A mechanical approach to derive identity-based protocols from Diffie–Hellman-based protocols☆

Kim-Kwang Raymond Choo[a], Junghyun Nam[b,*], Dongho Won[c]

[a]Information Assurance Research Group, Advanced Computing Research Centre, University of South Australia, Mawson Lakes, SA 5095, Australia

[b]Department of Computer Engineering, Konkuk University, 322 Danwol-dong, Chungju-si, Chungcheongbuk-do 380-701, Korea

[c]Department of Computer Engineering, Sungkyunkwan University, 300 Cheoncheon-dong, Suwon-si, Gyeonggi-do 440-746, Korea

Abstract

We describe a mechanical approach to derive identity-based (ID-based) protocols from existing Diffie–Hellman-based ones. As case studies, we present the ID-based versions of the Unified Model protocol, UMP-ID, Blake-Wilson, Johnson & Menezes (1997)'s protocol, BJM-ID, and Krawczyk (2005)'s HMQV protocol, HMQV-ID. We describe the calculations required to be modified in existing proofs. We conclude with a comparative security and efficiency of the three proposed ID-based protocols (relative to other similar published protocols) and demonstrate that our proposed ID-based protocols are computationally efficient.

Keywords: Key establishment protocols, Identity-based (ID-based) protocols, Diffie–Hellman-based protocols

1. Introduction

Key distribution is one of the most fundamental problems in cryptography, and was revolutionized by the introduction of the key exchange protocol by Diffie and Hellman in 1976 [20]. The Diffie–Hellman (DH) protocol illustrated that:

> *arbitrary two parties even with no prior acquaintance and no secure physical/electronic channels can establish a shared secret key (called a* session key*) simply by exchanging their public keys over an insecure public network*

☆This is the authors' post-print version of the paper accepted by Information Sciences, and citation to this paper should be as follows: Choo K-K R, Nam J and Won D. A mechanical approach to derive identity-based protocols from Diffie-Hellman-based protocols. Information Sciences [In press; Also available from `http://eprint.iacr.org/2014/358.pdf`].

*Corresponding author. Email: jhnam@kku.ac.kr; Tel.: +82-43-840-3608; fax: +82-43-840-3600.

as long as integrity of public keys is guaranteed and the underlying computational problem (known as the computational Diffie-Hellman problem*) is hard.*

We note that the public keys exchanged in the DH protocol are usually ephemeral (short-term) rather than static (long-term) keys, although this (i.e. Whether the keys are ephemeral or static?) was not in the original protocol specification. Perhaps, this was not an issue at that time. While public key cryptography facilitates key distribution over an insecure communication channel, the integrity of public keys is crucial for security against an active adversary – it is well known that the basic (unauthenticated) DH protocol is susceptible to active man-in-the-middle attacks.

Many of the popular key establishment protocols are based on the DH key exchange and are implicitly authenticated via public key certificates[1] [25, 41]. Examples include the MTI protocol [35], the Unified Model protocol (UMP) [2, 8], the MQV protocol [37, 34], and the HMQV protocol [31]. Throughout the paper, we will use the term "DH-based protocols" to refer to these implicitly authenticated DH-based protocols. A key goal of DH-based protocols is to achieve the same level of efficiency as the basic DH protocol, both in terms of communication and computation, when the possible transmission and verification of public key certificates are excluded from consideration. The design and security of DH-based protocols have been extensively studied over the last decades and are now fairly well-understood. For example, some recent DH-based protocols were proven secure in the extended Canetti-Krawczyk (eCK) model [33, 47, 30].

While public key certificates have been widely used to bind public keys to identities, their management has turned out to be more challenging than was initially anticipated. The quest for a solution to this problem has led to the invention of identity-based (ID-based) cryptography [42]. At the price of key escrow, ID-based cryptography eliminates the need for certificates by allowing parties to use their identity as their public key. Typically, we would already know the identity of our communication peer and, thus, do not need a signed certificate for it. This is of great benefit in simplifying the management of public keys [40]. From an ID-based scheme user's perspective, an obvious benefit is an absence of certificate transmission and verification.

In the past decade, we have witnessed a surge of interest in ID-based cryptography, particularly the use of elliptic curve pairings to realize cryptographic structures that seemed impossible before. To illustrate how elliptic curve pairings can be used to build novel cryptographic schemes with interesting properties, we refer the reader to the work of Al-Riyami [1]. Published schemes include a number of ID-based key establishment protocols using pairing, which we will refer to simply as "ID-based protocols". Examples include the protocols of Smart [45], Shim [43], Chen and Kudla [14], Choie,

[1] A public key certificate is an electronic document signed by a trusted third party (called a *certificate authority*) to prove that a given public key belongs to a specific individual.

Jeong and Lee [16], Xie [51], McCullagh and Barreto [36], Wang, Cao and Cao [50], and Wang [49].

The security properties required for key establishment protocols are well studied, and an excellent overview is presented by Blake-Wilson and Menezes [9]. The most basic property is that a passive adversary eavesdropping on the protocol should be unable to obtain the session key. Other desirable properties include:

Known key security. It is often reasonable to assume that the adversary will be able to obtain session keys from any session different from the one under attack. A protocol has known key security if it is secure under this assumption. This is generally regarded as a standard requirement for key establishment protocols.

Unknown key-share security. Sometimes the adversary may be unable to obtain any useful information about a session key, but can deceive the protocol principals about the identity of the peer entity. Such an attack was first described by Diffie, van Oorschot and Wiener [21], and can result in principals giving away information to the wrong party or accepting data as coming from the wrong party.

As discussed by Boyd and Mathuria [12, Chapter 5.1.2], a malicious adversary $\mathcal{A}$ need not obtain the session key to profit from this attack. Consider the scenario whereby Alice will deliver some information of value (such as e-cash) to Bob. Since Bob believes the session key is shared with $\mathcal{A}$, $\mathcal{A}$ can claim this credit deposit as his. Also, $\mathcal{A}$ can exploit such an attack in a number of ways if the established session key is subsequently used to provide encryption or integrity [29]. Consequently, security against unknown key-share attacks is regarded as a standard requirement.

Forward secrecy. When the static key of an entity is compromised, the adversary will be able to masquerade as that entity in any future protocol runs. However, the situation will be even worse if the adversary can also use the compromised static key to obtain session keys that were established before the compromise. Protocols that prevent this are said to provide forward secrecy. Since there is usually a computational cost in providing forward secrecy, it is sometimes sacrificed in the interest of efficiency.

Forward secrecy in the setting of ID-based cryptography is similar as in conventional public key cryptography. However, there is an additional concern since the master key of the key generation center (KGC) is another secret that could become compromised. There could exist a protocol that provides forward secrecy in the usual sense but gives away old session keys if the master key becomes known. We will say that a protocol that retains confidentiality of old session keys even when the master key is known provides *KGC forward secrecy* (KGC-FS). As the static keys of all users can be easily computed from the master key, it is clear that KGC forward secrecy implies forward secrecy.

Key compromise impersonation resistance. Another problem that may occur when the static key of an entity A is compromised is that the adversary may be able to masquerade not only *as* A but also *to* A as another party B. Such a protocol is said to allow key compromise impersonation. Resistance to such attacks is often seen as desirable.

A survey by Boyd and Choo [11] shows that many existing ID-based protocols have been published without a careful security analysis or a systematic comparison with alternatives, highlighting the need for more rigorously tested ID-based protocols. In addition, their survey suggests some interesting similarities between ID-based protocols and various DH-based protocols. They then conjectured that these similarities may well extend to the security properties of these protocols, and the key mapping technique described in Table 1 of Section 3.2 was designed by Choo in 2005. In 2009, Wang [48] independently proposed a similar technique, referred to as the key substitution rules. Although the motivations behind both techniques were similar, the actual rules of mapping are different and the security of the resultant ID-based protocols was not discussed [48].

In this paper, our main contribution is to present a systematic approach to mechanically derive provably-secure ID-based protocols from their DH-based versions. In our approach, we

1. first propose ID-based versions of DH-based protocols based on some rules for parameters conversion,
2. describe the computational assumptions required to be modified due to the parameters conversion, and
3. describe the calculations required to be modified in comparison to the original proof.

To demonstrate that our approach is independent of the underlying security model (i.e. our approach can be applied to protocols proven secure in different security models), we use three popular protocols — the UMP protocol [2, 8], the BJM protocol [8, protocol 4], and the HMQV protocol [31] — as case studies. UMP was proven secure in a restricted model where the adversary is not allowed to reveal session keys [8]. We provide a proof of security for the ID-based version of UMP, which we denote by UMP-ID$_0$, in the same restricted model. We also show that a slight variant of UMP-ID$_0$, denoted as UMP-ID, can be proven secure in the model of Bellare and Rogaway (BR) [7] which does not restrict the adversary from revealing session keys. The original BJM protocol does not carry any proof of security but its variant due to Kudla and Paterson [32] was proven secure in a model adapted from the BR model to capture the notion of key compromise impersonation resistance. We prove the security of the ID-based version of BJM, BJM-ID, in the same model as the one used for the BJM variant of Kudla and Paterson. Lastly, the HMQV protocol was proven secure in the model of Canetti and Krawczyk (CK) [13]. As suggested by Choo, Boyd and

Hitchcock [18], protocols proven secure in the BR model are not necessarily secure in the CK model but the converse is true; for example, the adversary is allowed to obtain the ephemeral private keys of parties only in the CK model. For the ID-based version of HMQV (HMQV-ID), we provide a proof of forward security in the eCK model [33], which is an extesion of the (original) CK model.

The next section presents the mathematical preliminaries and an overview of both the BR and eCK models. In Section 3, we present the mechanics of mapping the protocol parameters and the computational assumptions from DH-based to ID-based protocols. In Sections 4 to 6, we present the ID-based versions of UMP, BJM and HMQV, followed by the calculations required to be modified in their existing proofs. We conclude with a comparative security and efficiency of the derived ID-based protocols (relative to other similar published protocols) in Section 7.

2. Preliminaries

In cryptographic algorithms, the value of k is important since negligibility of functions and complexity of algorithms are often parameterized by k (e.g., the size of cryptographic groups and key lengths within those algorithms). The larger the value of k is, the more computation is required to run an algorithm. The value k relates to the bounds on an adversary's success probability (i.e., k is often known as the security parameter). All cryptographic algorithms in this paper receive the security parameter k as input and their security is measured in k. We recall the definition of a negligible function.

Definition 1 (A negligible function [6]). A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if it approaches zero faster than the reciprocal of any polynomial. That is, for every $c \in \mathbb{N}$ there is an integer k_c such that $f(k) \leq k^{-c}$ for all $k \geq k_c$.

In general, a cryptographic algorithm is considered secure if for any adversary against the algorithm, its success probability is a negligible function of the security parameter k.

2.1. Bilinear maps from elliptic curve pairings

Using the notation of Boneh and Franklin [10], we let $\mathbb{G}_1$ be an additive group of prime order q with $|q| = k$, where k is the security parameter, and $\mathbb{G}_2$ be a multiplicative group of the same order q. We assume the existence of a map $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Typically, $\mathbb{G}_1$ will be a subgroup of the group of points on an elliptic curve over a finite field, $\mathbb{G}_2$ will be a subgroup of the multiplicative group of a related finite field and the map $\hat{e}$ will be derived from either the Weil or Tate pairing on the elliptic curve[2]. The mapping $\hat{e}$ must be efficiently computable and has the following properties.

[2]We note that Tate pairing appears to be more computationally efficient than Weil pairing [22, 27].

Bilinearity. For $Q, W, Z \in \mathbb{G}_1$, both

$$\hat{e}(Q, W + Z) = \hat{e}(Q, W) \cdot \hat{e}(Q, Z) \quad \text{and} \quad \hat{e}(Q + W, Z) = \hat{e}(Q, Z) \cdot \hat{e}(W, Z).$$

Non-degeneracy. For some elements $P, Q \in \mathbb{G}_1$, we have $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$.

Computability. For some elements $P, Q \in \mathbb{G}_1$, we have an efficient algorithm to compute $\hat{e}(P, Q)$.

A bilinear map, $\hat{e}$, is said to be an *admissible* bilinear map if it satisfies all three properties. Since $\hat{e}$ is bilinear, the map $\hat{e}$ is also symmetric.

2.2. Computational problems and assumptions

In the provable security paradigm, the underlying computational assumptions employed form the basis of security for the protocol. In the definition of such assumptions, protocol designers have various degrees of freedom related to the concrete mathematical formulation of the assumption (e.g., what kind of attackers are considered or over what values the probability spaces are defined). In the case of DH-based protocols and ID-based protocols, security is usually proved by finding a reduction to the Computational Diffie–Hellman (CDH) problem [20] or its variants and the Bilinear Diffie–Hellman (BDH) problem [10] or its variants respectively, whose intractability is assumed. In other words, we assume that there exists no probabilistic polynomial-time (PPT) algorithm whose advantage in solving the problem is non-negligible. In the following we briefly describe the CDH and BDH problems and their variants. Assume that DH-based protocols work in a finite cyclic group $\mathbb{G}$ of prime order q (with $|q| = k$) while ID-based protocols operate on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ which are defined as above. Let g and P be generators of $\mathbb{G}$ and $\mathbb{G}_1$, respectively.

Computational Diffie–Hellman (CDH) problem. Given an instance of $(g^a, g^b) \in \mathbb{G}^2$ (or $(aP, bP) \in \mathbb{G}_1^2$), where $a, b \in_R \mathbb{Z}_q^*$, output $g^{ab} \in \mathbb{G}$ (or $abP \in \mathbb{G}_1$ respectively).

