of event E.1.3, we modify the analysis of E.1.1.1 as follows:

- $\mathscr{A}$'s environment is simulated in exactly the same as E.1.1.1, but $\mathscr{S}$ chooses $i, j \in_R \{k \mid \hat{P}_k$ follows the second implementation approach$\}$
- $\mathscr{S}$ aborts if any of the following happens:
  - $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  - $\mathscr{A}$ issues a *SessionKeyReveal(sid)* or *SessionKeyReveal*(sid') query, where sid' is the matching session of sid.
  - $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.
  - $\mathscr{A}$ issues a $SessionSignatureReveal(sid')$ or $SecretExponentReveal(sid')$ query.

Under the same arguments as in analysis of E.1.1.1, $\mathscr{S}$ is a polynomial time CDH solver with probability $\left((n-n_1)^2 m\right)^{-1} \cdot$ Pr(E.1.3.1). However this contradicts the GDH assumption. Similar arguments hold for the analysis of E.1.3.2.

Analysis of E.2. Event E.2 is the event that $\mathscr{A}$ succeeds in forging the signature of a session that is fresh and does not have a matching session. This can be further analyzed case by case as follows:

- E.2.1: E.2 $\wedge$ and both ID_i and ID_j follows the first implementation approach.
- E.2.2: E.2 $\wedge$ and both ID_i and ID_j follows the second implementation approach.
- E.2.3: E.2 $\wedge$ and both ID_i and ID_j follows different implementation approach.

If event E.2 occurs with non-negligible probability, then at least one of these events occurs with non-negligible probability.

Analysis of E.2.1. Event E.2.1 can be further analyzed as follows:

- E.2.1.1: E.2.1 $\wedge$ $\mathscr{A}$ issues a $Corrupt(ID_i)$ and
- E.2.1.2: E.2.1 $\wedge$ $\mathscr{A}$ issues a $EphemeralKeyReveal(sid)$ where sid is the test session.

These are the strongest query that $\mathscr{A}$ can issue in event E.2.1.

Event E.2.1.1: Here we modify the simulation in the analysis of E.1.1.3 to take as input W_{i0} and $X_{j_0} \in_R \mathbb{G}$ (for ID_i, $\mathscr{S}$ chooses $d_{i_1} \in_R \mathbb{Z}_p^*$ and sets $X_{i_0} = g^{d_{i_1}}$). The simulation environment remains perfect except with negligible probability. If the challenger or solver $\mathscr{S}$ guesses the test session correctly and $\mathscr{A}$ succeeds with a proper forgery σ_0, then $\mathscr{S}$ outputs σ_0 as the FDCR-1 forgery on messages ID_i and ID_j with respect to their public keys. $\mathscr{S}$ succeeds with probability $\left((n-n_1)^2 m\right)^{-1} \cdot$ Pr(E.2.1.1), which contradicts **Proposition 3**, unless Pr(E.2.1.1) is negligible.

Event E.2.1.2: We modify the analysis of event E.1.1.2 slightly. Here the challenger aborts when $\mathscr{A}$ activates a session matching the t^{th} session at ID_i. If $\mathscr{A}$ succeeds and $\mathscr{S}$ guesses correctly the test session then $\mathscr{S}$ outputs $X_i^{w_{j_0} \cdot e_0 + d_{j_2}}$ (from $\sigma_0, w_{i_0}, d_0, e_0$). $\mathscr{S}$ is polynomial, and if E.2.1.2 occurs with non-negligibe probability, then W_{j_0} and $X_i^{w_{j_0} \cdot e_0 + d_{j_2}}$ with non-negligible probability. Hence, using the "oracle replay technique" [31], $\mathscr{S}$ yields a polynomial time CDH solver, which succeeds with non-negligible probability; contradicting the GDH assumption.

Event E.2.2: There are two cases to distinguish in the analysis of E.2.2. The reason is the difficulties that arise without matching when simulating a session initiated at $\hat{B}$ without matching session. Suppose an attacker, which does the following at some point of its execution as shown in Algorithm 1:

Let $\mathscr{B}$ be a class of polynomial time adversaries which at some point at time execute Sequence 1.

If event E.2.2 occurs with non-negligible probability, and if the adversary $\mathscr{A} \notin \mathscr{B}$, we can build a polynomial time CDH solver $\mathscr{S}$ using $\mathscr{A}$ that succeeds with non-negligible probability.

