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

Kim-Kwang Raymond Choo\textsuperscript{a}, Junghyun Nam\textsuperscript{b,\textasteriskcentered}, Dongho Won\textsuperscript{c}

\textsuperscript{a}Information Assurance Research Group, Advanced Computing Research Centre, University of South Australia, Mawson Lakes, SA 5095, Australia
\textsuperscript{b}Department of Computer Engineering, Konkuk University, 322 Danwol-dong, Chungju-si, Chungcheongbuk-do 380-701, Korea
\textsuperscript{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:

\begin{center}
\textit{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}
\end{center}
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.
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 $A$ 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 $A$, $A$ can claim this credit deposit as his. Also, $A$ 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 extension 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 func-
tions 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 \(f : \mathbb{N} \rightarrow \mathbb{R}\) 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 \(G_1\) be an additive group of
prime order \(q\) with \(|q| = k\), where \(k\) is the security parameter, and \(G_2\) be a multiplicative
group of the same order \(q\). We assume the existence of a map \(\hat{e}\) from \(G_1 \times G_1\) to \(G_2\).
Typically, \(G_1\) will be a subgroup of the group of points on an elliptic curve over a finite
field, \(G_2\) 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 $Q, W, Z \in G_1$, 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 $P, Q \in G_1$, we have $\hat{e}(P, Q) \neq 1_{G_2}$.

Computability. For some elements $P, Q \in G_1$, 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 $G$ of prime order $q$ (with $|q| = k$) while ID-based protocols operate on $\langle G_1, G_2, \hat{e} \rangle$ which are defined as above. Let $g$ and $P$ be generators of $G$ and $G_1$, respectively.

Computational Diffie–Hellman (CDH) problem. Given an instance of $(g^a, g^b) \in G^2$ (or $(aP, bP) \in G_1^2$), where $a, b \in \mathbb{Z}_q^*$, output $g^{ab} \in G$ (or $abP \in G_1$ respectively).

An algorithm, $A_{\text{CDH}}$, running in time $t$ has advantage $\epsilon$ in solving the CDH problem in $G$ (or $G_1$) if $\Pr[A_{\text{CDH}}(g^a, g^b) = g^{ab}] \geq \epsilon$ (or $\Pr[A_{\text{CDH}}(aP, bP) = abP] \geq \epsilon$), where the probability is over the random choice of $a, b \in \mathbb{Z}_q^*$, the random choice of $g \in G^*$ (or $P \in G_1^*$ respectively), and the random bits of $A_{\text{CDH}}$.

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

An algorithm, $A_{\text{DDH}}$, running in time $t$ has advantage $\epsilon$ in solving the DDH problem in $G$ if $|\Pr[A_{\text{DDH}}(g^a, g^b, g^{ab}) = 1] - \Pr[A_{\text{DDH}}(g^a, g^b, g^c) = 1]| \geq \epsilon$, where the probability is over the random choice of $a, b, c \in \mathbb{Z}_q^*$, the random choice of $g \in G^*$, and the random bits of $A_{\text{DDH}}$. 
Gap Diffie–Hellman (GDH) problem. Given an instance of \((g^a, g^b) \in G^2\), as well as an oracle \(O_{\text{GDH}}(\cdot, \cdot, \cdot)\) that solves the DDH problem in \(G\), output \(g^{ab} \in G\). Here, the oracle \(O_{\text{GDH}}(\cdot, \cdot, \cdot)\) outputs 1 if the given problem instance is a decisional Diffie–Hellman tuple, and 0 otherwise.

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

Bilinear Diffie–Hellman (BDH) problem. Given an instance of \((aP, bP, cP) \in G_1^3\), where \(a, b, c \in_R \mathbb{Z}_q^*\), output \(\hat{e}(P, P)^{abc} \in G_2\).

An algorithm, \(\mathcal{A}_{\text{BDH}}\), running in time \(t\) has advantage \(\epsilon\) in solving the BDH problem on \(\langle G_1, G_2, \hat{e} \rangle\) if \(\Pr[\mathcal{A}_{\text{BDH}}(aP, bP, cP) = \hat{e}(P, P)^{abc}] \geq \epsilon\), where the probability is over the random choice of \(a, b, c \in \mathbb{Z}_q^*\), the random choice of \(P \in G_1^*\), and the random bits of \(\mathcal{A}_{\text{BDH}}\).

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, \(\mathcal{A}_{\text{DBDH}}\), running in time \(t\) has advantage \(\epsilon\) in solving the DBDH problem on \(\langle G_1, G_2, \hat{e} \rangle\) if \(\Pr[\mathcal{A}_{\text{DBDH}}(aP, bP, cP, \hat{e}(P, P)^{abc}) = 1] - \Pr[\mathcal{A}_{\text{DBDH}}(aP, bP, cP, \hat{e}(P, P)^{d}) = 1] \geq \epsilon\), where the probability is over the random choice of \(a, b, c, d \in \mathbb{Z}_q^*\), the random choice of \(P \in G_1^*\), and the random bits of \(\mathcal{A}_{\text{DBDH}}\).

Gap bilinear Diffie–Hellman (GBDH) problem. Given \((aP, bP, cP) \in G_1^3\), as well as an oracle \(O_{\text{GBDH}}(\cdot, \cdot, \cdot, \cdot)\) that solves the DBDH problem on \(\langle G_1, G_2, \hat{e} \rangle\), output \(\hat{e}(P, P)^{abc} \in G_2\). Here, the oracle \(O_{\text{GBDH}}(\cdot, \cdot, \cdot, \cdot)\) outputs 1 if the given problem instance is a bilinear Diffie–Hellman tuple, and 0 otherwise.