An algorithm, $\mathcal{A}_{\text{CDH}}$, running in time t has advantage ϵ in solving the CDH problem in $\mathbb{G}$ (or $\mathbb{G}_1$) if $\Pr[\mathcal{A}_{\text{CDH}}(g^a, g^b) = g^{ab}] \geq \epsilon$ (or $\Pr[\mathcal{A}_{\text{CDH}}(aP, bP) = abP] \geq \epsilon$), where the probability is over the random choice of $a, b \in \mathbb{Z}_q^*$, the random choice of $g \in \mathbb{G}^*$ (or $P \in \mathbb{G}_1^*$ respectively), and the random bits of $\mathcal{A}_{\text{CDH}}$.

Decisional Diffie–Hellman (DDH) problem. Distinguish between two distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where $a, b, c \in_R \mathbb{Z}_q^*$.

An algorithm, $\mathcal{A}_{\text{DDH}}$, running in time t has advantage ϵ in solving the DDH problem in $\mathbb{G}$ if $|\Pr[\mathcal{A}_{\text{DDH}}(g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}_{\text{DDH}}(g^a, g^b, g^c) = 1]| \geq \epsilon$, where the probability is over the random choice of $a, b, c \in \mathbb{Z}_q^*$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}_{\text{DDH}}$.

Gap Diffie–Hellman (GDH) problem. Given an instance of $(g^a, g^b) \in \mathbb{G}^2$, as well as an oracle $\mathcal{O}_{\mathrm{DDH}}(\cdot,\cdot,\cdot)$ that solves the DDH problem in $\mathbb{G}$, output $g^{ab} \in \mathbb{G}$. Here, the oracle $\mathcal{O}_{\mathrm{DDH}}(\cdot,\cdot,\cdot)$ outputs 1 if the given problem instance is a decisional Diffie–Hellman tuple, and 0 otherwise.

An algorithm, $\mathcal{A}_{\mathrm{GDH}}$, running in time t has advantage ϵ in solving the GDH problem in $\mathbb{G}$ if $\Pr[\mathcal{A}_{\mathrm{GDH}}(g^a, g^b, \mathcal{O}_{\mathrm{DDH}}(\cdot,\cdot,\cdot)) = g^{ab}] \geq \epsilon$, where the probability is over the random choice of $a, b \in \mathbb{Z}_q^*$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}_{\mathrm{GDH}}$.

Bilinear Diffie–Hellman (BDH) problem. Given an instance of $(aP,\ bP,\ cP) \in \mathbb{G}_1^3$, where $a, b, c \in_R \mathbb{Z}_q^*$, output $\hat{e}(P,P)^{abc} \in \mathbb{G}_2$.

An algorithm, $\mathcal{A}_{\mathrm{BDH}}$, running in time t has advantage ϵ in solving the BDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ if $\Pr[\mathcal{A}_{\mathrm{BDH}}(aP, bP, cP) = \hat{e}(P,P)^{abc}] \geq \epsilon$, where the probability is over the random choice of $a, b, c \in \mathbb{Z}_q^*$, the random choice of $P \in \mathbb{G}_1^*$, and the random bits of $\mathcal{A}_{\mathrm{BDH}}$.

Decisional bilinear Diffie–Hellman (DBDH) problem. Distinguish between two distributions $(aP, bP, cP, \hat{e}(P,P)^{abc})$ and $(aP, bP, cP, \hat{e}(P,P)^{d})$, where $a, b, c, d \in_R \mathbb{Z}_q^*$.

An algorithm, $\mathcal{A}_{\mathrm{DBDH}}$, running in time t has advantage ϵ in solving the DBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ if $|\Pr[\mathcal{A}_{\mathrm{DBDH}}(aP, bP, cP, \hat{e}(P,P)^{abc}) = 1] - \Pr[\mathcal{A}_{\mathrm{DBDH}}(aP, bP,\ cP, \hat{e}(P,P)^{d}) = 1]| \geq \epsilon$, where the probability is over the random choice of $a, b, c, d \in \mathbb{Z}_q^*$, the random choice of $P \in \mathbb{G}_1^*$, and the random bits of $\mathcal{A}_{\mathrm{DBDH}}$.

Gap bilinear Diffie–Hellman (GBDH) problem. Given $(aP,\ bP,\ cP) \in \mathbb{G}_1^3$, as well as an oracle $\mathcal{O}_{\mathrm{DBDH}}(\cdot,\cdot,\cdot,\cdot)$ that solves the DBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$, output $\hat{e}(P,P)^{abc} \in \mathbb{G}_2$. Here, the oracle $\mathcal{O}_{\mathrm{DBDH}}(\cdot,\cdot,\cdot,\cdot)$ outputs 1 if the given problem instance is a bilinear Diffie–Hellman tuple, and 0 otherwise.

An algorithm, $\mathcal{A}_{\mathrm{GBDH}}$, running in time t has advantage ϵ in solving the GBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ if $\Pr[\mathcal{A}_{\mathrm{GBDH}}(aP, bP, cP, \mathcal{O}_{\mathrm{DBDH}}(\cdot,\cdot,\cdot,\cdot)) = \hat{e}(P,P)^{abc}] \geq \epsilon$, where the probability is over the random choice of $a, b, c \in \mathbb{Z}_q^*$, the random choice of $P \in \mathbb{G}_1^*$, and the random bits of $\mathcal{A}_{\mathrm{GBDH}}$.

2.3. Communication model

Participants. Let $\mathcal{U}$ be a nonempty set of participants (also called users). We assume each user $U \in \mathcal{U}$ is identified by a string, and we interchangeably use U and ID_U to refer to this identifier string. For a key exchange protocol P, each user is able to execute P multiple times with different participants, and we model this by allowing unlimited number of *instances* of each user. We use Π_U^i to denote instance i of user U, and use $\Pi_{U,U'}^i$ to denote instance i of user U attempting to establish a session key with (an instance of) user $U' \in \mathcal{U}$. An instance Π_U^i is said to *accept* when it computes a session key sk_U^i as a result of a protocol execution.

Partnering. We say, informally, that two instances are *partnered* if they participate in a protocol execution and establish a (shared) session key. Formally, partnering between instances is defined in terms of the notions of session and partner identifiers (See the work of Choo [17] on the role and the possible construct of session and partner identifiers as a form of partnering mechanism that enables the right session key to be identified in concurrent protocol executions). Session identifier (sid) is a unique identifier of a protocol session and is usually defined as a function of the messages transmitted in the session (although this may not be possible in a multi-party protocol where not all participants have the same view). sid^i_U denotes the sid of instance Π^i_U. A partner identifier (pid) is a sequence of identities of participants of a specific protocol session. Instances are given as input a pid before they can run the protocol. pid^i_U denotes the pid given to instance Π^i_U. Then, either $pid^i_U = (U, U')$ or $pid^i_U = (U', U)$ must be true, where U' is another user with whom Π^i_U believes it runs the protocol. We say that two instances, Π^i_U and $\Pi^j_{U'}$, are partnered if all the following hold: (1) both Π^i_U and $\Pi^j_{U'}$ have accepted, (2) $sid^i_U = sid^j_{U'}$, and (3) $pid^i_U = pid^j_{U'}$.

Adversary capabilities. A PPT adversary $\mathcal{A}$ has complete control over the environment (mainly the network), and its capabilities are modeled via a pre-defined set of oracle queries described below.

- $\mathsf{Send}(U, i, M)$ causes message M to be sent to instance Π^i_U. The instance computes what the protocol says to and any outgoing messages are given to $\mathcal{A}$. If this query causes Π^i_U to accept, this will also be shown to $\mathcal{A}$. If $M = (U, U')$ (or $M = (U', U)$), then the query will prompt instance Π^i_U to initiate the protocol with $pid^i_U = (U, U')$ (or $pid^i_U = (U', U)$ respectively).

- $\mathsf{Reveal}(U, i)$ causes the output of the session key sk^i_U held by Π^i_U.

- $\mathsf{Corrupt}(U)$ returns any static secret key(s) that U holds. U could be KGC in the ID-based setting and in this case, the master secret of KGC is returned in response to the query.

- $\mathsf{Test}(U, i)$ causes the oracle to choose a bit b uniformly at random. If $b = 1$, the session key sk^i_U is output; otherwise, a string is drawn uniformly from the space of session keys and output. A Test query may be asked at any time during the execution of P, but may only be asked once.

In the eCK model, the adversary is allowed to ask the $\mathsf{EphemeralKeyReveal}(U, i)$ query that will return the ephemeral private key(s) of the instance Π^i_U to the adversary. In contrast, most other models (e.g. the BR model) only allow the adversary to reveal session keys for uncorrupted parties.

Session key (SK) security. We now proceed to define the basic security, called the SK security, of protocol P. The notion of *freshness* is a key element in defining the SK security. Intuitively, a fresh instance is one that holds a session key which should not be known to the adversary $\mathcal{A}$, and an unfresh instance is one whose session key can be known by trivial means. A formal definition of freshness follows:

Definition 2 (Freshness). An instance $\Pi^i_{U,U'}$ is fresh unless one of the following occurs: (1) the adversary queries $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$, where $\Pi^j_{U'}$ is an instance partnered with Π^i_U; or (2) the adversary queries $\mathsf{Corrupt}(U)$ or $\mathsf{Corrupt}(U')$.

The SK security of protocol P is defined in the context of the following two-stage experiment:

Stage 1. $\mathcal{A}$ makes any oracle queries at will as many times as it wishes as long as (1) the Test query is not asked against an unfresh instance and (2) the test instance remains fresh until the end of the stage.

Stage 2. Once $\mathcal{A}$ decides that Stage 1 is over, it outputs a bit b' as a guess on the hidden bit b chosen by the Test oracle. $\mathcal{A}$ is said to succeed if $b = b'$.

Let Succ be the event that $\mathcal{A}$ succeeds in this experiment. Then, the advantage of $\mathcal{A}$ in attacking protocol P is defined as $\mathsf{Adv}_P(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1$.

Definition 3 (SK security). A key exchange protocol P is *SK-secure* if $\mathsf{Adv}_P(\mathcal{A})$ is negligible for any PPT adversary $\mathcal{A}$.

3. Mechanics of protocol derivation

3.1. System setup

Assume a DH-based protocol, DHP, for which the system parameters are defined as $(\mathbb{G}, q, g)$ and the static private/public keys of each $U \in \mathcal{U}$ are set to $(u \in_R \mathbb{Z}^*_q,\ g^u \in \mathbb{G})$. Given the protocol DHP, we define the following system parameters for IDP, an ID-based version of DHP:

- An additive group $\mathbb{G}_1$ with a generator P of order q, a multiplicative group $\mathbb{G}_2$ of the same order q, and a bilinear map $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$.
- A cryptographic hash function $G : \{0, 1\}^* \to \mathbb{G}_1$, which is modelled as a random oracle in our proofs of security.

Depending on the instantiation of DHP, the system parameters for IDP may include additional hash functions to be used for session-key derivation and other purposes.

We set the master private/public keys of KGC to $s \in_R \mathbb{Z}^*_q$ and $P_{Pub} = sP \in \mathbb{G}_1$, and the static private/public keys of each $U \in \mathcal{U}$ to sQ_U and $Q_U = G(ID_U)$, where ID_U is the identity of user U.

3.2. Keys mapping

Assume two protocol participants A and B of DHP whose static private/public keys are (a, g^a) and (b, g^b), respectively, as defined above. Let (x, g^y) and (y, g^y) denote the ephemeral private/public keys to be generated by A and B, respectively, during the execution of DHP. Table 1 describes the mapping of various protocol keys between DHP and IDP.

Table 1: Keys mapping: from DH-based protocol (DHP) to ID-based protocol (IDP).

Key types	DHP	IDP
Ephemeral private/public keys	$x \in \mathbb{Z}_q^*,\ g^x \in \mathbb{G}$	$x \in \mathbb{Z}_q^*,\ xP \in \mathbb{G}_1$
	$y \in \mathbb{Z}_q^*,\ g^y \in \mathbb{G}$	$y \in \mathbb{Z}_q^*,\ yP \in \mathbb{G}_1$
Static private/public keys	$a \in \mathbb{Z}_q^*,\ g^a \in \mathbb{G}$	$sQ_A \in \mathbb{G}_1, Q_A \in \mathbb{G}_1$
	$b \in \mathbb{Z}_q^*,\ g^b \in \mathbb{G}$	$sQ_B \in \mathbb{G}_1, Q_B \in \mathbb{G}_1$
Ephemeral Diffie–Hellman key	$g^{xy} \in \mathbb{G}$	$xyP \in \mathbb{G}_1$
		$\hat{e}(xP, yP)^s \in \mathbb{G}_2$
Static Diffie–Hellman key	$g^{ab} \in \mathbb{G}$	$\hat{e}(Q_A, Q_B)^s \in \mathbb{G}_2$
Static-ephemeral Diffie–Hellman keys	$g^{ay} \in \mathbb{G}$	$\hat{e}(Q_A, yP)^s \in \mathbb{G}_2$
	$g^{bx} \in \mathbb{G}$	$\hat{e}(Q_B, xP)^s \in \mathbb{G}_2$

The Diffie–Hellman keys g^{xy}, g^{ab}, g^{ay} and g^{bx} are mapped to xyP (or $\hat{e}(xP,\ yP)^s$), $\hat{e}(Q_A,\ Q_B)^s$, $\hat{e}(Q_A,\ yP)^s$ and $\hat{e}(Q_B, xP)^s$, respectively, so that no one can compute any of these keys without knowing the right private key. The ephemeral Diffie–Hellman key g^{xy} is mapped to either xyP or $\hat{e}(xP, yP)^s$, depending on how the key is used in DHP. If g^{xy} is used in a mathematically-combined form with any types of static or static-ephemeral keys, we replace it with $\hat{e}(xP, yP)^s$ in IDP (see, for example, the HMQV-ID protocol in Section 6.1). Otherwise, we replace it with xyP (see the UMP-ID protocol in Section 4.1). In practice, the static-ephemeral Diffie–Hellman key $\hat{e}(Q_A, yP)^s$ (resp. $\hat{e}(Q_B, xP)^s$) can be obtained by computing $\hat{e}(sQ_A, yP)$ or $\hat{e}(yQ_A, P_{Pub})$ (resp. $\hat{e}(sQ_B, xP)$ or $\hat{e}(xQ_B, P_{Pub})$) (see the BJM-ID protocol in Section 5.1).