Suppose that E.2.2 occurs with non-negligible probability. If $\mathscr{A} \notin \mathscr{B}$ be a polynomial time attacker, using $\mathscr{A}$ we can build a polynomial time FXCR-1 signature forger, which succeds with non-negligible probability. We modify the analysis of event E.1.1.1 as follows:

- $\mathscr{S}$ takes as input $W_{i0}, X_j \in_R \mathbb{G}$.

Algorithm 1: Sequence 1.

1 $\mathscr{A}$ issues the activation message (ID_j, P_i) to obtain (F_j, V_j, W_j, X_j)
2 Issue an arbitrary number of digest queries on $\left(F_j, V_j, W_j, X_j Z_i = \langle F_i, V_i, W_i, X_i \rangle, ID_j, \hat{P}_i\right)$ where $Z_i \in_R \mathbb{G}^4$.
3 Choose $Z_i' \in_R \{Z_i\}$ and send the activation query $\left(\hat{P}_i, ID_j, F_j, V_j, W_j, Z_i' = \langle F_i', V_i', W_i', X_i' \rangle\right)$
4 Issue a *SecretExponentReveal* or a *SessionSignatureReveal* query on the session $\left(ID_j, \hat{P}_i, F_j, V_j, W_j, X_j Z_i', \mathscr{I}\right)$

- $\mathscr{S}$ chooses $i, j \in_R \{k \mid \hat{P}_k$ follows the second implementation approach$\}$; choose $d_{i1} \in_R \mathbb{Z}_p^*$ and sets $X_i = g^{d_{i1}}$.
- At the activation query $\left(\hat{P}_l, \hat{B}, W_i\right)$ query, $\mathscr{S}$ does the following:
  - $\mathscr{S}$ chooses $s_B \in_R \mathbb{Z}_p^*, \quad d \in_R \mathbb{Z}_p^*$ and sets $W_j = \left(g^{s_B} \cdot X_j^{-1}\right)^{d^{-1}}$. If there is some d' such that $\left(W_i, W_j, \hat{P}_l, ID_j, d'\right)$ already belongs to L_{h1}, $\mathscr{S}$ aborts, else $\mathscr{S}$ appends $\left(W_i, W_j, \hat{P}_l, ID_j, d'\right)$ to L_{h1}. $\mathscr{S}$ creates a session with identifier $sid' = \left(\hat{B}, \hat{P}_l, F_j, V_j, W_j, X_j, F_i, V_i, W_i, X_i \mathscr{R}\right)$ and completes the session sid' and provides $\mathscr{A}$ with $\left(ID_j, \hat{P}_l, F_j, V_j, W_j, X_j\right)$.
- At $\mathscr{A}$'s activation query $\left(ID_j, \hat{P}_l\right)$, $\mathscr{S}$ does the following:
  - $\mathscr{S}$ chooses $s_B \in_R \mathbb{Z}_p*, d \in_R \mathbb{Z}_p*$ and sets $W_j = \left(g^{s_B} \cdot X_j^{-1}\right)^{e^{-1}}$. If there is some W_i and e' such that $\left(W_j, W_i, ID_j, \hat{P}_l, e'\right)$ already belongs to L_{h1}, $\mathscr{S}$ aborts. $\mathscr{S}$ creates a session with identifier $sid' = \left(ID_j, \hat{P}_l, F_j, V_j, W_j, X_j \star, \mathscr{R}\right)$, and provides $\mathscr{A}$ with $\left(ID_j, \hat{P}_l, F_j, V_j, W_j, X_j\right)$. Later, when $\mathscr{A}$ issues activation query of the form $\left(ID_j, \hat{P}_l, W_j, W_i\right)$, $\mathscr{S}$ sets $e = \left(W_j, W_i, ID_j, \hat{P}_l\right)$ and completes the session $\left(ID_j, \hat{P}_l, F_j, V_j, W_j, X_j, F_i, V_i, W_i, X_i \mathscr{I}\right)$
- When $\mathscr{A}$ activates the t^{th} session at ID_i, if the peer is not ID_j, $\mathscr{S}$ aborts, else $\mathscr{S}$ provides $\mathscr{A}$ with $(ID_i, ID_j, F_i, V_i, W_{i0}, X_i)$.
- $\mathscr{S}$ aborts if any of the following happens:
  - $\mathscr{A}$ activates at ID_j a session matching the t^{th} session at ID_i.
  - $\mathscr{A}$ halts with a test session different from the t^{th} session at ID_i.
  - $\mathscr{A}$ issues a *SessionKeyReveal(sid)* query.
  - $\mathscr{A}$ issues an $EstablishParty(ID_i)$ or $EstablishParty(ID_j)$ query.
  - $\mathscr{A}$ issues a $Corrupt(ID_j)$ query.
  - $\mathscr{A}$ issues a $SessionSignatureReveal(sid)$ or $SecretExponentReveal(sid)$ query.
- If $\mathscr{A}$ comes up with a guess σ_0, $\mathscr{S}$ outputs : $\left(\sigma_0 \left(W_{j0}^{e_0} \cdot X_j\right)^{-d_{i_1}}\right)^{d_0^{-1}} = (W_{i0})^{w_{j0} \cdot e_0 + d_{j_1}}$ as a guess for FXCR-1 forgery on challenge W_{i0} and message (ID_i, ID_j), with respect to X_j.