An algorithm, \(\mathcal{A}_{\text{GBDH}}\), running in time \(t\) has advantage \(\epsilon\) in solving the GBDH problem on \(\langle G_1, G_2, \hat{e} \rangle\) if \(\Pr[\mathcal{A}_{\text{GBDH}}(aP, bP, cP, O_{\text{GBDH}}(\cdot, \cdot, \cdot, \cdot)) = \hat{e}(P, P)^{abc}] \geq \epsilon\), where the probability is over the random choice of \(a, b, c \in \mathbb{Z}_q^*\), the random choice of \(P \in G_1^*\), and the random bits of \(\mathcal{A}_{\text{GBDH}}\).

2.3. Communication model

Participants. Let \(U\) be a nonempty set of participants (also called users). We assume each user \(U \in U\) 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 \(\Pi_U^i\) 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 \(U' \in U\). An instance \(\Pi_U^i\) 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 Π_j^U', are partnered if all the following hold: (1) both Π_i^U and Π_j^U' have accepted, (2) sid_i^U = sid_j^U', and (3) pid_i^U = pid_j^U'.

Adversary capabilities. A PPT adversary 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.

- **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 A. If this query causes Π_i^U to accept, this will also be shown to 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).
- **Reveal**(U, i) causes the output of the session key sk_i^U held by Π_i^U.
- **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.
- **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 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 $A$, 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 $\Pi_{U,U'}^i$ is fresh unless one of the following occurs: (1) the adversary queries $\text{Reveal}(U, i)$ or $\text{Reveal}(U', j)$, where $\Pi_U^j$ is an instance partnered with $\Pi_U^i$; or (2) the adversary queries $\text{Corrupt}(U)$ or $\text{Corrupt}(U')$.

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

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

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

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

3. Mechanics of protocol derivation

3.1. System setup

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

- An additive group $G_1$ with a generator $P$ of order $q$, a multiplicative group $G_2$ of the same order $q$, and a bilinear map $\hat{e}$ from $G_1 \times G_1$ to $G_2$.

- A cryptographic hash function $G : \{0, 1\}^* \to G_1$, 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 $P_{\text{pub}} = sP \in G_1$, and the static private/public keys of each $U \in U$ to $sQ_U$ and $Q_U = G(ID_U)$, where $ID_U$ is the identity of user $U$. 

9
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 $(x, g^x)$ 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>
<thead>
<tr>
<th>Key types</th>
<th>DHP</th>
<th>IDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral private/public keys</td>
<td>$x \in \mathbb{Z}_q^*, \ g^x \in \mathbb{G}$</td>
<td>$x \in \mathbb{Z}_q^*, \ xP \in \mathbb{G}_1$</td>
</tr>
<tr>
<td></td>
<td>$y \in \mathbb{Z}_q^*, \ g^y \in \mathbb{G}$</td>
<td>$y \in \mathbb{Z}_q^*, \ yP \in \mathbb{G}_1$</td>
</tr>
<tr>
<td>Static private/public keys</td>
<td>$a \in \mathbb{Z}_q^*, \ g^a \in \mathbb{G}$</td>
<td>$sQ_A \in \mathbb{G}_1, \ Q_A \in \mathbb{G}_1$</td>
</tr>
<tr>
<td></td>
<td>$b \in \mathbb{Z}_q^*, \ g^b \in \mathbb{G}$</td>
<td>$sQ_B \in \mathbb{G}_1, \ Q_B \in \mathbb{G}_1$</td>
</tr>
<tr>
<td>Ephemeral Diffie–Hellman key</td>
<td>$g^{xy} \in \mathbb{G}$</td>
<td>$xyP \in \mathbb{G}_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\hat{e}(xP, yP)^s \in \mathbb{G}_2$</td>
</tr>
<tr>
<td>Static Diffie–Hellman key</td>
<td>$g^{ab} \in \mathbb{G}$</td>
<td>$\hat{e}(Q_A, Q_B)^s \in \mathbb{G}_2$</td>
</tr>
<tr>
<td>Static-ephemeral Diffie–Hellman keys</td>
<td>$g^{ay} \in \mathbb{G}$</td>
<td>$\hat{e}(Q_A, yP)^s \in \mathbb{G}_2$</td>
</tr>
<tr>
<td></td>
<td>$g^{bx} \in \mathbb{G}$</td>
<td>$\hat{e}(Q_B, xP)^s \in \mathbb{G}_2$</td>
</tr>
</tbody>
</table>

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 $\phi$ of DHP was proven under the CDH assumption in $\mathbb{G}$, which we denote as $\text{CDH} \leq_{\phi} \text{DHP}$.

- If the CDH-problem instance $(g^a, g^b) \in \mathbb{G}^2$ was used in place of the ephemeral public keys $(g^x, g^y)$ in the proof simulation for DHP (denoted as $(g^a, g^b) \propto (g^x, g^y)$), 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 \( \phi \) of IDP under the CDH assumption in \( \mathbb{G}_1 \). Let \( (\alpha P, \beta P) \in \mathbb{G}_1 \) 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:
  
  \(- (g^a, g^\beta) \propto (g^x, g^y) \) and \( g^{xy} \Rightarrow \hat{e}(xP, yP)^s \)
  
  \(- (g^a, g^\beta) \propto (g^a, g^b) \) and \( g^{ab} \Rightarrow \hat{e}(Q_A, Q_B)^s \)
  
  \(- (g^a, g^\beta) \propto (g^a, g^y) \) and \( g^{ay} \Rightarrow \hat{e}(Q_A, yP)^s \), or \( (g^a, g^\beta) \propto (g^b, g^x) \) and \( g^{bx} \Rightarrow \hat{e}(Q_B, xP)^s \)

In all of these three cases, we can prove the security property \( \phi \) of IDP under the BDH assumption on \( \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle \). Let \( (\alpha P, \beta P, \gamma P) \in \mathbb{G}_1 \) 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 \( \phi \) of DHP is proven under the DDH assumption in \( \mathbb{G} \) (i.e., \( \text{DDH} \subseteq \phi \), DHP). Let \( (g^a, g^\beta, g^\gamma) \in \mathbb{G}^3 \) be the DDH-problem instance given to the simulator in the proof for DHP.