3.3. Assumptions mapping

We now describe the mapping of computational assumptions between DH-based protocols and ID-based protocols. We focus on considering the CDH, DDH and GDH assumptions under which most DH-based protocols are proven secure.

CDH. Suppose that a security property ϕ of DHP was proven under the CDH assumption in $\mathbb{G}$, which we denote as CDH $\leq_\phi$ DHP.

- If the CDH-problem instance $(g^\alpha, g^\beta) \in \mathbb{G}^2$ was used in place of the ephemeral public keys (g^x, g^y) in the proof simulation for DHP (denoted as $(g^\alpha,\ g^\beta) \propto (g^x,\ g^y)$), and the ephemeral Diffie–Hellman key g^{xy} was replaced with xyP

in the keys-mapping stage (denoted as $g^{xy} \Rightarrow xyP$), then we can prove the security property ϕ of IDP under the CDH assumption in $\mathbb{G}_1$. Let $(\alpha P, \beta P) \in \mathbb{G}_1^2$ be the given instance of the CDH problem. Then, the simulator in the proof for IDP will embed the problem instance into the simulation by using it in place of the ephemeral public keys (xP, yP) (see, for example, the proof given in Section 4.4).

- Otherwise, one of the following is true:
  - $(g^\alpha, g^\beta) \propto (g^x, g^y)$ and $g^{xy} \Rightarrow \hat{e}(xP, yP)^s$
  - $(g^\alpha, g^\beta) \propto (g^a, g^b)$ and $g^{ab} \Rightarrow \hat{e}(Q_A, Q_B)^s$
  - $(g^\alpha, g^\beta) \propto (g^a, g^y)$ and $g^{ay} \Rightarrow \hat{e}(Q_A, yP)^s$, or $(g^\alpha, g^\beta) \propto (g^b, g^x)$ and $g^{bx} \Rightarrow \hat{e}(Q_B, xP)^s$

  In all of these three cases, we can prove the security property ϕ of IDP under the BDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Let $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ be the given instance of the BDH problem. Then, the simulator in the proof for IDP will embed the problem instance into the simulation by using it in place of (xP, yP, P_{Pub}), (Q_A, Q_B, P_{Pub}), or (Q_A, yP, P_{Pub}) (or (Q_B, xP, P_{Pub})), in each of the three cases, respectively. (See, for example, the proofs in Sections 4.2 and 6.2.)

DDH. Suppose that a security property ϕ of DHP is proven under the DDH assumption in $\mathbb{G}$ (i.e., DDH $\leq_\phi$ DHP). Let $(g^\alpha, g^\beta, g^\gamma) \in \mathbb{G}^3$ be the DDH-problem instance given to the simulator in the proof for DHP.

- Consider, first, the case that $(g^\alpha, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy})$ and $g^{xy} \Rightarrow xyP$. In this case, we cannot rely on the DDH assumption to prove the security property ϕ of IDP since the DDH problem in $\mathbb{G}_1$ is easy [28]. To see this, observe that, given $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1$, one can easily decide whether $\gamma = \alpha\beta \bmod q$ by testing if $\hat{e}(P, \gamma P) = \hat{e}(\alpha P, \beta P)$. One possible solution to overcome this problem is to replace the ephemeral Diffie–Hellman key g^{xy} with $\hat{e}(xP, yP)^s$ in the keys-mapping stage and then prove the security property ϕ of IDP under the DBDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ (see below for details). But, this solution comes at the price of reduced efficiency of IDP since pairing is typically much more expensive than scalar-point multiplication [5].
- Consider next the other cases:
  - $(g^\alpha, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy})$ and $g^{xy} \Rightarrow \hat{e}(xP, yP)^s$
  - $(g^\alpha, g^\beta, g^\gamma) \propto (g^a, g^b, g^{ab})$ and $g^{ab} \Rightarrow \hat{e}(Q_A, Q_B)^s$
  - $(g^\alpha, g^\beta, g^\gamma) \propto (g^a, g^y, g^{ay})$ and $g^{ay} \Rightarrow \hat{e}(Q_A, yP)^s$, or $(g^\alpha, g^\beta, g^\gamma) \propto (g^b, g^x, g^{bx})$ and $g^{bx} \Rightarrow \hat{e}(Q_B, xP)^s$

  In these cases, we can prove the security property ϕ of IDP under the DBDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Let $(\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \in \mathbb{G}_1^3 \times \mathbb{G}_2$ be the given instance of the DBDH problem. Then, the simulator in the proof for IDP will embed the problem instance into the simulation by using it in place of

– $(xP, yP, P_{Pub}, \hat{e}(xP, yP)^s)$,
– $(Q_A, Q_B, P_{Pub}, \hat{e}(Q_A, Q_B)^s)$, or
– $(Q_A, yP, P_{Pub}, \hat{e}(Q_A, yP)^s)$ (or $(Q_B, xP, P_{Pub}, \hat{e}(Q_B, xP)^s)$),

in each of the three cases, respectively.

GDH. Suppose that a security property ϕ of DHP is proven under the GDH assumption in $\mathbb{G}$ (i.e., GDH $\leq_\phi$ DHP). Let $(g^\alpha, g^\beta) \in \mathbb{G}^2$ be the GDH-problem instance given to the simulator in the proof for DHP. In the case that $(g^\alpha, g^\beta) \propto (g^x, g^y)$ and $g^{xy} \Rightarrow xyP$, we can prove the security property ϕ of IDP under the CDH assumption in $\mathbb{G}_1$. Note that a DDH oracle for $\mathbb{G}_1$ is not needed for the proof since the DDH problem in $\mathbb{G}_1$ is easy; each DDH-oracle access can be replaced by two evaluations of the bilinear map $\hat{e}$. Given the CDH-problem instance $(\alpha P, \beta P) \in \mathbb{G}_1^2$, the simulator in the proof for IDP will embed it into the simulation by using αP and βP in place of xP and yP. In the other three cases where we considered above to replace the CDH assumption with the BDH assumption, we can prove the security property ϕ of IDP under the GBDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Problem instances between GBDH and BDH are identical and thus, can be embedded in the same way. (See, for example, the proofs in Sections 4.3 and 5.2.)

Table 2 summarizes the mapping of computational assumptions we have described. According to the mapping in the table, the simulator in the proof for IDP sets the master public key P_{Pub} to be γP when embedding the BDH- or GBDH-problem instance $(\alpha P, \beta P, \gamma P)$ or the DBDH-problem instance $(\alpha P,\ \beta P,\ \gamma P,\ \hat{e}(P, P)^\delta)$. As a result of setting $P_{Pub} = \gamma P$, the simulator is unable to compute the master private key $s = \gamma$. This explains why the HMQV-ID protocol whose forward secrecy is proven under the BDH assumption cannot be proven to provide KGC forward secrecy (see Section 6.2); unlike HMQV-ID, UMP-ID can be proven to provide KGC forward secrecy since the master keys can be honestly generated when a proof is based on the CDH assumption in $\mathbb{G}_1$ (see Section 4.4). However, even when the master private key is unavailable, the simulator can still generate the static private keys of users and thus can correctly answer Corrupt queries of the adversary. Suppose a user $U \in \mathcal{U}$ whose static public key was not set to a value contained in the problem instance. For each such user U, the simulator simply chooses a random $r_U \in \mathbb{Z}_q^*$ and sets the private/public keys of U as $(sQ_U = r_u\gamma P,\ Q_U = r_u P)$. In order for this strategy to work, we require that the adversary never query the random oracle G which is used in generating static public keys of users. This restriction[3] is implicit in all our proofs except for the proof

[3]We note that Chen and Kudla [14] also made the same restriction in the security proof for their ID-based protocol.

Table 2: Assumptions mapping: from DH-based protocol to ID-based protocol.

DH-based		ID-based	
Assumptions	Embedding	Assumptions	Embedding
CDH	$(g^\alpha, g^\beta) \propto (g^x, g^y)$	CDH	$(\alpha P, \beta P) \propto (xP, yP)$
		BDH	$(\alpha P, \beta P, \gamma P) \propto (xP, yP, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^a, g^b)$	BDH	$(\alpha P, \beta P, \gamma P) \propto (Q_A, Q_B, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^a, g^y)$	BDH	$(\alpha P, \beta P, \gamma P) \propto (Q_A, yP, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^b, g^x)$		$(\alpha P, \beta P, \gamma P) \propto (Q_B, xP, P_{Pub})$
DDH	$(g^\alpha, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy})$	DDH ($\times$)	The DDH problem in $\mathbb{G}_1$ is easy
		DBDH	$(\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \propto (xP, yP, P_{Pub}, \hat{e}(xP, yP)^s)$
	$(g^\alpha, g^\beta, g^\gamma) \propto (g^a, g^b, g^{ab})$	DBDH	$(\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \propto (Q_A, Q_B, P_{Pub}, \hat{e}(Q_A, Q_B)^s)$
	$(g^\alpha, g^\beta, g^\gamma) \propto (g^a, g^y, g^{ay})$	DBDH	$(\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \propto (Q_A, yP, P_{Pub}, \hat{e}(Q_A, yP)^s)$
	$(g^\alpha, g^\beta, g^\gamma) \propto (g^b, g^x, g^{bx})$		$(\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \propto (Q_B, xP, P_{Pub}, \hat{e}(Q_B, xP)^s)$
GDH	$(g^\alpha, g^\beta) \propto (g^x, g^y)$	CDH	$(\alpha P, \beta P) \propto (xP, yP)$
		GBDH	$(\alpha P, \beta P, \gamma P) \propto (xP, yP, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^a, g^b)$	GBDH	$(\alpha P, \beta P, \gamma P) \propto (Q_A, Q_B, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^a, g^y)$	GBDH	$(\alpha P, \beta P, \gamma P) \propto (Q_A, yP, P_{Pub})$
	$(g^\alpha, g^\beta) \propto (g^b, g^x)$		$(\alpha P, \beta P, \gamma P) \propto (Q_B, xP, P_{Pub})$

in Section 4.4 whereby UMP-ID is shown to provide KGC forward secrecy under the CDH assumption.

4. UMP and its ID-Based versions

The 'unified model' protocol (UMP) is an implicitly-authenticated Diffie–Hellman protocol that has been standardized in IEEE P1363 [24], ANSI X9.63 [4] and ANSI X9.42 [3]. Let A and B be two users who wish to agree on a session key. We assume that A and B have pre-established their static private/public keys (a, g^a) and (b, g^b), respectively. UMP runs as shown in Fig. 1 where (1) g is a generator of a cyclic group $\mathbb{G}$ of prime order q and (2) H is a cryptographic hash function mapping arbitrary strings into k-bit session keys. (The required validity checking of received messages by the recipient is omitted from our discussion in this paper.) UMP – proposed originally by Ankney, Johnson and Matyas [2] – was proven SK-secure by Blake-Wilson, Johnson and Menezes [8, protocol 3] under the CDH assumption in a restricted model whereby the adversary is restricted from asking Reveal queries. Later, Jeong, Katz and Lee [26]

$$
\begin{array}{lcr}
\boxed{A} & & \boxed{B} \\
(a, g^a) & & (b, g^b) \\
\hline
x \in_R \mathbb{Z}_q^* & & y \in_R \mathbb{Z}_q^* \\
& \xrightarrow{\qquad\qquad g^x \qquad\qquad} & \\
& \xleftarrow[\qquad\qquad g^y \qquad\qquad]{} & \\
\rho_A = (g^y)^x & & \rho_B = (g^x)^y \\
\sigma_A = (g^b)^a & & \sigma_B = (g^a)^b \\
sk_A = H(\rho_A \| \sigma_A) & & sk_B = H(\rho_B \| \sigma_B) \\
& sk_A = H(g^{xy} \| g^{ab}) = sk_B &
\end{array}
$$

Figure 1: UMP: The unified model protocol.

proposed a variant of UMP, where the session key is defined as $H(A\|B\|g^x\|g^y\|g^{xy}\|g^{ab})$, and proved its forward secrecy[4] under the CDH assumption.