The above simulation of $\mathscr{A}$'s environment remains perfect except with negligible probability. The problem happens when the same ephemeral component W_j is chosen twice in sessions involving ID_j with the same peer $\hat{P}_l$. However this happens with probability less than $\frac{m}{q}$ (which is negligible). Hence the simulation of E.2.2 almost remains perfect. When $\mathscr{A}$ comes up with the proper forgery, and $\mathscr{S}$ properly guesses the test session, $\mathscr{S}$ outputs a valid FXCR-1 forgery on challenge W_{i0} and message (ID_i, ID_j). $\mathscr{S}$ succeeds with probability $\left((n-n_1)^2 m\right)^{-1} \cdot \Pr(\text{E.2.2})$ which contradicts **Theorem 1**, unless Pr(E.2.2) is negligible. Except with negligible probability E.2.2 cannot occur for attackers not in $\mathscr{B}$.

For attackers in $\mathscr{B}$ instead of doing a simulation we will bound their success probability by another class of attackers which can be efficiently simulated. Let $\mathbb{B} \in \mathscr{B}$ and let $d(|q|)$ be an upper bound on the number of Z_i's $\mathscr{A}$

chooses at step **2** of the algorithm Sequence 1. For all $\mathbb{B} \in \mathscr{B}$, let $\mathbb{B}_R$ be an attacker, which in addition to $\mathbb{B}$'s input receives the vector $\mathbf{v} = \left((i_i^0, \cdots, i_i^m), (Z_{11}, \cdots, Z_{1d}), \cdots, (Z_{m1}, \cdots, Z_{md})\right)$ where Z_{ij}'s $\in_R \mathbb{G}^*$and $i_i^0 \in [d+1]$ (recall there are n parties and each party can activated at most m times), and performs exactly the same way as $\mathbb{B}$, except that whenever $\mathbb{B}$ executes the sequence of queries Sequence 1 for the l-th time, $\mathbb{B}_R$ executes the modified sequence Sequence 2. And, when $\mathbb{B}$ uses, for any other computation, a Z_i chosen during the l-th execution Sequence 1, $\mathbb{B}_R$ uses Z_{li}.

Algorithm 2: Sequence 2.

1. $\mathscr{A}$ issues the activation message (B, P_i) to obtain (F_B, V_B, W_B)
2. Issue an arbitrary number of digest queries on $\left(F_B, V_B, W_B, Z_{li} = \langle F_{li}, V_{li}, W_{li}\rangle, \hat{B}, \hat{P}_i\right)$ where $Z_{li} \in_R \{Z_{l1} \cdots, Z_{ld}\}$.
3. Sends the activation message $\left(\hat{B}, \hat{P}_i, F_j, V_j, W_j, X_j, Z_{i_l^0}\right)$
4. Issue a *SecretExponentReveal* or a *SessionSignatureReveal* query on the session $\left(ID_j, \hat{P}_i, F_j, V_j, W_j, X_j, Z_{i_l^0}, \mathscr{I}\right)$