- Consider, first, the case that \( (g^a, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy}) \) and \( g^{xy} \Rightarrow xP \). In this case, we cannot rely on the DDH assumption to prove the security property \( \phi \) 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 \( \gamma = \alpha \beta \mod q \) 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 \( \phi \) 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:
  
  \(- (g^a, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy}) \) and \( g^{xy} \Rightarrow \hat{e}(xP, yP)^s \)
  
  \(- (g^a, g^\beta, g^\gamma) \propto (g^a, g^b, g^{ab}) \) and \( g^{ab} \Rightarrow \hat{e}(Q_A, Q_B)^s \)
  
  \(- (g^a, g^\beta, g^\gamma) \propto (g^x, g^y, g^{xy}) \) and \( g^{ay} \Rightarrow \hat{e}(Q_A, yP)^s \), or \( (g^a, g^\beta, g^\gamma) \propto (g^b, g^x, g^{bx}) \) and \( g^{bx} \Rightarrow \hat{e}(Q_B, xP)^s \)

In these cases, we can prove the security property \( \phi \) of IDP under the DBDH assumption on \( \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle \). Let \( (\alpha P, \beta P, \gamma P, \hat{e}(P, P)^\delta) \in \mathbb{G}_1 \times \mathbb{G}_2 \) 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

\[- (xP, yP, P_{\text{Pub}}, \hat{e}(xP, yP)^s),\]

\[- (Q_A, Q_B, P_{\text{Pub}}, \hat{e}(Q_A, Q_B)^s),\]

\[- (Q_A, yP, P_{\text{Pub}}, \hat{e}(Q_A, yP)^s) \text{ or } (Q_B, xP, P_{\text{Pub}}, \hat{e}(Q_B, xP)^s)),\]

in each of the three cases, respectively.

GDH. Suppose that a security property \( \phi \) of DHP is proven under the GDH assumption in \( G \) (i.e., GDH \( \leq \phi \) DHP). Let \( (g^\alpha, g^\beta) \in G^2 \) 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 \( \phi \) of IDP under the CDH assumption in \( G_1 \).

Note that a DDH oracle for \( G_1 \) is not needed for the proof since the DDH problem in \( G_1 \) is easy; each DDH-oracle access can be replaced by two evaluations of the bilinear map \( \hat{e} \). Given the CDH-problem instance \( (\alpha P, \beta P) \in G_1^2 \), the simulator in the proof for IDP will embed it into the simulation by using \( \alpha P \) and \( \beta P \) 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 \( \phi \) of IDP under the GBDH assumption on \( \langle G_1, G_2, \hat{e} \rangle \). 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 \( P_{\text{Pub}} \) to be \( \gamma P \) 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 \( P_{\text{Pub}} = \gamma P \), 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 \( G_1 \) (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 \( U \in U \) 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 \( r_U \in Z_q^* \) 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.

<table>
<thead>
<tr>
<th>Assumptions</th>
<th>Embedding</th>
<th>Assumptions</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDH</td>
<td>((g^a, g^b) \propto (g^x, g^y))</td>
<td>CDH</td>
<td>((\alpha, \beta, \gamma) \propto (P, \hat{P}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>BDH</td>
<td>((\alpha, \beta, \gamma) \propto (xP, yP, P_{pub}))</td>
</tr>
<tr>
<td>DDH</td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>DDH</td>
<td>The DDH problem in (G_1) is easy</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b, g^c) \propto (g^a, g^b, g^{ab}))</td>
<td>DDH ((\times))</td>
<td>((\alpha, \beta, \gamma, \hat{P}, \hat{e}(P, P)^{\hat{\beta}}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b, g^c))</td>
<td>DBDH</td>
<td>((\alpha, \beta, \gamma, \hat{P}, \hat{e}(P, P)^{\hat{\beta}}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b, g^c))</td>
<td>DBDH</td>
<td>((\alpha, \beta, \gamma, \hat{P}, \hat{e}(P, P)^{\hat{\beta}}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b, g^c))</td>
<td>DBDH</td>
<td>((\alpha, \beta, \gamma, \hat{P}, \hat{e}(P, P)^{\hat{\beta}}))</td>
</tr>
<tr>
<td>GDH</td>
<td>((g^a, g^b) \propto (g^x, g^y))</td>
<td>CDH</td>
<td>((\alpha, \beta) \propto (xP, yP))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>GBDH</td>
<td>((\alpha, \beta, \gamma) \propto (xP, yP, P_{pub}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>GBDH</td>
<td>((\alpha, \beta, \gamma) \propto (Q_A, Q_B, P_{pub}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>GBDH</td>
<td>((\alpha, \beta, \gamma) \propto (Q_A, Q_B, P_{pub}))</td>
</tr>
<tr>
<td></td>
<td>((g^a, g^b) \propto (g^a, g^b))</td>
<td>GBDH</td>
<td>((\alpha, \beta, \gamma) \propto (Q_A, Q_B, P_{pub}))</td>
</tr>
</tbody>
</table>

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 \(G\) 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]
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 \((G_1, G_2, \hat{e}, P, G, H)\), the master private/public keys \((s, P_{pub} = sP)\) of KGC, and the static private/public keys \((s_{Q_U}, Q_U = G(ID_U))\) of each user \(U \in \mathcal{U}\). The two groups \(G_1\) and \(G_2\) have the order \(q\) and we assume that \(q\) is implicit in \(G_1\) and \(G_2\). 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 \((s_{Q_A}, Q_A)\) and \((s_{Q_B}, Q_B)\), the ephemeral Diffie–Hellman key \(g^{xy}\) with \(xyP\), and the static Diffie–Hellman key \(g^{ab}\) with \(\hat{e}(Q_A, Q_B)\). 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.
As mentioned above, UMP was proven SK-secure in a restricted adversary model [8]. Since $\text{CDH} \leq_{\text{SK}} \text{UMP}$, $(g^a, g^b) \propto (g^a, g^b)^5$ 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 $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ 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 $(\alpha P, \beta P, \gamma P) \in \mathbb{G}_3^1$ 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