4.1. ID-based versions of UMP

We now derive an ID-based version of UMP by conducting the system setup and then mapping the protocol keys, as described in Sections 3.1 and 3.2. Specifically, we define the system parameters $(\mathbb{G}_1,\ \mathbb{G}_2,\ \hat{e},\ P,\ G,\ H)$, the master private/public keys $(s,\ P_{Pub} = sP)$ of KGC, and the static private/public keys $(sQ_U,\ Q_U = G(ID_U))$ of each user $U \in \mathcal{U}$. The two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ have the order q and we assume that q is implicit in $\mathbb{G}_1$ and $\mathbb{G}_2$. The hash function H has been added into the system parameters since UMP uses it as the key derivation function. We then apply the mapping of Table 1 to all kinds of keys used in UMP, replacing the ephemeral private/public keys (x, g^x) and (y, g^y) with (x, xP) and (y, yP), the static private/public keys (a, g^a) and (b, g^b) with (sQ_A, Q_A) and (sQ_B, Q_B), the ephemeral Diffie–Hellman key g^{xy} with xyP, and the static Diffie–Hellman key g^{ab} with $\hat{e}(Q_A, Q_B)^s$. The resulting protocol, UMP-ID$_0$, is outlined in Fig. 2. Since $\rho_A = xyP = \rho_B$ and $\sigma_A = \hat{e}(Q_A, Q_B)^s = \sigma_B$, A and B will compute the same session key

$$sk = H(xyP \| \hat{e}(Q_A, Q_B)^s)$$

in the presence of a passive adversary.

[4]In the UMP variant of Jeong, Katz and Lee [26], forward secrecy holds only for session keys established without active intervention by an adversary. However, as pointed out by Krawczyk [31], this limitation is not just a weakness of a particular protocol but it is inherent to any (implicitly-authenticated) two-message key exchange protocols, including all the DH-based and ID-based protocols presented in this paper.

$$
\begin{array}{lcr}
\boxed{A} & & \boxed{B} \\
(sQ_A, Q_A) & & (sQ_B, Q_B) \\
\hline
x \in_R \mathbb{Z}_q^* & & y \in_R \mathbb{Z}_q^* \\
& \xrightarrow{\quad T_A = xP \quad} & \\
& \xleftarrow[\quad T_B = yP \quad]{} & \\
\rho_A = x \cdot T_B & & \rho_B = y \cdot T_A \\
\sigma_A = \hat{e}(sQ_A, Q_B) & & \sigma_B = \hat{e}(Q_A, sQ_B) \\
sk_A = H(\rho_A \| \sigma_A) & & sk_B = H(\rho_B \| \sigma_B)
\end{array}
$$

Figure 2: UMP-ID$_0$: An ID-based version of UMP.

As mentioned above, UMP was proven SK-secure in a restricted adversary model [8]. Since CDH $\leq_{\text{SK}}$ UMP, $(g^\alpha, g^\beta) \propto (g^a, g^b)$[5] and $g^{ab} \Rightarrow \hat{e}(Q_A, Q_B)^s$, we can prove the SK security of UMP-ID$_0$ under the BDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ in the same restricted model. Our proof is provided in the next subsection and, as shown in Table 2, the simulator in the proof will embed the BDH-problem instance $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ into the simulation by using it in place of (Q_A, Q_B, P_{Pub}).

Similar to UMP, UMP-ID$_0$ is not secure if the adversary is allowed to ask a Reveal query. However, this weakness can be easily removed by modifying the session-key computation to

$$sk = H(pid \| sid \| xyP \| \hat{e}(Q_A, Q_B)^s)$$

where $pid = (A, B)$ and $sid = Q_A \| Q_B \| T_A \| T_B$. This modification ensures (with high probability) that two user instances must hold the same sets of *pid* and *sid* to compute the same session key, and thus prevents key-replication attacks such as the one presented by Blake-Wilson, Johnson and Menezes [8] against UMP. We denote this variant of UMP-ID$_0$ by UMP-ID. As claimed by Theorem 2 in Section 4.3, UMP-ID is SK-secure in the random oracle model under the GBDH assumption. Our proof of Theorem 2 is based on the result of Kudla and Paterson [32] which shows how "*protocols that are proven secure in a restricted model whereby the adversary is restricted from asking* Reveal *queries can be proven secure without imposing the restriction to the adversary by using a GAP assumption [39] in the random oracle model*".

UMP-ID also provides KGC forward secrecy (see Theorem 3 in Section 4.4). We will derive the proof of this claim from the proof of forward secrecy for the UMP

[5] As described in Section 3.3, this notation means that in the proof for UMP, the simulator embedded the CDH-problem instance (g^α, g^β) into the simulation by using it in place of the static public keys (g^a, g^b).

variant of Jeong, Katz and Lee [26], which is named $\mathcal{TS}2$. Since CDH $\leq_{\text{FS}}$ $\mathcal{TS}2$, $(g^{\alpha}, g^{\beta}) \propto (g^x, g^y)$ and $g^{xy} \Rightarrow xyP$, we can prove KGC forward secrecy of UMP-ID under the CDH assumption in $\mathbb{G}_1$. As described in Table 2, the simulator in the proof will embed the CDH-problem instance $(\alpha P, \beta P) \in \mathbb{G}_1^2$ into the simulation by using it in place of (xP, yP).

4.2. Proof of SK security for UMP-ID$_0$

Theorem 1. *In the random oracle model and under the BDH assumption, UMP-ID$_0$ is a SK-secure key exchange protocol as long as the adversary makes no* Reveal *queries.*

PROOF. Assume an adversary $\mathcal{A}$ who makes no Reveal queries and can gain a non-negligible advantage in distinguishing the test session key from random. Then we prove the theorem by constructing from $\mathcal{A}$ an algorithm $\mathcal{A}_{\text{BDH}}$ that solves the BDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with a non-negligible advantage. The objective of $\mathcal{A}_{\text{BDH}}$ is to compute and output the value $\hat{e}(P,P)^{\alpha\beta\gamma} \in \mathbb{G}_2$ when given a BDH-problem instance $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ where $\alpha, \beta, \gamma \in_R \mathbb{Z}_q^*$. Let $\Pi^*_{U,U'}$ denote $\Pi^i_{U,U'}$ for any i.

$\mathcal{A}_{\text{BDH}}$ runs $\mathcal{A}$ while simulating the oracles on its own. $\mathcal{A}_{\text{BDH}}$ embeds the BDH-problem instance $(\alpha P, \beta P, \gamma P)$ into the simulation by setting $Q_A = \alpha P$, $Q_B = \beta P$ and $P_{Pub} = \gamma P$. Here, A and B are two users chosen at random from the set of all users, in the hope that $\mathcal{A}$ will ask its Test query against $\Pi^*_{A,B}$ or $\Pi^*_{B,A}$. For each $U \in \mathcal{U} \setminus \{A, B\}$, $\mathcal{A}_{\text{BDH}}$ chooses a random $r_u \in \mathbb{Z}_q^*$ and sets their private/public keys to be $(r_u \gamma P, r_u P)$. (Recall that, as mentioned in Section 3.3, $\mathcal{A}$ will never get direct access to the random oracle G.) $\mathcal{A}_{\text{BDH}}$ outputs a random k-bit string in response to each distinct H query while storing the input-output pairs of H into a list, which we denote as HList. If $\mathcal{A}$ corrupts A or B, then $\mathcal{A}_{\text{BDH}}$ aborts. When $\mathcal{A}$ asks the Test query, $\mathcal{A}_{\text{BDH}}$ responds with a random k-bit string. For all other queries of $\mathcal{A}$, $\mathcal{A}_{\text{BDH}}$ handles them exactly in the same way as the oracles would do. At some point in time, $\mathcal{A}$ will terminate and output its guess b'. When this happens, $\mathcal{A}_{\text{BDH}}$ selects an entry of the form $(\rho \| \sigma, h)$ at random from HList, terminates and outputs σ.

Let Ask be the event that $\mathcal{A}$ makes the query $H(\rho^t_U \| \sigma^t_U)$ when Π^t_U is the test instance. Let q_H be the number of H queries made by $\mathcal{A}$. Then, the following is immediate from the simulation:

- Since Reveal queries are not allowed (meaning that key-replication attacks are not possible), $\mathcal{A}_{\text{BDH}}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability at least $1/q_H$ if Ask occurs and if the test instance is $\Pi^*_{A,B}$ or $\Pi^*_{B,A}$.

- The probability that the test instance is $\Pi^*_{A,B}$ or $\Pi^*_{B,A}$ is $1/\binom{|\mathcal{U}|}{2} = \frac{2}{|\mathcal{U}|(|\mathcal{U}|-1)}$. (We stress that this probability is independent of the number of instances but depends only on the number of users.)

Combining these observations yields that: $\mathcal{A}_{\text{BDH}}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability at least

$$\Pr[\mathsf{Ask}] \frac{1}{q_H} \frac{2}{|\mathcal{U}|(|\mathcal{U}|-1)}$$

which is non-negligible as long as $\Pr[\mathsf{Ask}]$ is non-negligible.

Now, to prove the theorem, it suffices to prove that $\Pr[\mathsf{Ask}]$ is non-negligible. Since H is a random oracle and the Reveal oracle is not available, $\mathcal{A}$ gains no advantage in distinguishing the test session key from a random key if the event Ask does not occur, as indicated by the following equation:

$$\begin{aligned}\mathsf{Adv}_{\text{UMP-ID}_0}(\mathcal{A}) &= 2 \cdot \Pr[\mathsf{Succ}] - 1 \\ &\leq 2(\Pr[\mathsf{Ask}] + \frac{1}{2}(1 - \Pr[\mathsf{Ask}])) - 1 \\ &= \Pr[\mathsf{Ask}].\end{aligned}$$

Therefore, if the advantage of $\mathcal{A}$ is non-negligible, $\Pr[\mathsf{Ask}]$ is non-negligible and so is the probability that $\mathcal{A}_{\text{BDH}}$ succeeds in solving the BDH problem. This completes the proof of Theorem 1. □

4.3. Proof of SK security for UMP-ID

Theorem 2. *In the random oracle model and under the GBDH assumption, UMP-ID is a SK-secure key exchange protocol.*

Proof. Assume that there exists an adversary $\mathcal{A}$ who can gain a non-negligible advantage in distinguishing the test session key from random. Then we can construct an algorithm $\mathcal{A}_{\text{GBDH}}$ that has a non-negligible advantage in solving the GBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. The goal of $\mathcal{A}_{\text{GBDH}}$ is to output the value $\hat{e}(P,P)^{\alpha\beta\gamma} \in \mathbb{G}_2$ when given a triple of elements $\alpha P, \beta P, \gamma P \in \mathbb{G}_1$, where $\alpha, \beta, \gamma \in_R \mathbb{Z}_q^*$, as well as an oracle $\mathcal{O}_{\text{DBDH}}(\cdot,\cdot,\cdot,\cdot)$ that solves the DBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Let $\Pi^*_{U,U'}$ be $\Pi^i_{U,U'}$ for any i. In UMP-ID, the session key is computed by applying the random oracle H to some string which we call a *key derivation string* (*kds*). Let kds^i_U denote the *kds* of instance Π^i_U. Then, $kds^i_U = pid^i_U \| sid^i_U \| \rho^i_U \| \sigma^i_U$.

$\mathcal{A}_{\text{GBDH}}$ begins by choosing two users A and B at random from $\mathcal{U}$ and setting $Q_A = \alpha P$, $Q_B = \beta P$ and $P_{Pub} = \gamma P$. For each $U \in \mathcal{U} \setminus \{A, B\}$, $\mathcal{A}_{\text{GBDH}}$ chooses a random $r_u \in \mathbb{Z}_q^*$ and sets their private/public keys to be $(r_u \gamma P, r_u P)$. $\mathcal{A}_{\text{GBDH}}$ then invokes $\mathcal{A}$ as a subroutine and handles the queries of $\mathcal{A}$ as follows:

- Send: $\mathcal{A}_{\text{GBDH}}$ answers each Send query as per the protocol specification, except that it aborts if the following event Repeat occurs.

  Repeat: The event that an ephemeral private key used by any user in response to a Send query is used again by that user (in response to a Send query).

  A straightforward calculation shows:

  $$\Pr[\mathsf{Repeat}] \leq \frac{q_{\text{send}}(q_{\text{send}} - 1)}{2|\mathbb{Z}_q^*|},$$

  where q_{send} is the number of Send queries made by $\mathcal{A}$. Note that no two unpartnered instances can compute the same *kds* unless Repeat occurs.

- $\mathsf{Corrupt}$: If $\mathcal{A}$ corrupts A or B, then $\mathcal{A}_{\text{GBDH}}$ aborts. Otherwise, $\mathcal{A}_{\text{GBDH}}$ returns the static private key of the queried user.

- H: $\mathcal{A}_{\text{GBDH}}$ uses a list, HList, to maintain input-output pairs of H. For each H query on a string m, $\mathcal{A}_{\text{GBDH}}$ first checks if HList contains an entry of the form (m, h). If it does, $\mathcal{A}_{\text{GBDH}}$ returns h. Otherwise, $\mathcal{A}_{\text{GBDH}}$ checks that $\mathcal{O}_{\text{DBDH}}(\alpha P, \beta P, \gamma P, m_{\mathbb{G}_2}) = 1$, where $m_{\mathbb{G}_2}$ is a $|\mathbb{G}_2|$-bit string that is a suffix of m. If this is true, $m_{\mathbb{G}_2} = \hat{e}(P, P)^{\alpha\beta\gamma}$ and therefore, $\mathcal{A}_{\text{GBDH}}$ will terminate and output $m_{\mathbb{G}_2}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ returns a random k-bit string str to $\mathcal{A}$ and adds (m, str) to HList.