Let $\mathbf{V}$ be the set of resource vectors and t the number of times $\mathbb{B}$ executes Sequence 1 ($t \leq m$). For $\mathbf{v} \in \mathbf{V}$, we say that $\mathbb{B}_R(\mathbf{v})$ matches $\mathbb{B}$, if for all $l \in [t+1]$, the l-th time $\mathbb{B}$ executes Sequence 1, it chooses $\{Z_{l1} \cdots, Z_{ld}\}$ in step 2, and poses $\left(ID_j, \hat{P}_i, F_j, V_j, W_j, X_j, Z_{i_l^0}\right)$ at step 3. If $\mathbb{B}_R(\mathbf{v})$ matches $\mathbb{B}$, $\Pr(\text{E.2.2}_{\mathbb{B}}) = \Pr(\text{E.2.2}_{\mathbb{B}_R}(\mathbf{v}))$. For $\mathbb{B} \in \mathscr{B}$, we say $\mathbf{v} \in \mathbf{V}$ possible if there is nonzero probability that $\mathbb{B}_R(\mathbf{v})$ matches $\mathbb{B}$. Let $Poss(\mathbf{V})$ denote the set of possible resource vectors. For every run of $\mathbb{B}$, there is some $\mathbf{v} \in Poss(\mathbf{V})$ such that $\mathbb{B}_R(\mathbf{v})$ matches $\mathbb{B}$. Hence:

$$\Pr(\text{E.2.2}_{\mathbb{B}}) \leq \max_{\mathbf{v} \in Poss(\mathbf{V})} \Pr(\text{E.2.2}_{\mathbb{B}_R}(\mathbf{v}))$$

To show that the success probability in E.2.2 of an attacker $\mathbb{B} \in \mathscr{B}$ is negligible, it suffices to show that $\Pr(\text{E.2.2}_{\mathbb{B}_R}(\mathbf{v}))$ is negligi ble for all $\mathbf{v} \in Poss(\mathbf{V})$.For this purpose, the simulator is provided with $\mathbf{v}$. Note here that the simulator and the attacker $\mathbb{B}_R$ are jointly used to create a FXCR-1 forgery. We modify the activation of the sessions initiated at ID_j as follows:

- When the attacker issues the activation message (ID_j, P_i) for the l-th time, the simulator $\mathscr{S}$ does the following
  - Choose $s_B \in_R [q]$, $e \in_R \mathbb{Z}_p*$ and set $W_j = \left(g^{s_B} X_j^{-1}\right)^{e^{-1}}$ and $e = H_1\left(F_j, V_j, W_j, X_j, Z_{i_l^0}, ID_j, \hat{P}_l\right)$.
  - Creates a session with identifier $\left(\hat{B}, \hat{P}_i, F_j, V_j, W_j, X_j\star, \mathscr{I}\right)$ and provide $\mathscr{A}$ with $\left(ID_j, \hat{P}_i, F_j, V_j, W_j, X_j\right)$.
- When the attacker issues a $SecretExponentReveal\left(\mathscr{I}, ID_j, \hat{P}_i, F_j, V_j, W_j, Z_{i_l^0}\right)$, the simulator provides the attacker with s_B.

The simulation is consistent for all $\mathbf{v} \in Poss(\mathbf{V})$ and $\mathbb{B}_R$. As $\mathscr{S}$ knows, from the resource vector, what will be the incoming ephemeral public key, *SecretExponentReveal* and *SessionSignatureReveal* queries are consistently simulated. If $\mathbb{B}_R(\mathbf{v})$ succeeds in E.2.2 with non-negligible probability, $\mathscr{S}$ succeeds in FXCR-1 forgery with non-negligible probability. Hence $\Pr(\text{E.2.2}_{\mathbb{B}_R}(\mathbf{v}))$ is negligible, for all $\mathbf{v} \in Poss(\mathbf{V})$ and $\mathbb{B}_R$. Hence we conclude $\Pr(\text{E.2.2}_{\mathbb{B}})$ is negligible.

Analysis of E.2.3. Here the matching session to the test session does not exist and both the parties follow different implementation approach. Event E.2.3 can be analyzed as follows:

- E.2.3.1: E.2.3 $\wedge$ ID_i follows the first implemetation approach.
- E.2.3.2: E.2.3 $\wedge$ ID_i follows the second implemetation approach.

E.2.3.1 can be further analysed as follows:

- E.2.3.1.1: E.2.3.1 $\wedge$ $\mathscr{A}$ issues $Corrupt(ID_i)$ query.