---

As described in Section 3.3, this notation means that in the proof for UMP, the simulator embedded the CDH-problem instance $(g^a, g^b)$ 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 T S 2. Since CDH \leq_{FS} T S 2, (g^a, g^b) \propto (g^x, g^y) and g^{xy} \Rightarrow xyP, we can prove KGC forward secrecy of UMP-ID under the CDH assumption in G_1. As described in Table 2, the simulator in the proof will embed the CDH-problem instance \((\alpha P, \beta P) \in G_2^2\) 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 \(A\) 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 an algorithm \(A_{\text{BDH}}\) that solves the BDH problem on \(\langle G_1, G_2, \hat{e} \rangle\) with a non-negligible advantage. The objective of \(A_{\text{BDH}}\) is to compute and output the value \(\hat{e}(P, P)^{\alpha \beta \gamma} \in G_2^2\) when given a BDH-problem instance \((\alpha P, \beta P, \gamma P) \in G_3^2\) where \(\alpha, \beta, \gamma \in \mathbb{F}_q^*\). Let \(\Pi'_{U,U'}\) denote \(\Pi_{U,U'}^i\) for any \(i\).

\(A_{\text{BDH}}\) runs \(A\) while simulating the oracles on its own. \(A_{\text{BDH}}\) 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 \(P_{\text{pub}} = \gamma P\). Here, \(A\) and \(B\) are two users chosen at random from the set of all users, in the hope that \(A\) will ask its Test query against \(\Pi'_{A,B}\) or \(\Pi'_{B,A}\). For each \(U \in \mathbb{U} \setminus \{A, B\}\), \(A_{\text{BDH}}\) 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, \(A\) will never get direct access to the random oracle \(G\).) \(A_{\text{BDH}}\) 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 \(A\) corrupts \(A\) or \(B\), then \(A_{\text{BDH}}\) aborts. When \(A\) asks the Test query, \(A_{\text{BDH}}\) responds with a random \(k\)-bit string. For all other queries of \(A\), \(A_{\text{BDH}}\) handles them exactly in the same way as the oracles would do. At some point in time, \(A\) will terminate and output its guess \(b'\). When this happens, \(A_{\text{BDH}}\) selects an entry of the form \((\rho || \sigma, h)\) at random from HList, terminates and outputs \(\sigma\).

Let \(\text{Ask}\) be the event that \(A\) makes the query \(H(\rho_U || \sigma_U)\) when \(\Pi'_{U}^t\) is the test instance. Let \(q_H\) be the number of \(H\) queries made by \(A\). Then, the following is immediate from the simulation:

- Since Reveal queries are not allowed (meaning that key-replication attacks are not possible), \(A_{\text{BDH}}\) outputs the desired result \(\hat{e}(P, P)^{\alpha \beta \gamma}\) with probability at least \(1/q_H\) if \(\text{Ask}\) 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 \(1/\binom{|\mathbb{U}|}{2} = \frac{2}{|\mathbb{U}|(|\mathbb{U}| - 1)}.\) (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: \(A_{\text{BDH}}\) outputs the desired result \(\hat{e}(P, P)^{\alpha \beta \gamma}\) with probability at least

\[
\Pr[\text{Ask}] \cdot \frac{1}{q_H} \cdot \frac{2}{|\mathbb{U}|(|\mathbb{U}| - 1)}
\]

16
which is non-negligible as long as $\Pr[\text{Ask}]$ is non-negligible.

Now, to prove the theorem, it suffices to prove that $\Pr[\text{Ask}]$ is non-negligible. Since $H$ is a random oracle and the Reveal oracle is not available, $A$ 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:

$$\text{Adv}_{\text{UMP-ID}}(A) = 2 \cdot \Pr[\text{Succ}] - 1$$

$$\leq 2(\Pr[\text{Ask}] + \frac{1}{2}(1 - \Pr[\text{Ask}])) - 1$$

$$= \Pr[\text{Ask}].$$

Therefore, if the advantage of $A$ is non-negligible, $\Pr[\text{Ask}]$ is non-negligible and so is the probability that $A_{\text{BDH}}$ 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 $A$ who can gain a non-negligible advantage in distinguishing the test session key from random. Then we can construct an algorithm $A_{\text{GBDH}}$ that has a non-negligible advantage in solving the GBDH problem on $(G_1, G_2, \hat{e})$. The goal of $A_{\text{GBDH}}$ is to output the value $\hat{e}(P, P)^{\alpha\beta\gamma} \in G_2$ when given a triple of elements $\alpha P, \beta P, \gamma P \in G_1$, where $\alpha, \beta, \gamma \in R Z_q^*$, as well as an oracle $O_{\text{DBDH}}(\cdot, \cdot, \cdot, \cdot)$ that solves the DBDH problem on $(G_1, G_2, \hat{e})$. Let $\Pi^*_U,U'$ be $\Pi_U,U'$ 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 $\Pi^*_U$. Then, $kds^i_U = \text{pid}^i_U \parallel \text{sid}^i_U \parallel \rho^i_U \parallel \sigma^i_U$.

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

- **Send:** $A_{\text{GBDH}}$ 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:

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

where $q_{\text{send}}$ is the number of Send queries made by $A$. Note that no two unpartnered instances can compute the same kds unless Repeat occurs.
• **Corrupt**: If $\mathcal{A}$ corrupts $A$ or $B$, then $\mathcal{A}_{GBDH}$ aborts. Otherwise, $\mathcal{A}_{GBDH}$ returns the static private key of the queried user.