- Reveal: Given a Reveal query on any instance Π_U^i, $\mathcal{A}_{\text{GBDH}}$ proceeds as follows:
  1. If $\Pi_U^i \neq \Pi_{A,B}^*$ and $\Pi_U^i \neq \Pi_{B,A}^*$, $\mathcal{A}_{\text{GBDH}}$ computes kds_U^i and checks if HList contains an entry of the form (kds_U^i, h). If it does, $\mathcal{A}_{\text{GBDH}}$ returns h to $\mathcal{A}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ selects a random k-bit string str, returns str to $\mathcal{A}$, and adds (kds_U^i, str) into HList.
  2. If $\Pi_U^i = \Pi_{A,B}^*$ or $\Pi_U^i = \Pi_{B,A}^*$, $\mathcal{A}_{\text{GBDH}}$ cannot compute kds_U^i as it cannot compute $\sigma_U^i = \hat{e}(P, P)^{\alpha\beta\gamma}$. In this case, $\mathcal{A}_{\text{GBDH}}$ computes ρ_U^i and checks if an entry of the form $(pid_U^i \| sid_U^i \| \rho_U^i, sk)$ is in a list called RList which $\mathcal{A}_{\text{GBDH}}$ maintains to store the revealed session keys of $\Pi_{A,B}^*$ and $\Pi_{B,A}^*$. If it is, $\mathcal{A}_{\text{GBDH}}$ returns sk to $\mathcal{A}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ selects a random k-bit string str, returns str to $\mathcal{A}$, and adds $(pid_U^i \| sid_U^i \| \rho_U^i, str)$ into the RList.

When $\mathcal{A}$ asks its Test query, $\mathcal{A}_{\text{GBDH}}$ responds with a random k-bit string. Let Ask be the event that $\mathcal{A}$ makes the query $H(pid_U^t \| sid_U^t \| \rho_U^t \| \sigma_U^t)$ when Π_U^t is the test instance. When $\mathcal{A}$ terminates and outputs its guess b' (meaning that Ask did not occur or $\mathcal{A}_{\text{GBDH}}$'s guess on the test instance was wrong), $\mathcal{A}_{\text{GBDH}}$ terminates and outputs a random $|\mathbb{G}_2|$-bit string.

From the simulation above, the following can be easily observed:

- If Ask occurs and if the test instance is $\Pi_{A,B}^*$ or $\Pi_{B,A}^*$, $\mathcal{A}_{\text{GBDH}}$ outputs the desired result $\hat{e}(P, P)^{\alpha\beta\gamma}$ with probability 1 (see the simulation of H above) unless Repeat occurs.

- The probability that $\mathcal{A}$ makes its Test query against $\Pi_{A,B}^*$ or $\Pi_{B,A}^*$ is $\frac{2}{|\mathcal{U}|(|\mathcal{U}|-1)}$.

- The probability that Repeat does not occur is at least $1 - \frac{q_{\text{send}}(q_{\text{send}}-1)}{2|\mathbb{Z}_q^*|}$, which is non-negligible.

These observations together mean that $\mathcal{A}_{\text{GBDH}}$ can output the desired result $\hat{e}(P, P)^{\alpha\beta\gamma}$ with probability at least

$$\Pr[\mathsf{Ask}] \frac{2}{|\mathcal{U}|(|\mathcal{U}| - 1)} (1 - \frac{q_{\text{send}}(q_{\text{send}} - 1)}{2|\mathbb{Z}_q^*|})$$

which is non-negligible if $\Pr[\mathsf{Ask}]$ is non-negligible.

We are now left to prove that $\Pr[\mathsf{Ask}]$ is non-negligible. Since H is a random oracle and $\Pr[\mathsf{Repeat}]$ is negligible, $\mathcal{A}$ cannot gain a non-negligible advantage in distinguishing the test session key from a random key if $\Pr[\mathsf{Ask}]$ is negligible, as shown by the following equation:

$$\begin{aligned} \mathsf{Adv}_{\text{UMP-ID}}(\mathcal{A}) &= 2 \cdot \Pr[\mathsf{Succ}] - 1 \\ &\leq 2(\Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}] + \frac{1}{2}(1 - \Pr[\mathsf{Ask}])) - 1 \\ &= 2 \cdot \Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}]. \end{aligned} \tag{1}$$

But since the advantage of $\mathcal{A}$ is non-negligible (by assumption), we obtain that $\Pr[\mathsf{Ask}]$ is non-negligible and so is the probability of $\mathcal{A}_{\text{GBDH}}$ solving the GBDH problem. This completes the proof of Theorem 2. □

4.4. Proof of KGC-FS for UMP-ID

Definition 4. For a $\mathsf{Send}(U, i, M)$ query, we say that the Send query was *passively-generated* if the message M was output by a previous Send query.

Definition 5 (FS-freshness). We say that an instance is *FS-fresh* if (1) the Send queries directed to the instance are passively-generated ones and (2) the adversary has not issued a Reveal query against the instance and its partner instance.

Definition 6 (KGC forward secrecy). A key exchange protocol provides *KGC forward secrecy (KGC-FS)* if, for any FS-fresh instance and for any PPT adversary $\mathcal{A}$ with access to the master private key of KGC, the advantage of $\mathcal{A}$ in distinguishing the session key from random is negligible.

Theorem 3. *In the random oracle model and under the CDH assumption, UMP-ID provides KGC-FS.*

Proof. Assume an adversary $\mathcal{A}$ who breaks, with a non-negligible advantage, the KGC-FS property of UMP-ID. Given the adversary $\mathcal{A}$, we prove the theorem by constructing an algorithm $\mathcal{A}_{\text{CDH}}$ that solves, with a non-negligible advantage, the CDH problem in $\mathbb{G}_1$. The goal of $\mathcal{A}_{\text{CDH}}$ is to compute and output the value $\alpha\beta P \in \mathbb{G}_1$ when given a CDH-problem instance $(\alpha P, \beta P) \in \mathbb{G}_1^2$.

$\mathcal{A}_{\text{CDH}}$ starts by faithfully generating all the static private/public keys. Since we are now considering KGC-FS, the adversary $\mathcal{A}$ is given the master private key of KGC but is required to test only a FS-fresh instance ($\mathcal{A}$ can trivially compute the static private keys of all users since it has been given KGC's master secret). Let n be the maximum number of instances that $\mathcal{A}$ may activate. $\mathcal{A}_{\text{CDH}}$ chooses two instances at random from all the n instances, and uses αP and βP as the outgoing messages of the two instances. For all other instances, $\mathcal{A}_{\text{CDH}}$ honestly interacts with $\mathcal{A}$ except that it aborts if Repeat occurs, where Repeat is as defined in the proof of Theorem 2. $\mathcal{A}_{\mathrm{CDH}}$ outputs a random k-bit string in response to each distinct H query while storing the input-output pairs of H into a list called HList. When $\mathcal{A}$ asks its Test query, $\mathcal{A}_{\mathrm{CDH}}$ simply outputs a random k-bit string. For all other queries of $\mathcal{A}$, $\mathcal{A}_{\mathrm{CDH}}$ handles them in the straightforward way. When $\mathcal{A}$ terminates and outputs its guess b', $\mathcal{A}_{\mathrm{CDH}}$ selects an entry of the form $(pid\|sid\|\rho\|\sigma, h)$ at random from HList, terminates and outputs ρ.

Let kds_U^t be the key derivation string of the test instance (see the proof of Theorem 2 for the definition of a key derivation string), and Ask be the event that $\mathcal{A}$ makes the query $H(kds_U^t)$. The statement of the theorem easily follows if we make the following observations:

- If Ask occurs and if $\mathcal{A}_{\mathrm{CDH}}$'s guess on the test instance is correct, $\mathcal{A}_{\mathrm{CDH}}$ outputs the desired result $\alpha\beta P$ with probability at least $1/q_H$ unless Repeat occurs. Here, q_H denotes the number of H queries made by $\mathcal{A}$.

- The probability that $\mathcal{A}_{\mathrm{CDH}}$'s guess on the test instance turns out to be correct (i.e., the probability that the session key of the test instance is computed using both αP and βP) is $\frac{2}{n(n-1)}(= 1/\binom{n}{2})$.

- The probability that Repeat does not occur is at least $1-\frac{q_{\mathrm{send}}(q_{\mathrm{send}}-1)}{2|\mathbb{Z}_q^*|}$, where q_{send} is the number of Send queries made by $\mathcal{A}$.

Therefore, the probability that $\mathcal{A}_{\mathrm{CDH}}$ outputs $\alpha\beta P$ is at least

$$\Pr[\mathsf{Ask}]\frac{1}{q_H}\frac{2}{n(n-1)}(1-\frac{q_{\mathrm{send}}(q_{\mathrm{send}}-1)}{2|\mathbb{Z}_q^*|})$$

which is non-negligible as long as $\Pr[\mathsf{Ask}]$ is non-negligible.

It is not hard to see that $\Pr[\mathsf{Ask}]$ is non-negligible. Since H is a random oracle and $\Pr[\mathsf{Repeat}]$ is negligible, $\mathcal{A}$ cannot gain a non-negligible advantage in distinguishing the test session key from a random key if $\Pr[\mathsf{Ask}]$ is negligible (see Eq. (1) in the proof of Theorem 2). Therefore, $\Pr[\mathsf{Ask}]$ is non-negligible and so is the probability of $\mathcal{A}_{\mathrm{CDH}}$ solving the CDH problem. This completes the proof of Theorem 3. □

As the static private keys of all users can be computed from the master private key of KGC, Theorem 3 implies that: *UMP-ID provides forward secrecy.*

5. BJM protocol and its ID-based version

The BJM protocol was presented in 1997 by Blake-Wilson, Johnson and Menezes [8, protocol 4], and runs as shown in Fig. 3 where (1) g is a generator of a cyclic group $\mathbb{G}$ of prime order q, (2) (a, g^a) and (b, g^b) are pairs of static private/public keys of A and B respectively, and (3) H is a cryptographic hash function mapping arbitrary strings to

$\boxed{A}$		$\boxed{B}$
(a, g^a)		(b, g^b)
$x \in_R \mathbb{Z}_q^*$		$y \in_R \mathbb{Z}_q^*$
	$\xrightarrow{\quad g^x \quad}$	
	$\xleftarrow[\quad g^y \quad]{}$	
$\rho_A = (g^y)^a$		$\rho_B = (g^a)^y$
$\sigma_A = (g^b)^x$		$\sigma_B = (g^x)^b$
$sk_A = H(\rho_A \| \sigma_A)$		$sk_B = H(\rho_B \| \sigma_B)$

$$sk_A = H(g^{ay} \| g^{bx}) = sk_B$$

Figure 3: BJM: Protocol 4 of Blake-Wilson, Johnson and Menezes [8].

k-bit session keys. BJM is similar to earlier protocols, like MTI/A0 [35], Goss [23] and KEA [38], in the sense that the session key is defined as a function of two shared keys g^{ay} and g^{bx} established through the protocol execution. Kudla and Paterson [32] proposed a variant of the BJM protocol, where the session key is defined as $H(g^{bx} \| g^{ay} \| sid)$ with $sid = g^a \| g^b \| g^x \| g^y$, and proved its security under the GDH assumption in a model that captures the notion of key compromise impersonation resistance.

5.1. ID-based version of BJM

We now construct an ID-based protocol from BJM. We first conduct the system setup as described in Section 3.1 to define the system parameters ($\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e}$, P, G, H), the master private/public keys (s, $P_{Pub} = sP$) of KGC, and the static private/public keys (sQ_U, $Q_U = G(ID_U)$) of each $U \in \mathcal{U}$. The two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ have order q, and the hash function H is as defined in BJM. We then apply the mapping of Table 1 to the protocol keys of BJM. The shared secrets g^{ay} and g^{bx} are keys of type "Static-ephemeral Diffie–Hellman keys", and thus are replaced with $\hat{e}(Q_A, yP)^s$ and $\hat{e}(Q_B, xP)^s$, respectively. Similar to the BJM variant of Kudla and Paterson [32] and as in the construction of UMP-ID, we include both *pid* and *sid* as part of the input to the key derivation function H so that different sets of *pid* and *sid* lead to different session keys (with an overwhelming probability). The resulting ID-based protocol is depicted in Fig. 4, and we denote it by BJM-ID. The correctness of BJM-ID can be

$$
\begin{array}{lcr}
\boxed{A} & & \boxed{B} \\
(sQ_A, Q_A) & & (sQ_B, Q_B) \\
 & pid_A = pid_B = (A, B) & \\
\hline
x \in_R \mathbb{Z}_q^* & & y \in_R \mathbb{Z}_q^* \\
 & \xrightarrow{\quad T_A = xP \quad} & \\
 & \xleftarrow[\quad T_B = yP \quad]{} & \\
sid_A = Q_A \| Q_B \| T_A \| T_B & & sid_B = Q_A \| Q_B \| T_A \| T_B \\
\rho_A = \hat{e}(sQ_A, T_B) & & \rho_B = \hat{e}(yQ_A, P_{Pub}) \\
\sigma_A = \hat{e}(xQ_B, P_{Pub}) & & \sigma_B = \hat{e}(sQ_B, T_A) \\
sk_A = H(pid_A \| sid_A \| \rho_A \| \sigma_A) & & sk_B = H(pid_B \| sid_B \| \rho_B \| \sigma_B)
\end{array}
$$

Figure 4: BJM-ID: An ID-based version of protocol BJM.

easily verified from:

$$
\begin{aligned}
\rho_A &= \hat{e}(sQ_A, T_B) \\
&= \hat{e}(sQ_A, yP) \\
&= \hat{e}(Q_A, P)^{ys} \\
&= \hat{e}(yQ_A, sP) \\
&= \hat{e}(yQ_A, P_{Pub}) = \rho_B,
\end{aligned}
$$

$$
\begin{aligned}
\sigma_A &= \hat{e}(xQ_B, P_{Pub}) \\
&= \hat{e}(xQ_B, sP) \\
&= \hat{e}(Q_B, P)^{xs} \\
&= \hat{e}(sQ_B, xP) \\
&= \hat{e}(sQ_B, T_A) = \sigma_B.
\end{aligned}
$$

BJM-ID is a SK-secure protocol that provides key compromise impersonation resistance (KCIR). We will derive the proof for BJM-ID from the proof that the BJM variant of Kudla and Paterson [32], KP-BJM, is a SK-secure protocol providing KCIR. Since GDH $\leq_{\text{SK,KCIR}}$ KP-BJM, $(g^\alpha, g^\beta) \propto (g^b, g^x)$ and $g^{bx} \Rightarrow \hat{e}(Q_B, xP)^s$, we can prove the security of BJM-ID under the GBDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. As described in Table 2, the simulator in the proof for BJM-ID will embed the GBDH-problem instance $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ into the simulation by using it in place of (Q_B, xP, P_{Pub}).