- E.2.3.1.2: E.2.3.1 $\wedge$ $\mathscr{A}$ issues $EphemeralKeyReveal(sid)$ where sid is the test session.

Event E.2.3.1.1: Let E.2.3.1.1 occur with non-negligible probability. We modify the analysis of E.1.1.1 to take as input $w_{i0}, X_j \in_R \mathscr{G}$ and simulate ID_j's role as in E.2.2 (ID_i's role is simulated as in E.1.1.1). The arguments for the class of attackers $\mathscr{B}$ for which we bound the success probability by a different efficiently simulatable class of attackers remains valid. So the success probability of $\mathscr{A}$ is related to the success probability of the FXCR-1 forger in the sense that if $\mathscr{A}$ succeeds with non-negligible probability, then the FXCR-1 forger can forge FXCR-1 signature forger which succeeds with non-negligible probability in probabilistic polynomial time; contradicting Proposition 1.

Event E.2.3.1.2: Let E.2.3.1.2 occur with non-negligible probability. We modify the analysis of E.1.1.1 to take as input $X_i, X_j \in_R \mathscr{G}$. Also if $\mathscr{A}$ activates a session matching the t^{th} session at A, the challenger aborts. We simulate ID_i's role as in E.1.1.2 and ID_j's role as in E.2.2, and the arguments against the class of attackers $\mathscr{B}$ also holds good here. From any valid forgery σ_0, $\mathscr{S}$ outputs $\sigma_0\left(W_{j0}^{e_0} \cdot X_j\right)^{-w_{i0} \cdot d_0} = (X_i)^{w_{j0} \cdot e_0 + d_{j_1}}$.

Event E.2.3.2: If $\mathscr{A}$ follows the second implementation approach, we make $\mathscr{S}$ take as input $X_i, X_j \in_R \mathscr{G}$, simulate ID_i's role as that of $\hat{B}$ in E.2.2, and ID_j's role as in E.1.1.2. If $\mathscr{A}$ activates the t^{th} session at A, the solver $\mathscr{S}$ chooses $w_{i0} \in_R \mathbb{Z}_p^*$ as gives (ID_i, ID_j, W_{i0}) as input to $\mathscr{A}$. Also if $\mathscr{A}$ activates a session matching the t^{th} session at ID_i, the challenger aborts. If $\mathscr{A}$ succeeds with non-negligible probability, $\mathscr{S}$ outputs $(X_i)^{w_{j0} \cdot e_0 + d_{j_1}}$ as solution to the CDH problem and using the oracle replay technique, $\mathscr{S}$ yields an efficient CDH solver. This contradicts the GDH assumption. Hence E.2.3 cannot occur except with negligible probability. □

7 Conclusion

In this paper we proposed a new security model for ID-AKE protocols that is resilient to leakage of intermediate values that may be possible if RAM-Scraper-like malware are planted on the host machine. These malwares once implanted in the host machine gets all the sensitive data that is stored in the insecure memory and leak it to the adversary via remote logins. The main advantage of our proposed protocol is is that it is single round, pairing-free and completely asynchronous. To the best of our knowledge our ID-AKE protocol is the first to withstand these stronger class of attacks apart from the regular attacks that are handled by the black box models like the eCK model. Our protocol also admits a tight security proof under the GDH assumption in the random oracle model. The next advantage is that forking lemma is not used in the security reduction contributing to the tight security reduction. It would be interesting to achieve the same security guarantees in the standard model. In particular we leave open the construction of ID-AKE in standard model in our new ID-seCK model resilient to these advanced classes of threats.

References

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

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology, Proc. of CRYPTO'84, Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 196. Springer Berlin Heidelberg, August 1984, pp. 47–53.

[3] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology - Proc. of EUROCRYPT'01, Innsbruck, Austria*, ser. Lecture Notes in Computer Science, vol. 2045. Springer Berlin Heidelberg, May 2001, pp. 453–474.

[4] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *Proc. of the 1st International Conference on Provable Security (ProvSec'07), Wollongong, Australia*, ser. Lecture Notes in Computer Science, vol. 4784. Springer Berlin Heidelberg, November 2007, pp. 1–16.

[5] A. P. Sarr, P. Elbaz-Vincent, and J.-C. Bajard, "A new security model for authenticated key agreement," in *Proc. of the 7th International Conference on Security and Cryptography for Networks (SCN'10), Amalfi, Italy*, ser. Lecture Notes in Computer Science, vol. 6280. Springer Berlin Heidelberg, September 2010, pp. 219–234.