• **H**: $\mathcal{A}_{GBDH}$ uses a list, HList, to maintain input-output pairs of $H$. For each $H$ query on a string $m$, $\mathcal{A}_{GBDH}$ first checks if HList contains an entry of the form $(m, h)$. If it does, $\mathcal{A}_{GBDH}$ returns $h$. Otherwise, $\mathcal{A}_{GBDH}$ checks that $O_{DBDH}(\alpha P, \beta P, \gamma P, m_{G_2}) = 1$, where $m_{G_2}$ is a $|G_2|$-bit string that is a suffix of $m$. If this is true, $m_{G_2} = \hat{e}(P, P)^{\alpha \beta \gamma}$ and therefore, $\mathcal{A}_{GBDH}$ will terminate and output $m_{G_2}$. Otherwise, $\mathcal{A}_{GBDH}$ returns a random $k$-bit string $str$ to $\mathcal{A}$ and adds $(m, str)$ to HList.

• **Reveal**: Given a Reveal query on any instance $\Pi^*_U$, $\mathcal{A}_{GBDH}$ proceeds as follows:

1. If $\Pi^*_U \neq \Pi^*_{A,B}$ and $\Pi^*_U \neq \Pi^*_{B,A}$, $\mathcal{A}_{GBDH}$ computes $kds^*_U$ and checks if HList contains an entry of the form $(kds^*_U, h)$. If it does, $\mathcal{A}_{GBDH}$ returns $h$ to $\mathcal{A}$. Otherwise, $\mathcal{A}_{GBDH}$ selects a random $k$-bit string $str$, returns $str$ to $\mathcal{A}$, and adds $(kds^*_U, str)$ into HList.

2. If $\Pi^*_U = \Pi^*_{A,B}$ or $\Pi^*_U = \Pi^*_{B,A}$, $\mathcal{A}_{GBDH}$ cannot compute $kds^*_U$ as it cannot compute $\sigma^*_U = \hat{e}(P, P)^{\alpha \beta \gamma}$. In this case, $\mathcal{A}_{GBDH}$ computes $\rho^*_U$, and checks if an entry of the form $(pid^*_U||sid^*_U||\rho^*_U||sk)$ is in a list called RList which $\mathcal{A}_{GBDH}$ maintains to store the revealed session keys of $\Pi^*_{A,B}$ and $\Pi^*_{B,A}$. If it is, $\mathcal{A}_{GBDH}$ returns $sk$ to $\mathcal{A}$. Otherwise, $\mathcal{A}_{GBDH}$ selects a random $k$-bit string $str$, returns $str$ to $\mathcal{A}$, and adds $(pid^*_U||sid^*_U||\rho^*_U, str)$ into the RList.

When $\mathcal{A}$ asks its Test query, $\mathcal{A}_{GBDH}$ responds with a random $k$-bit string. Let Ask be the event that $\mathcal{A}$ makes the query $H(pid^*_U||sid^*_U||\rho^*_U||\sigma^*_U)$ when $\Pi^*_U$ is the test instance. When $\mathcal{A}$ terminates and outputs its guess $b'$ (meaning that Ask did not occur or $\mathcal{A}_{GBDH}$’s guess on the test instance was wrong), $\mathcal{A}_{GBDH}$ terminates and outputs a random $|G_2|$-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 $\Pi^*_{B,A}$, $\mathcal{A}_{GBDH}$ 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 $\frac{2}{|U||U| - 1}$.

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

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

$$\Pr[\text{Ask}] \geq \frac{2}{|U||U| - 1} \left(1 - \frac{q_{send}(q_{send} - 1)}{2|Z_q^*|}\right)$$
which is non-negligible if \( \Pr[\text{Ask}] \) is non-negligible.

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

\[
\text{Adv}_{\text{UMP-ID}}(\mathcal{A}) = 2 \cdot \Pr[\text{Succ}] - 1
\leq 2(\Pr[\text{Repeat}] + \Pr[\text{Ask}] + \frac{1}{2}(1 - \Pr[\text{Ask}])) - 1
\leq 2 \cdot \Pr[\text{Repeat}] + \Pr[\text{Ask}].
\]

But since the advantage of \( \mathcal{A} \) is non-negligible (by assumption), we obtain that \( \Pr[\text{Ask}] \) 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 \( \text{Send}(U, i, M) \) query, we say that the \( \text{Send} \) query was passively-generated if the message \( M \) was output by a previous \( \text{Send} \) query.

**Definition 5 (FS-freshness).** We say that an instance is FS-fresh if (1) the \( \text{Send} \) queries directed to the instance are passively-generated ones and (2) the adversary has not issued a \( \text{Reveal} \) 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 \( \alpha P \) and \( \beta P \) 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. \(A_{\text{CDH}}\) 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 \(A\) asks its Test query, \(A_{\text{CDH}}\) simply outputs a random \(k\)-bit string. For all other queries of \(A\), \(A_{\text{CDH}}\) handles them in the straightforward way. When \(A\) terminates and outputs its guess \(b'\), \(A_{\text{CDH}}\) selects an entry of the form \((\text{pid}||\text{sid}||\rho||\sigma, h)\) at random from HList, terminates and outputs \(\rho\).

Let \(kds^t_U\) 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 \(A\) makes the query \(H(kds^t_U)\). The statement of the theorem easily follows if we make the following observations:

- If Ask occurs and if \(A_{\text{CDH}}\)’s guess on the test instance is correct, \(A_{\text{CDH}}\) 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 \(A\).

- The probability that \(A_{\text{CDH}}\)’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 \(\alpha P\) and \(\beta P\)) is \(\frac{2}{n(n-1)} = \frac{1}{n^2}\).