5.2. Proof of SK security with KCIR for BJM-ID

Definition 7 (KCIR-freshness). An instance $\Pi^i_{U,U'}$ is *KCIR-fresh* if none of the following occurs:

1. The adversary queries $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$, where $\Pi^j_{U'}$ is partnered with Π^i_U.
2. The adversary queries $\mathsf{Corrupt}(U')$.

This definition considers an instance $\Pi^i_{U,U'}$ as fresh even after the adversary obtained the static private key of U, and thus captures the notion of KCIR as well as the SK-security notion.

Definition 8 (SK security with KCIR). A key exchange protocol is *SK-secure with KCIR* if, for any KCIR-fresh instance and for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing the session key from random is negligible.

Theorem 4. *In the random oracle model and under the GBDH assumption, BJM-ID is SK-secure with KCIR.*

Proof. Assuming an adversary $\mathcal{A}$ who can distinguish the test session key from random with a non-negligible advantage, we build an algorithm $\mathcal{A}_{\text{GBDH}}$ that solves the GBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with a non-negligible advantage. Let $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ be an instance of the GBDH problem given to $\mathcal{A}_{\text{GBDH}}$. Then, the goal of $\mathcal{A}_{\text{GBDH}}$ is to compute and output the value $\hat{e}(P, P)^{\alpha\beta\gamma} \in \mathbb{G}_2$ when given access to an oracle $\mathcal{O}_{\text{DBDH}}(\cdot, \cdot, \cdot, \cdot)$ that solves the DBDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$.

$\mathcal{A}_{\text{GBDH}}$ first sets the master public key of KGC to be γP (i.e., $P_{Pub} = \gamma P$). $\mathcal{A}_{\text{GBDH}}$ then chooses a random user $B \in \mathcal{U}$ and sets its public key to be αP. For each other user, $\mathcal{A}_{\text{GBDH}}$ selects a random $r \in \mathbb{Z}_q^*$ and sets the private/public keys to be $(r\gamma P, rP)$. Let n be the maximum number of instances that $\mathcal{A}$ may activate. $\mathcal{A}_{\text{GBDH}}$ chooses an instance Π^t_A, where $A \neq B$, at random from all the n instances, in the hope that $\mathcal{A}$ will choose $\Pi^t_{A,B}$ as the test instance.

Now, $\mathcal{A}_{\text{GBDH}}$ invokes $\mathcal{A}$ as a subroutine and answers the oracle queries on its own. $\mathcal{A}_{\text{GBDH}}$ answers all the Send queries of $\mathcal{A}$ as per the protocol specification, except that $\mathcal{A}_{\text{GBDH}}$ uses βP as the ephemeral public key of Π^t_A and aborts if the event Repeat occurs (see the proof of Theorem 2 for details on the event Repeat). When $\mathcal{A}$ asks a $\mathsf{Corrupt}(U)$ query, $\mathcal{A}_{\text{GBDH}}$ aborts if $U = B$ and otherwise, returns the static private key of U.

As defined in the proof of Theorem 2, we let kds be a *key derivation string* from which a session key is computed by applying the random oracle H, and $kds^i_U = pid^i_U \| sid^i_U \| \rho^i_U \| \sigma^i_U$ denote the kds of instance Π^i_U. As is clear from the above simulation, $\mathcal{A}_{\text{GBDH}}$ cannot compute kds^s_B for any s if the ephemeral public key received by Π^s_B has been generated by $\mathcal{A}$. But, given a string m, $\mathcal{A}_{\text{GBDH}}$ can determine whether $m = kds^s_B$ or not, as shown below:

Deciding $m \stackrel{?}{=} kds_B^s$

Let $x'P$ be the ephemeral public key received by Π_B^s. Given a string m, $\mathcal{A}_{\text{GBDH}}$ first checks if the (bit) length of m is equal to the length of key derivation strings. If it is, then $\mathcal{A}_{\text{GBDH}}$ checks that (1) $pid_B^s \| sid_B^s \| \rho_B^s$ is a prefix of m and (2) $\mathcal{O}_{\text{DBDH}}(\alpha P, x'P, \gamma P, m_{\mathbb{G}_2}) = 1$ where $m_{\mathbb{G}_2}$ is a $|\mathbb{G}_2|$-bit string that is a suffix of m. If both are true, then $m = kds_B^s$.

By using this deciding operation, $\mathcal{A}_{\text{GBDH}}$ can maintain consistency between answers to H queries and Reveal queries. $\mathcal{A}_{\text{GBDH}}$ simulates the random oracle H and the Reveal oracle as follows:

- H: $\mathcal{A}_{\text{GBDH}}$ uses a list, HList, to maintain input-output pairs of H. For each H query on a string m, $\mathcal{A}_{\text{GBDH}}$ first checks if an entry of the form (m, h) is in HList. If it is, then $\mathcal{A}_{\text{GBDH}}$ returns h to $\mathcal{A}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ checks that $\mathcal{O}_{\text{DBDH}}(\alpha P, \beta P, \gamma P, m_{\mathbb{G}_2}) = 1$, where $m_{\mathbb{G}_2}$ is a $|\mathbb{G}_2|$-bit string that is a suffix of m. If this is true, $m_{\mathbb{G}_2} = \hat{e}(P,P)^{\alpha\beta\gamma}$ and therefore, $\mathcal{A}_{\text{GBDH}}$ will terminate and output $m_{\mathbb{G}_2}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ checks if $m = kds_B^s$ for some s. Given the string m, this check can be done by performing the above deciding operation for each tuple $(pid_B^s \| sid_B^s \| \rho_B^s, sk)$ in the RList which is maintained by $\mathcal{A}_{\text{GBDH}}$ to store revealed session keys of instances of B. If such a tuple $(pid_B^s \| sid_B^s \| \rho_B^s, sk)$ exists in RList, $\mathcal{A}_{\text{GBDH}}$ returns sk to $\mathcal{A}$ and adds (m, sk) into HList. Otherwise, $\mathcal{A}_{\text{GBDH}}$ returns a random k-bit string str to $\mathcal{A}$ and adds (m, str) into HList.
- Reveal: Given a Reveal query on any instance Π_U^i, $\mathcal{A}_{\text{GBDH}}$ proceeds as follows:
  1. If $\Pi_U^i = \Pi_A^t$ or $\Pi_U^i = \Pi_{U'}^j$, where $\Pi_{U'}^j$ is partnered with Π_A^t, $\mathcal{A}_{\text{GBDH}}$ aborts.
  2. If $\Pi_U^i = \Pi_B^s$ for any s, $\mathcal{A}_{\text{GBDH}}$ computes ρ_B^s and checks if a tuple of the form $(pid_B^s \| sid_B^s \| \rho_B^s, sk)$ is in the RList. If it is, $\mathcal{A}_{\text{GBDH}}$ returns sk in response to the query. Otherwise, $\mathcal{A}_{\text{GBDH}}$ checks if HList contains an entry (m, h) such that $m = kds_B^s$. Given $pid_B^s \| sid_B^s \| \rho_B^s$, this check can be done by performing the deciding operation for all entries in HList. If such entry (m, h) exists in HList, $\mathcal{A}_{\text{GBDH}}$ returns h to $\mathcal{A}$ and adds $(pid_B^s \| sid_B^s \| \rho_B^s, h)$ into RList. Otherwise, $\mathcal{A}_{\text{GBDH}}$ returns a random k-bit string str to $\mathcal{A}$, and adds $(pid_B^s \| sid_B^s \| \rho_B^s, str)$ into RList.
  3. Otherwise, $\mathcal{A}_{\text{GBDH}}$ computes kds_U^i and checks if an entry of the form (kds_U^i, h) is in HList. If it is, $\mathcal{A}_{\text{GBDH}}$ returns h to $\mathcal{A}$. Otherwise, $\mathcal{A}_{\text{GBDH}}$ returns a random k-bit string str to $\mathcal{A}$, and adds (kds_U^i, str) into HList.

When $\mathcal{A}$ asks its Test query, $\mathcal{A}_{\text{GBDH}}$ responds with a random k-bit string. Let Ask be the event that $\mathcal{A}$ makes an H query on kds of the test instance. When $\mathcal{A}$ terminates and outputs its guess b' (meaning that Ask did not occur or $\mathcal{A}_{\text{GBDH}}$'s guess on the test instance was wrong), $\mathcal{A}_{\text{GBDH}}$ terminates and outputs a random $|\mathbb{G}_2|$-bit string.

The probability that $\mathcal{A}_{\text{GBDH}}$ succeeds in solving the GBDH problem can be easily obtained by noticing the following:

- If Ask occurs and if the test instance is $\Pi^t_{A,B}$, $\mathcal{A}_{\text{GBDH}}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability 1 (see the simulation of H above) unless Repeat occurs.
- The probability that $\mathcal{A}$ makes its Test query against $\Pi^t_{A,B}$ is at least $\frac{1}{n|\mathcal{U}|}$.
- The probability that Repeat does not occur is at least $1 - \frac{q_{\text{send}}(q_{\text{send}}-1)}{2|\mathbb{Z}^*_q|}$.

Combining these observations immediately yields that $\mathcal{A}_{\text{GBDH}}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability at least

$$\Pr[\mathsf{Ask}]\frac{1}{n|\mathcal{U}|}(1 - \frac{q_{\text{send}}(q_{\text{send}} - 1)}{2|\mathbb{Z}^*_q|})$$

which is non-negligible as long as $\Pr[\mathsf{Ask}]$ is non-negligible.

To prove the theorem, we are now left to prove that $\Pr[\mathsf{Ask}]$ is non-negligible. Since H is a random oracle and $\Pr[\mathsf{Repeat}]$ is negligible, $\mathcal{A}$ cannot gain a non-negligible advantage in distinguishing the test session key from a random key if $\Pr[\mathsf{Ask}]$ is negligible, as indicated by the following equation:

$$\begin{aligned}\mathsf{Adv}_{\text{BJM-ID}}(\mathcal{A}) &= 2 \cdot \Pr[\mathsf{Succ}] - 1 \\ &\leq 2(\Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}] + \frac{1}{2}(1 - \Pr[\mathsf{Ask}])) - 1 \\ &= 2 \cdot \Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}].\end{aligned}$$

Since the advantage of $\mathcal{A}$ is non-negligible (by assumption), we obtain that $\Pr[\mathsf{Ask}]$ is non-negligible and so is the probability of $\mathcal{A}_{\text{GBDH}}$ solving the GBDH problem. This concludes the proof of Theorem 4. $\square$

6. HMQV protocol and its ID-based version

The HMQV protocol of Krawczyk [31] is among the most efficient of all known public-key authenticated Diffie–Hellman protocols. As shown in Fig. 5, HMQV uses two cryptographic hash functions $\bar{H} : \{0,1\}^* \rightarrow \{0,1\}^{|q|/2}$ and $H : \{0,1\}^* \rightarrow \{0,1\}^k$, where q is the prime order of the underlying group $\mathbb{G}$ and k is the bit length of the session key. It can be easily verified that at the end of HMQV, A and B compute the same session key $H(g^{xy+ady+bex+abde})$. HMQV was proven secure under the CDH assumption in the CK model [13] where the adversary is allowed access to ephemeral private keys of users.

$$
\begin{array}{lcr}
\boxed{A} & & \boxed{B} \\
(a, g^a) & & (b, g^b) \\
\hline
x \in_R \mathbb{Z}_q^* & & y \in_R \mathbb{Z}_q^* \\
 & \xrightarrow{\qquad\qquad g^x \qquad\qquad} & \\
 & \xleftarrow[\qquad\qquad g^y \qquad\qquad]{} & \\
 & d = \bar{H}(g^x, B),\ e = \bar{H}(g^y, A) & \\
\sigma_A = (g^y (g^b)^e)^{x+da} & & \sigma_B = (g^x (g^a)^d)^{y+eb} \\
sk_A = H(\sigma_A) & & sk_B = H(\sigma_B)
\end{array}
$$

Figure 5: The HMQV protocol.