[6] N. Huq, "Defending against pos ram scrapers," https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-defending-against-pos-ram-scrapers.pdf, 2015, [Online; Accessed on March 30, 2016].

[7] V. D. S. Alert, https://usa.visa.com/dam/VCOM/download/merchants/Bulletin-Memory-Parser-Update-012014.pdf, [Online; Accessed on March 30, 2016].

[8] M. Scott, "MIRACL-a multiprecision integer and rational arithmetic C/C++ library. Shamus Software Ltd," 2013.

[9] C. G. Günther, "An identity-based key-exchange protocol," in *Advances in Cryptology - Proc. of EUROCRYPT'89, Belgium*, ser. Lecture Notes in Computer Science, vol. 434. Springer Berlin Heidelberg, April 1989, pp. 29–37.

[10] S. Saeednia, "Improvement of gunther's identity-based key exchange protocol," *Electronics Letters*, vol. 36, no. 18, pp. 1535–1536, 2000.

[11] D. Fiore and R. Gennaro, "Making the diffie-hellman protocol identity-based," in *Topics in Cryptology - Proc. of the Cryptographers' Track at the RSA Conference 2010 (CT-RSA'10), San Francisco, California, USA*, ser. Lecture Notes in Computer Science, vol. 5985. Springer Berlin Heidelberg, March 2010, pp. 165–178.

[12] D. Mishra and S. Mukhopadhyay, "Cryptanalysis of pairing-free identity-based authenticated key agreement protocols," in *Proc. of the 9th International Conference Information Systems Security (ICISS'13), Kolkata, India*, ser. Lecture Notes in Computer Science, vol. 8303. Springer Berlin Heidelberg, December 2013, pp. 247–254.

[13] Q. Cheng and C. Ma, "Ephemeral key compromise attack on the ib-ka protocol," *IACR Cryptology ePrint Archive*, vol. 2009, p. 568, 2009.

[14] X. Cao, W. Kou, and X. Du, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges." *Information Sciences*, vol. 180, no. 15, pp. 2895–2903, 2010.

[15] S. H. Islam and G. Biswas, "An improved pairing-free identity-based authenticated key agreement protocol based on {ECC}," *Procedia Engineering*, vol. 30, pp. 499–507, 2012.

[16] S. S. Vivek, S. S. D. Selvi, L. R. Venkatesan, and C. P. Rangan, "Efficient, pairing-free, authenticated identity based key agreement in a single round," in *Proc. of the 7th International Conference on Provable Security (ProvSec'13), Melaka, Malaysia*, ser. Lecture Notes in Computer Science, vol. 8209. Springer Berlin Heidelberg, October 2013, pp. 38–58.

[17] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama, "Strongly secure authenticated key exchange from factoring, codes, and lattices," *Designs, Codes and Cryptography*, vol. 76, no. 3, pp. 469–504, 2015.

[18] S. S. Vivek, S. S. D. Selvi, and C. P. Rangan, "Stronger public key encryption schemes withstanding ram scraper like attacks," *IACR Cryptology ePrint Archive*, vol. 2012, p. 118, 2012.

[19] T. S. Messerges, E. Dabbish, R. H. Sloan *et al.*, "Examining smart-card security under the threat of power analysis attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541–552, 2002.

[20] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology - CRYPTO'96, Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 1109. Springer Berlin Heidelberg, August 1996, pp. 104–113.

[21] M. Hutter, S. Mangard, and M. Feldhofer, *Power and EM Attacks on Passive 13.56\,\ textrm {MHz} RFID Devices*. Springer, 2007.

[22] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *Computer Security - ESORICS'11, Leuven, Belgium*, ser. Lecture Notes in Computer Science, vol. 6879. Springer Berlin Heidelberg, August 2011, pp. 355–371.

[23] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," in *Advances in Cryptology-CRYPTO'09, Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 5677. Springer Berlin Heidelberg, August 2009, pp. 36–54.

[24] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs, "Cryptography against continuous memory attacks," in *Proc. of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10), Las Vegas, Nevada, USA*. IEEE, October 2010, pp. 511–520.

[25] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, "Public-key encryption schemes with auxiliary inputs," in *Proc. of the 7th Theory of Cryptography Conference on Theory of Cryptography*