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

Therefore, the probability that \(A_{\text{CDH}}\) outputs \(\alpha\beta P\) is at least
\[
\Pr[\text{Ask}] \cdot \frac{1}{q_H} \cdot \frac{2}{n(n-1)} \left(1 - \frac{q_{\text{send}}(q_{\text{send}} - 1)}{2|\mathbb{Z}_q^*|}\right)
\]
which is non-negligible as long as \(\Pr[\text{Ask}]\) is non-negligible.

It is not hard to see that \(\Pr[\text{Ask}]\) is non-negligible. Since \(H\) is a random oracle and \(\Pr[\text{Repeat}]\) is negligible, \(A\) cannot gain a non-negligible advantage in distinguishing the test session key from a random key if \(\Pr[\text{Ask}]\) is negligible (see Eq. (1) in the proof of Theorem 2). Therefore, \(\Pr[\text{Ask}]\) is non-negligible and so is the probability of \(A_{\text{CDH}}\) 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: \textit{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
$x \in \mathbb{Z}_q^\ast \quad y \in \mathbb{Z}_q^\ast$

$\rho_A = (g^y)^x \quad \rho_B = (g^a)^y$

$\sigma_A = (g^b)^y \quad \sigma_B = (g^x)^b$

$s_{k_A} = H(\rho_A \| \sigma_A) \quad s_{k_B} = H(\rho_B \| \sigma_B)$

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

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 $\preceq_{SK,KCIR}$ KP-BJM, $(g^\alpha, g^\beta) \propto (g^h, g^x)$ and $g^{hx} \Rightarrow \hat{e}(Q_B, xP)^s$, we can prove the security of BJM-ID under the GBDH assumption on $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$. 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 $\Pi_{U,U'}^i$ is KCIR-fresh if none of the following occurs:
1. The adversary queries Reveal$(U, i)$ or Reveal$(U', j)$, where $\Pi_i^j$ is partnered with $\Pi_i^j$.
2. The adversary queries 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 $A$, the advantage of $A$ 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 $A$ who can distinguish the test session key from random with a non-negligible advantage, we build an algorithm $A_{GBDH}$ that solves the GBDH problem on $\langle G_1, G_2, \hat{e} \rangle$ with a non-negligible advantage. Let $(\alpha P, \beta P, \gamma P) \in G_3^3$ be an instance of the GBDH problem given to $A_{GBDH}$. Then, the goal of $A_{GBDH}$ is to compute and output the value $\hat{e}(P, P)^{\alpha \beta \gamma} \in G_2$ when given access to an oracle $O_{DBDH}(\cdot, \cdot, \cdot, \cdot)$ that solves the DBDH problem on $\langle G_1, G_2, \hat{e} \rangle$.

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

Now, $A_{GBDH}$ invokes $A$ as a subroutine and answers the oracle queries on its own. $A_{GBDH}$ answers all the Send queries of $A$ as per the protocol specification, except that $A_{GBDH}$ uses $\beta P$ as the ephemeral public key of $\Pi_A^t$ and aborts if the event Repeat occurs (see the proof of Theorem 2 for details on the event Repeat). When $A$ asks a Corrupt$(U)$ query, $A_{GBDH}$ 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 $kds_U^t = pid_U^t || sid_U^t || r_U^t || \sigma_U^t$ denote the $kds$ of instance $\Pi_U^t$. As is clear from the above simulation, $A_{GBDH}$ cannot compute $kds_B^s$ for any $s$ if the ephemeral public key received by $\Pi_B^s$ has been generated by $A$. But, given a string $m$, $A_{GBDH}$ can determine whether $m = kds_B^s$ or not, as shown below:
Deciding $m = kds^*_B$

Let $x'P$ be the ephemeral public key received by $\Pi'_B$. Given a string $m$, $A_{\text{GBDH}}$ first checks if the (bit) length of $m$ is equal to the length of key derivation strings. If it is, then $A_{\text{GBDH}}$ checks that (1) $\phi_B^i||sid_B^i||\rho^*_B$ is a prefix of $m$ and (2) $O_{\text{DBDH}}(\alpha P, x'P, \gamma P, m_{G_2}) = 1$ where $m_{G_2}$ is a $|G_2|$-bit string that is a suffix of $m$. If both are true, then $m = kds^*_B$.

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

- **$H$:** $A_{\text{GBDH}}$ uses a list, $H$List, to maintain input-output pairs of $H$. For each $H$ query on a string $m$, $A_{\text{GBDH}}$ first checks if an entry of the form $(m, h)$ is in $H$List. If it is, then $A_{\text{GBDH}}$ returns $h$ to $A$. Otherwise, $A_{\text{GBDH}}$ checks that $O_{\text{DBDH}}(\alpha P, \beta P, \gamma P, m_{G_2}) = 1$, where $m_{G_2}$ is a $|G_2|$-bit string that is a suffix of $m$. If this is true, $m_{G_2} = \hat{e}(P, P)^{\alpha \beta \gamma}$ and therefore, $A_{\text{GBDH}}$ will terminate and output $m_{G_2}$. Otherwise, $A_{\text{GBDH}}$ checks if $m = kds^*_B$ for some $s$. Given the string $m$, this check can be done by performing the above deciding operation for each tuple $(\phi_B^i||sid_B^i||\rho^*_B, sk)$ in the RList which is maintained by $A_{\text{GBDH}}$ to store revealed session keys of instances of $B$. If such a tuple $(\phi_B^i||sid_B^i||\rho^*_B, sk)$ exists in RList, $A_{\text{GBDH}}$ returns $sk$ to $A$ and adds $(m, sk)$ into $H$List. Otherwise, $A_{\text{GBDH}}$ returns a random $k$-bit string $str$ to $A$ and adds $(m, str)$ into $H$List.