6.1. ID-based version of HMQV

We transform HMQV into an ID-based protocol, starting with the system setup followed by the mapping of keys. The system parameters include two hash functions $\bar{H} : \{0,1\}^* \to \{0,1\}^{|q|/2}$ and $H : \{0,1\}^* \to \{0,1\}^k$, in addition to the basic parameters $(\mathbb{G}_1,\ \mathbb{G}_2,\ \hat{e},\ P,\ G)$. KGC's master keys $(s,\ P_{Pub})$ and user U's static keys $(sQ_U,\ Q_U)$ are defined as in Section 3.1. Then, the protocol keys are mapped as shown in Table 1. For the key derivation secrets σ_A and σ_B, we change their computations to $\sigma_A = \hat{e}(T_B + eQ_B, xP_{Pub} + dsQ_A)$ and $\sigma_B = \hat{e}(T_A + dQ_A, yP_{Pub} + esQ_B)$ by first replacing their factors g^{xy}, g^{ady}, g^{bex} and g^{abde} with $\hat{e}(xP, yP)^s$, $\hat{e}(Q_A, yP)^{sd}$, $\hat{e}(Q_B, xP)^{se}$ and $\hat{e}(Q_A, Q_B)^{sde}$, respectively, and then making some mathematical derivations, as shown below:

$$
\begin{aligned}
\sigma_A &= \hat{e}(xP, yP)^s \cdot \hat{e}(Q_A, yP)^{sd} \cdot \hat{e}(Q_B, xP)^{se} \cdot \hat{e}(Q_A, Q_B)^{sde} \\
&= \hat{e}(xyP, sP) \cdot \hat{e}(sQ_A, dyP) \cdot \hat{e}(xeQ_B, sP) \cdot \hat{e}(sQ_A, deQ_B) \\
&= \hat{e}(xT_B, P_{Pub}) \cdot \hat{e}(sQ_A, dT_B) \cdot \hat{e}(xeQ_B, P_{Pub}) \cdot \hat{e}(sQ_A, deQ_B) \\
&= \hat{e}(xT_B, P_{Pub}) \cdot \hat{e}(xeQ_B, P_{Pub}) \cdot \hat{e}(sQ_A, dT_B) \cdot \hat{e}(sQ_A, deQ_B) \\
&= \hat{e}(xT_B + xeQ_B, P_{Pub}) \cdot \hat{e}(sQ_A, dT_B + deQ_B) \\
&= \hat{e}(T_B + eQ_B, P_{Pub})^x \cdot \hat{e}(sQ_A, T_B + eQ_B)^d \\
&= \hat{e}(T_B + eQ_B, xP_{Pub}) \cdot \hat{e}(dsQ_A, T_B + eQ_B) \\
&= \hat{e}(T_B + eQ_B, xP_{Pub}) \cdot \hat{e}(T_B + eQ_B, dsQ_A) \\
&= \hat{e}(T_B + eQ_B, xP_{Pub} + dsQ_A),
\end{aligned}
$$

$$
\begin{array}{lcr}
\boxed{A} & & \boxed{B} \\
(sQ_A, Q_A) & & (sQ_B, Q_B) \\
& pid_A = pid_B = (A, B) & \\
\hline
x \in_R \mathbb{Z}_q^* & & y \in_R \mathbb{Z}_q^* \\
& \xrightarrow{\quad T_A = xP \quad} & \\
& \xleftarrow[\quad T_B = yP \quad]{} & \\
& d = \bar{H}(T_A, B),\ e = \bar{H}(T_B, A) & \\
sid_A = Q_A \| Q_B \| T_A \| T_B & & sid_B = Q_A \| Q_B \| T_A \| T_B \\
\sigma_A = \hat{e}(T_B + eQ_B, xP_{Pub} + dsQ_A) & & \sigma_B = \hat{e}(T_A + dQ_A, yP_{Pub} + esQ_B) \\
sk_A = H(pid_A \| sid_A \| \sigma_A) & & sk_B = H(pid_B \| sid_B \| \sigma_B)
\end{array}
$$

Figure 6: HMQV-ID: An ID-based version of protocol HMQV.

$$
\begin{aligned}
\sigma_B &= \hat{e}(xP, yP)^s \cdot \hat{e}(Q_A, yP)^{sd} \cdot \hat{e}(Q_B, xP)^{se} \cdot \hat{e}(Q_A, Q_B)^{sde} \\
&= \hat{e}(xyP, sP) \cdot \hat{e}(dyQ_A, sP) \cdot \hat{e}(sQ_B, exP) \cdot \hat{e}(deQ_A, sQ_B) \\
&= \hat{e}(yT_A, P_{Pub}) \cdot \hat{e}(dyQ_A, P_{Pub}) \cdot \hat{e}(sQ_B, eT_A) \cdot \hat{e}(deQ_A, sQ_B) \\
&= \hat{e}(yT_A, P_{Pub}) \cdot \hat{e}(dyQ_A, P_{Pub}) \cdot \hat{e}(eT_A, sQ_B) \cdot \hat{e}(deQ_A, sQ_B) \\
&= \hat{e}(yT_A + dyQ_A, P_{Pub}) \cdot \hat{e}(eT_A + deQ_A, sQ_B) \\
&= \hat{e}(T_A + dQ_A, P_{Pub})^y \cdot \hat{e}(T_A + dQ_A, sQ_B)^e \\
&= \hat{e}(T_A + dQ_A, yP_{Pub}) \cdot \hat{e}(T_A + dQ_A, esQ_B) \\
&= \hat{e}(T_A + dQ_A, yP_{Pub} + esQ_B) = \sigma_A.
\end{aligned}
$$

Note above that the factor g^{xy} of σ_A and σ_B is replaced with $\hat{e}(xP, yP)^s$ instead of xyP since it is used in a multiplicatively-combined form with the static Diffie–Hellman key g^{abde} (as well as the static-ephemeral Diffie–Hellman keys g^{ady} and g^{bex}). The resulting ID-based protocol, HMQV-ID, is shown in Fig. 6. As in the cases of UMP-ID and BJM-ID, the inclusion of *pid* and *sid* into the session-key computation will significantly simplify our proof for HMQV-ID by minimizing the feasibility of key-replication attacks.

In HMQV-ID, the computational cost for each user involves only two half scalar-point multiplications (i.e., approximately one scalar-point multiplication) and one pairing online, and two scalar-point multiplications offline (i.e., xP_{Pub} and xP by A; yP_{Pub} and yP by B). This compares favourably to other published two-party, two-message ID-based protocols as shown in Section 7. We also note that HMQV-ID is similar to the ID-MQV protocol derived by Wang [48], except in the construction of the session key for HMQV-ID where $Q_A||Q_B$ are also part of the keying material[6]; and in our computations of both d and e in HMQV-ID, the partner's user ID (B and A respectively) are used instead of the partner's public key (Q_B and Q_A in the computation of h_A and h_B in ID-MQV).

HMQV was proven to provide SK security, forward secrecy, KCIR and, in addition, resilience to the leakage of ephemeral private keys [31]. For HMQV-ID, we only provide a proof of forward secrecy and proofs of other properties are deferred as future work. Since CDH $\leq_{\text{FS}}$ HMQV, $(g^\alpha, g^\beta) \propto (g^x, g^y)$ and $g^{xy} \Rightarrow \hat{e}(xP, yP)^s$, we can prove forward secrecy of HMQV-ID under the BDH assumption on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. As shown in Table 2, the simulator in the proof for HMQV-ID will embed the BDH-problem instance $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ into the simulation by using it in place of (xP, yP, P_{Pub}). This way of embedding the problem instance makes the master secret key unavailable to the simulator, which explains why, in contrast to UMP-ID, HMQV-ID is proven to provide forward secrecy instead of KGC forward secrecy.

6.2. Proof of forward secrecy for HMQV-ID

Since we prove forward secrecy of HMQV-ID in the eCK model, the adversary in the proof can make EphemeralKeyReveal queries in addition to other queries defined in the BR model (as outlined in Section 2.3).

Definition 9 (FS-freshness in eCK). An instance Π_U^i is *FS-fresh in eCK* if all the following hold:

1. The Send queries asked against Π_U^i are passively-generated ones.
2. The adversary has not issued a Reveal query against Π_U^i and its partner instance.
3. The adversary has not issued an EphemeralKeyReveal query against Π_U^i and its partner instance.

We refer to Definition 4 for the notion of a passively-generated Send query.

Definition 10 (Forward secrecy in eCK). A key exchange protocol provides *forward secrecy in eCK* if, for any instance who is FS-fresh in eCK, the advantage of any PPT adversary $\mathcal{A}$ in distinguishing the session key from random is negligible.

Theorem 5. *In the random oracle model and under the BDH assumption, HMQV-ID provides forward secrecy in eCK.*

[6]Informally as shown by Choo, Boyd and Hitchcock [19], including unique session identifiers – note that $Q_A||Q_B$ is part of the unique session identifier in HMQV-ID – in the construction of the session key of HMQV-ID ensures that session keys will be fresh and since the unique session identifiers in HMQV-ID are defined as the concatenation of messages exchanged during the protocol execution, messages altered during the transmission will result in different session keys.

PROOF. Assume an adversary $\mathcal{A}$ who has a non-negligible advantage in breaking forward secrecy of HMQV-ID. Given the adversary $\mathcal{A}$, we prove the theorem by constructing an algorithm $\mathcal{A}_{\rm BDH}$ that has a non-negligible advantage in solving the BDH problem on $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. The goal of $\mathcal{A}_{\rm BDH}$ is to compute and output the value $\hat{e}(P,P)^{\alpha\beta\gamma} \in \mathbb{G}_2$ when given a BDH-problem instance $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_1^3$ where $\alpha, \beta, \gamma \in_R \mathbb{Z}_q^*$.

Since we are now considering forward secrecy, the adversary $\mathcal{A}$ is given the static private keys of all users (but not the master secret of KGC) and is required to test only an instance who is FS-fresh in eCK. $\mathcal{A}_{\rm BDH}$ begins by setting $P_{Pub} = \gamma P$ and for each $U \in \mathcal{U}$, defining their private/public keys to be $(r_u \gamma P, r_u P)$ where $r_u \in_R \mathbb{Z}_q^*$. $\mathcal{A}_{\rm BDH}$ then runs $\mathcal{A}$ while simulating the oracles on its own. Let n be the maximum number of instances that $\mathcal{A}$ may activate. $\mathcal{A}_{\rm BDH}$ chooses two instances Π_U^i and $\Pi_{U'}^j$ at random from all the n instances. $\mathcal{A}_{\rm BDH}$ answers Send queries of $\mathcal{A}$ as per the protocol specification, except that: (1) it uses αP and βP as the outgoing messages of Π_U^i and $\Pi_{U'}^j$ and (2) it aborts if the event Repeat occurs (see the proof of Theorem 2 for the definition of Repeat). If $\mathcal{A}$ asks an $\mathsf{EphemeralKeyReveal}$ query against Π_U^i or $\Pi_{U'}^j$, then $\mathcal{A}_{\rm BDH}$ aborts. For all other $\mathsf{EphemeralKeyReveal}$ queries, $\mathcal{A}_{\rm BDH}$ answers them in the obvious way. $\mathcal{A}_{\rm BDH}$ outputs a random k-bit string in response to each distinct H query while storing the input-output pairs of H into a list called HList. When $\mathcal{A}$ asks its Test query, $\mathcal{A}_{\rm BDH}$ outputs a random k-bit string. For all other queries of $\mathcal{A}$, $\mathcal{A}_{\rm BDH}$ handles them in the straightforward way. When $\mathcal{A}$ terminates and outputs its guess b', $\mathcal{A}_{\rm BDH}$ selects an entry of the form $(pid \| sid \| \sigma, h)$ at random from HList, computes

$$\Delta = \frac{\sigma}{\hat{e}(sQ_A, \beta P)^d \cdot \hat{e}(sQ_B, \alpha P)^e \cdot \hat{e}(sQ_A, Q_B)^{de}},$$

where $d = \bar{H}(\alpha P, B)$ and $e = \bar{H}(\beta P, A)$, and terminates with the output Δ.

Let $q_{\rm send}$ and q_H be the numbers of Send queries and H queries, respectively, made by $\mathcal{A}$. Let Ask be the event that $\mathcal{A}$ makes an H query on kds of the test instance (see the proof of Theorem 2 for the definition of kds). Then, the probability that $\mathcal{A}_{\rm BDH}$ succeeds in solving the BDH problem can be easily calculated by observing the following:

- If Ask occurs and if $\mathcal{A}_{\rm BDH}$'s guess on the test instance is correct, $\mathcal{A}_{\rm BDH}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability at least $1/q_H$ unless Repeat occurs.

- The probability that $\mathcal{A}_{\rm BDH}$ correctly guesses on the test instance is $\frac{2}{n(n-1)}$.

- The probability that Repeat does not occur is at least $1 - \frac{q_{\rm send}(q_{\rm send}-1)}{2|\mathbb{Z}_q^*|}$.

By combining these observations, we immediately obtain that $\mathcal{A}_{\rm BDH}$ outputs the desired result $\hat{e}(P,P)^{\alpha\beta\gamma}$ with probability at least

$$\Pr[\mathsf{Ask}] \frac{1}{q_H} \frac{2}{n(n-1)} (1 - \frac{q_{\rm send}(q_{\rm send}-1)}{2|\mathbb{Z}_q^*|})$$

which is non-negligible if $\Pr[\mathsf{Ask}]$ is non-negligible.

We finally show that $\Pr[\mathsf{Ask}]$ is non-negligible. Since H is a random oracle and $\Pr[\mathsf{Repeat}]$ is negligible, $\mathcal{A}$ cannot gain a non-negligible advantage in distinguishing the test session key from a random key if $\Pr[\mathsf{Ask}]$ is negligible, as shown below:

$$\begin{aligned}\mathsf{Adv}_{\text{HMQV-ID}}(\mathcal{A}) &= 2 \cdot \Pr[\mathsf{Succ}] - 1 \\ &\leq 2(\Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}] + \frac{1}{2}(1 - \Pr[\mathsf{Ask}])) - 1 \\ &= 2 \cdot \Pr[\mathsf{Repeat}] + \Pr[\mathsf{Ask}].\end{aligned}$$

But, since the advantage of $\mathcal{A}$ is non-negligible, $\Pr[\mathsf{Ask}]$ is non-negligible and so is the probability that $\mathcal{A}_{\text{BDH}}$ succeeds in solving the BDH problem. This completes the proof of Theorem 5. □

7. Conclusion

We have demonstrated how an ID-based key exchange protocol as well as its security proof can be mechanically derived from an existing DH-based key exchange protocol and its corresponding security proof. As case studies, we derived the ID-based versions of three well-known DH-based protocols (UMP, UMP-ID; BJM, BJM-ID; and HMQV, HMQV-ID) along with the associated security proofs.