- **$\text{Reveal}$:** Given a $\text{Reveal}$ query on any instance $\Pi'_U$, $A_{\text{GBDH}}$ proceeds as follows:
  1. If $\Pi'_U = \Pi'_A$ or $\Pi'_U = \Pi'_A'$, where $\Pi'_A$, $\Pi'_A'$ is partnered with $\Pi'_A'$, $A_{\text{GBDH}}$ aborts.
  2. If $\Pi'_U = \Pi'_B$ for any $s$, $A_{\text{GBDH}}$ computes $\rho^*_B$ and checks if a tuple of the form $(\phi_B^i||sid_B^i||\rho^*_B, sk)$ is in the RList. If it is, $A_{\text{GBDH}}$ returns $sk$ in response to the query. Otherwise, $A_{\text{GBDH}}$ checks if $H$List contains an entry $(m, h)$ such that $m = kds^*_B$. Given $\phi_B^i||sid_B^i||\rho^*_B$, this check can be done by performing the deciding operation for all entries in $H$List. If such entry $(m, h)$ exists in $H$List, $A_{\text{GBDH}}$ returns $h$ to $A$ and adds $(\phi_B^i||sid_B^i||\rho^*_B, h)$ into RList. Otherwise, $A_{\text{GBDH}}$ returns a random $k$-bit string $str$ to $A$, and adds $(\phi_B^i||sid_B^i||\rho^*_B, str)$ into RList.
  3. Otherwise, $A_{\text{GBDH}}$ computes $kds^*_U$ and checks if an entry of the form $(kds^*_U, h)$ is in HList. If it is, $A_{\text{GBDH}}$ returns $h$ to $A$. Otherwise, $A_{\text{GBDH}}$ returns a random $k$-bit string $str$ to $A$, and adds $(kds^*_U, str)$ into HList.

When $A$ asks its $\text{Test}$ query, $A_{\text{GBDH}}$ responds with a random $k$-bit string. Let $\text{Ask}$ be the event that $A$ makes an $H$ query on $kds$ of the test instance. When $A$ terminates and outputs its guess $b'$ (meaning that $\text{Ask}$ did not occur or $A_{\text{GBDH}}$’s guess on the test instance was wrong), $A_{\text{GBDH}}$ terminates and outputs a random $|G_2|$-bit string.
The probability that $A_{\text{GBDH}}$ succeeds in solving the GBDH problem can be easily obtained by noticing the following:

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

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

$$\Pr[\text{Ask}] \frac{1}{n|U|} \left(1 - \frac{q_{\text{send}}(q_{\text{send}} - 1)}{2|Z_q^*|}\right)$$

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

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

$$\text{Adv}_{BJM-ID}(A) = 2 \cdot \Pr[\text{Succ}] - 1$$

$$\leq 2(\Pr[\text{Repeat}] + \Pr[\text{Ask}] + \frac{1}{2}(1 - \Pr[\text{Ask}]))) - 1$$

$$= 2 \cdot \Pr[\text{Repeat}] + \Pr[\text{Ask}]$$

Since the advantage of $A$ is non-negligible (by assumption), we obtain that $\Pr[\text{Ask}]$ is non-negligible and so is the probability of $A_{\text{GBDH}}$ 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 $\bar{H} : \{0, 1\}^* \rightarrow \{0, 1\}^{n/2}$ and $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$, where $q$ is the prime order of the underlying group $G$ 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.
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 \( H : \{0, 1\}^* \rightarrow \{0, 1\}^{|q|/2} \) and \( H : \{0, 1\}^* \rightarrow \{0, 1\}^k \), in addition to the basic parameters \((G_1, G_2, e, P, G)\). 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 \(\sigma_A\) and \(\sigma_B\), 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 \(g^{xy}, g^{ady}, g^{bx} \) and \(g^{abde}\) with \(\hat{e}(xP, yP)^s, \hat{e}(Q_A, yP)^{sd}, \hat{e}(Q_B, yP)^{se}\) and \(\hat{e}(Q_A, Q_B)^{sde}\), respectively, and then making some mathematical derivations, as shown below:

\[
\sigma_A = \hat{e}(xP, yP)^s \cdot \hat{e}(Q_A, yP)^{sd} \cdot \hat{e}(Q_B, xP)^{se} \cdot \hat{e}(Q_A, Q_B)^{sde}
\]

\[
= \hat{e}(xyP, sP) \cdot \hat{e}(sQ_A, dyP) \cdot \hat{e}(xeQ_B, sP) \cdot \hat{e}(sQ_A, deQ_B)
\]

\[
= \hat{e}(xT_B, P_{pub}) \cdot \hat{e}(sQ_A, dT_B) \cdot \hat{e}(xeQ_B, P_{pub}) \cdot \hat{e}(sQ_A, deQ_B)
\]

\[
= \hat{e}(xT_B + xeQ_B, P_{pub}) \cdot \hat{e}(sQ_A, dT_B + deQ_B)
\]

\[
= \hat{e}(T_B + eQ_B, P_{pub})^x \cdot \hat{e}(sQ_A, T_B + eQ_B)^d
\]

\[
= \hat{e}(T_B + eQ_B, xP_{pub}) \cdot \hat{e}(dsQ_A, T_B + eQ_B)
\]

\[
= \hat{e}(T_B + eQ_B, xP_{pub}) \cdot \hat{e}(T_B + eQ_B, dsQ_A)
\]

\[
= \hat{e}(T_B + eQ_B, xP_{pub} + dsQ_A),
\]

Figure 5: The HMQV protocol.
Figure 6: HMQV-ID: An ID-based version of protocol HMQV.