Table 3 describes a summary of the computational requirements and the security of two-party, two-message ID-based protocols. As pointed out by Boyd and Choo [11], most proofs for ID-based protocols have been attempted in the BR model [7] or its variant. We use M for scalar-point multiplication and P for pairing in the table. Note that one P is typically much more expensive than one M [5]. We categorize the protocols in the table into four classes according to their security levels: protocols designed to provide both forward secrecy and KCIR (class A), protocols designed to provide forward secrecy but not KCIR (class B), protocols designed to provide KCIR but not forward secrecy (class C), and protocols designed to provide only the basic SK security (class D). Our resultant protocols, HMQV-ID, UMP-ID and BJM-ID, are among the protocols that have the best online computational efficiency in their respective class. In particular, UMP-ID is superior to all other protocols listed in the table in the sense that the online computation it requires each user to perform is only one scalar-point multiplication. HMQV-ID, together with the ID-MQV protocol of Wang [48] and the protocol of Wang [49], are ranked top in class A in terms of the overall computational efficiency. UMP-ID and BJM-ID are the only protocols in class B and C, respectively, that have been proven secure in the (non-restricted) BR model. It is interesting to observe that UMP-ID is more efficient than HMQV-ID when the opposite is true for their DH-based versions.

Open problems. We end by noting that the proposal of a mechanical technique for deriving ID-based protocols from existing DH-based protocols is the main conceptual

Table 3: Comparative security and efficiency for two-party, two-message ID-based protocols from pairings.

Protocol	Computation Online	Computation Offline	Fwd. secrecy	KCIR	Security proof
HMQV-ID	$1M+1P$	$2M$	Yes (No KGC-FS)	Yes	for FS in CK model
ID-MQV [48]	$1M+1P$	$2M$	Yes (No KGC-FS)	Yes	No
Wang [49]	$2M+1P$	$1M$	Yes (No KGC-FS)	Yes	BR model
Chen & Kudla [14] #1′	$1M+1P$	$2M+1P$	Yes	Yes	No
Choie, Jeong & Lee [16] #1	$2M+3P$	$2M$	Yes (No KGC-FS)	Yes	Broken [11]
Choie, Jeong & Lee [16] #2	$1M+2P$	$1M$	Yes (No KGC-FS)	Yes	Broken [11]
Xie [51] #1	$1M+1P$	$2M+1P$	Yes (No KGC-FS)	Yes	Broken [44]
Xie [51] #2	$1M+1P$	$2M+1P$	Yes	Yes	Broken [44]
UMP-ID	$1M$	$1M+1P$	Yes (with KGC-FS)	No	BR model
McCullagh & Barreto [36] #1	$1M+1P$	$1M$	Yes (No KGC-FS)	No	Restricted BR model
McCullagh & Barreto [36] #2	$1M+1P$	$1M$	Yes (No KGC-FS)	No	Mistakes in proof [19, 15]
Wang, Cao & Cao [50]	$1M+1P$	$2M$	Yes (with KGC-FS)	No	No
BJM-ID	$1P$	$1M+1P$	No	Yes	BR model
Chen & Kudla [14] #2	$1P$	$2M$	No	Yes	Restricted BR model
Chen & Kudla [14] #2′	$1M+1P$	$2M$	No	Yes	Restricted BR model
Smart [45]	$1P$	$2M+1P$	No	Yes	No
Shim [43]	$1P$	$2M$	No	No	Broken [46]

contribution of this paper. However, in our attempts to provide a generic proof for the mapping from DH-based to ID-based protocols, we find it technically challenging. There is a need for further study on providing a generic proof approach for such a mapping, such that we can get two provably secure protocols for the price of one proof.

Acknowledgements

Part of this research was undertaken when the first author (Kim-Kwang Raymond Choo) was with Queensland University of Technology. This work was supported by Priority Research Centers Program through the National Research Foundation of Korea

(NRF) funded by the Ministry of Education (NRF-2010-0020210). We thank one of the three reviewers for bringing [48] to our attention, and Wang (author in [48]) for the subsequent correspondence.

References

[1] S. Al-Riyami, Cryptographic schemes based on elliptic curve pairings, Ph.D. thesis, Information Security Group, Department of Mathematics, Royal Holloway, University of London, 2004. Available at `http://www.isg.rhul.ac.uk/~kp/`

[2] R. Ankney, D. Johnson, M. Matyas, The unified model, contribution to X9F1 working group, 1995.

[3] ANSI X9.42, Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography, American National Standards Institute, 2003.

[4] ANSI X9.63, Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography, American National Standards Institute, 2001.

[5] P. Barreto, H. Kim, B. Lynn, M. Scott, Efficient algorithms for pairing-based cryptosystems, In Proceedings of the 22nd Annual International Cryptology Conference (CRYPTO'02), LNCS 2442: 354–368, 2002.

[6] M. Bellare, A note on negligible functions, Journal of Cryptology 15(4): 271–284, 2002.

[7] M. Bellare, P. Rogaway, Entity authentication and key distribution, In Proceedings of the 13th Annual International Cryptology Conference (CRYPTO'93), LNCS 773: 232–249, 1994.

[8] S. Blake-Wilson, D. Johnson, A. Menezes, Key agreement protocols and their security analysis, In Proceedings of the 6th IMA International Conference on Cryptography and Coding, LNCS 1355: 30–45, 1997. An extended version is available at `http://citeseerx.ist.psu.edu`

[9] S. Blake-Wilson, A. Menezes, Authenticated Diffie-Hellman key agreement protocols, In Proceedings of the 5th International Workshop on Selected Areas in Cryptography (SAC'98), LNCS 1556: 339–361, 1998.

[10] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, SIAM Journal on Computing 32(3): 585–615, 2003.

[11] C. Boyd, KKR. Choo, Security of two-party identity-based key agreement, In Proceedings of the 1st International Conference on Cryptology (MYCRYPT'05), LNCS 3715: 229–243, 2005.

[12] C. Boyd, A. Mathuria, Protocols for authentication and key establishment, Springer-Verlag, 2003.

[13] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, In Proceedings of the 20th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'01), LNCS 2045: 453–474, 2001. The full version is available at Cryptology ePrint Archive: Report 2001/040.

[14] L. Chen, C. Kudla, Identity based authenticated key agreement protocols from pairings, In Proceedings of the 16th IEEE Computer Security Foundations Workshop, 219–233, 2003. A revised version is available at Cryptology ePrint Archive: Report 2002/184.

[15] Z. Cheng, L. Chen, On security proof of McCullagh-Barreto's key agreement protocol and its variants, International Journal of Security and Networks 2(3/4): 251–259, 2007.

[16] Y. Choie, E. Jeong, E. Lee, Efficient identity-based authenticated key agreement protocol from pairings, Journal of Applied Mathematics and Computation 162(1): 179–188, 2005.

[17] KKR. Choo, A proof of revised Yahalom protocol in the Bellare and Rogaway (1993) model, Computer Journal 50(5): 591–601, 2007.

[18] KKR. Choo, C. Boyd, Y. Hitchcock, Examining indistinguishability-based proof models for key establishment protocols, In Proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'05), LNCS 3788: 585–604, 2005. An extended version is available at Cryptology ePrint Archive: Report 2005/270.

[19] KKR. Choo, C. Boyd, Y. Hitchcock, On session key construction in provably secure protocols, In Proceedings of the 1st International Conference on Cryptology (MYCRYPT'05), LNCS 3715: 116–131, 2005. An extended version is available at Cryptology ePrint Archive: Report 2005/206.

[20] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22(6): 644–654, 1976.

[21] W. Diffie, P. van Oorschot, M. Wiener, Authentication and authenticated key exchanges, Designs, Codes and Cryptography 2(2): 107–125, 1992.

[22] S. Galbraith, K. Harrison, D. Soldera, Implementing the Tate pairing, In Proceedings of the 5th International Symposium on Algorithmic Number Theory (ANTS-V), LNCS 2369: 324–337, 2002.

[23] K. Goss, Cryptographic method and apparatus for public key exchange with authentication, US Patent 4,956,863. 1990.

[24] IEEE P1363, Standard specifications for public-key cryptography, 2000.

[25] ITU-T Recommendation X.509, Information technology — open systems interconnection — the directory: Public-key and attribute certificate frameworks, 2005.

[26] I. Jeong, J. Katz, D. Lee, One-round protocols for two-party authenticated key exchange, In Proceedings of the 2nd International Conference on Applied Cryptography and Network Security (ACNS'04), LNCS 3089: 220–232, 2004. The full version is available at `http://www.cs.umd.edu/~jkatz/`

[27] A. Joux, The Weil and Tate pairings as building blocks for public key cryptosystems, In Proceedings of the 5th International Symposium on Algorithmic Number Theory (ANTS-V), LNCS 2369: 20–32, 2002.

[28] A. Joux, K. Nguyen, Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups, Journal of Cryptology 16(4): 239–247, 2003.

[29] B. Kaliski, An unknown key-share attack on the MQV key agreement protocol, ACM Transactions on Information and System Security 4(3): 275–288, 2001.

[30] M. Kim, A. Fujioka, B. Ustaoglu, Strongly secure authenticated key exchange without NAXOS' approach, In Proceedings of the 4th International Workshop on Security (IWSEC'09), LNCS 5824: 174–191, 2009.

[31] H. Krawczyk, HMQV: A high-performance secure Diffie-Hellman protocol, In Proceedings of the 25th Annual International Cryptology Conference (CRYPTO'05), LNCS 3621: 546–566, 2005. An extended version is available at Cryptology ePrint Archive: Report 2005/176.

[32] C. Kudla, K. Paterson, Modular security proofs for key agreement protocols, In Proceedings of the 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'05), LNCS 3788: 549-565, 2005.

[33] B. LaMacchia, K. Lauter, A. Mityagin, Stronger security of authenticated key exchange, In Proceedings of the 1st International Conference on Provable Security (ProvSec'07), LNCS 4784: 1–16, 2007.

[34] L. Law, A. Menezes, M. Qu, J. Solinas, S. Vanstone, An efficient protocol for authenticated key agreement, Designs, Codes and Cryptography 28(1): 119–134, 2003.

[35] T. Matsumoto, Y. Takashima, H. Imai, On seeking smart public-key distribution systems, IEICE Transactions E69(2): 99–106, 1986.

[36] N. McCullagh, P. Barreto, A new two-party identity-based authenticated key agreement, In Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA) 2005, LNCS 3376: 262–274, 2005. An extended version is available at Cryptology ePrint Archive: Report 2004/122.

[37] A. Menezes, M. Qu, S. Vanstone, Some new key agreement protocols providing implicit authentication, In Proceedings of the 2nd International Workshop on Selected Areas in Cryptography (SAC'95), 22–32, 1995.

[38] National Security Agency, SKIPJACK and KEA algorithm specifications, Version 2.0, 1998. Available from `http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf`

[39] T. Okamoto, D. Pointcheval, The gap-problems: A new class of problems for the security of cryptographic schemes, In Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC'01), LNCS 1992: 104–118, 2001.

[40] K.G. Paterson, G. Price, A comparison between traditional public key infrastructures and identity-based cryptography, Information Security Technical Report 8(3): 57–72, 2003.

[41] RFC 5280, Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, 2008.

[42] A. Shamir, Identity-based cryptosystems and signature schemes, In Proceedings of CRYPTO 1984, LNCS 196: 47–53, 1985.

[43] K. Shim, Cryptanalysis of mutual authentication and key exchange for low power wireless communications, IEEE Communications Letters 7(5): 248–250, 2003.

[44] K. Shim, S. Seo, Cryptanalysis of ID-based authenticated key agreement protocols from bilinear pairings, In Proceedings of the 8th International Conference on Information and Communications Security (ICICS'06), LNCS 4307: 410–419, 2006.

[45] N. Smart, Identity-based authenticated key agreement protocol based on the Weil pairing, Electronics Letters 38(13): 630–632, 2002.

[46] H. Sun, B. Hsieh, Security analysis of Shim's authenticated key agreement protocols from pairings, Cryptology ePrint Archive: Report 2003/113, 2003.

[47] B. Ustaoglu, Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS, Designs, Codes and Cryptography 46(3): 329–342, 2008.

[48] S. Wang, On the relations between Diffie-Hellman and ID-based key agreement from pairings, Cryptology ePrint Archive: Report 2009/437, 2009.

[49] Y. Wang, Efficient identity-based and authenticated key agreement protocol, Submitted to Transactions on Computational Sciences (arXiv: 1207.5438), 2012. An earlier version is available at Cryptology ePrint Archive: Report 2005/108.

[50] S. Wang, Z. Cao, F. Cao, Efficient identity-based authenticated key agreement protocol with PKG forward secrecy, International Journal of Network Security 7(2): 181–186, 2008.

[51] G. Xie, An ID-based key agreement scheme from pairing, Cryptology ePrint Archive: Report 2005/093, 2005.