\[
\begin{align*}
\sigma_B &= \hat{e}(xP, yP)^s \cdot \hat{e}(yQ_B, yP)^s \cdot \hat{e}(Q_B, xP)^s \cdot \hat{e}(Q_A, Q_B)^{sde} \\
&= \hat{e}(xP, sP) \cdot \hat{e}(dyQ_A, sP) \cdot \hat{e}(sQ_B, eT_A) \cdot \hat{e}(deQ_A, sQ_B) \\
&= \hat{e}(yT_A, P_{P_{ub}}) \cdot \hat{e}(dyQ_A, P_{P_{ub}}) \cdot \hat{e}(sQ_B, eT_A) \cdot \hat{e}(deQ_A, sQ_B) \\
&= \hat{e}(yT_A + dyQ_A, P_{P_{ub}}) \cdot \hat{e}(eT_A + deQ_A, sQ_B) \\
&= \hat{e}(T_A + dQ_A, P_{P_{ub}})^y \cdot \hat{e}(T_A + dQ_A, sQ_B)^e \\
&= \hat{e}(T_A + dQ_A, yP_{P_{ub}}) \cdot \hat{e}(T_A + dQ_A, eS_{Q_B}) \\
&= \hat{e}(T_A + dQ_A, yP_{P_{ub}} + eS_{Q_B}) = \sigma_A.
\end{align*}
\]

Note above that the factor \(g^{xy}\) of \(\sigma_A\) and \(\sigma_B\) 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 \(g^{abdde}\) (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., \(xP_{P_{ub}}\) and \(xP\) by \(A\); \(yP_{P_{ub}}\) 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\(\textsuperscript{6}\); 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 $\text{CDH} \leq_{FS} \text{HMQV}$, \((g^\alpha, g^\beta) \propto (g^x, g^y)\) and \(g^{xy} \Rightarrow \hat{e}(xP, yP)^s\), we can prove forward secrecy of HMQV-ID under the BDH assumption on \(\langle G_1, G_2, \hat{e} \rangle\). As shown in Table 2, the simulator in the proof for HMQV-ID will embed the BDH-problem instance \((\alpha P, \beta P, \gamma P) \in G_3^1\) 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 $\Pi_U$ is **FS-fresh in eCK** if all the following hold:

1. The Send queries asked against $\Pi_U$ are passively-generated ones.
2. The adversary has not issued a Reveal query against $\Pi_U$ and its partner instance.
3. The adversary has not issued an EphemeralKeyReveal query against $\Pi_U$ 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 $A$ 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.

\(\textsuperscript{6}\)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 $A$ who has a non-negligible advantage in breaking forward secrecy of HMQV-ID. Given the adversary $A$, we prove the theorem by constructing an algorithm $A_{BDH}$ that has a non-negligible advantage in solving the BDH problem on $(\mathbb{G}_1, \mathbb{G}_2, \tilde{e})$. The goal of $A_{BDH}$ is to compute and output the value $\tilde{e}(P, P)^{\alpha \beta \gamma} \in \mathbb{G}_2$ 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 $A$ 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. $A_{BDH}$ 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 $r_u \in R \mathbb{Z}_q^*$. $A_{BDH}$ then runs $A$ while simulating the oracles on its own. Let $n$ be the maximum number of instances that $A$ may activate. $A_{BDH}$ chooses two instances $\Pi^i_U$ and $\Pi^j_U$, at random from all the $n$ instances. $A_{BDH}$ answers $\text{Send}$ queries of $A$ as per the protocol specification, except that: (1) it uses $\alpha P$ and $\beta P$ as the outgoing messages of $\Pi^i_U$ and $\Pi^j_U$, and (2) it aborts if the event $\text{Repeat}$ occurs (see the proof of Theorem 2 for the definition of $\text{Repeat}$). If $A$ asks an $\text{EphemeralKeyReveal}$ query against $\Pi^i_U$ or $\Pi^j_U$, then $A_{BDH}$ aborts. For all other $\text{EphemeralKeyReveal}$ queries, $A_{BDH}$ answers them in the obvious way. $A_{BDH}$ 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 $\text{HList}$. When $A$ asks its $\text{Test}$ query, $A_{BDH}$ outputs a random $k$-bit string. For all other queries of $A$, $A_{BDH}$ handles them in the straightforward way. When $A$ terminates and outputs its guess $b'$, $A_{BDH}$ selects an entry of the form $(\text{pid}||\text{sid}||\sigma, h)$ at random from $\text{HList}$, computes

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

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

Let $q_{send}$ and $q_H$ be the numbers of $\text{Send}$ queries and $H$ queries, respectively, made by $A$. Let $\text{Ask}$ be the event that $A$ 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 $A_{BDH}$ succeeds in solving the BDH problem can be easily calculated by observing the following:

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

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

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

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

$$\Pr[\text{Ask}] \frac{1}{q_H n(n-1)} \left( 1 - \frac{q_{send}(q_{send} - 1)}{2|\mathbb{Z}_q^*|} \right).$$
which is non-negligible if $\Pr[\text{Ask}]$ is non-negligible. We finally show that $\Pr[\text{Ask}]$ is non-negligible. Since $H$ is a random oracle and $\Pr[\text{Repeat}]$ is negligible, $A$ cannot gain a non-negligible advantage in distinguishing the test session key from a random key if $\Pr[\text{Ask}]$ is negligible, as shown below:

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

But, since the advantage of $A$ is non-negligible, $\Pr[\text{Ask}]$ is non-negligible and so is the probability that $A_{\text{BDH}}$ 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.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Computation</th>
<th>Fwd. secrecy</th>
<th>KCIR</th>
<th>Security proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMQV-ID</td>
<td>$1M+1P$</td>
<td>$2M$</td>
<td>Yes (No KGC-FS)</td>
<td>Yes for FS in CK model</td>
</tr>
<tr>
<td>ID-MQV [48]</td>
<td>$1M+1P$</td>
<td>$2M$</td>
<td>Yes (No KGC-FS)</td>
<td>No</td>
</tr>
<tr>
<td>Wang [49]</td>
<td>$2M+1P$</td>
<td>$1M$</td>
<td>Yes (No KGC-FS)</td>
<td>Yes BR model</td>
</tr>
<tr>
<td>Chen &amp; Kudla [14] #1'</td>
<td>$1M+1P$</td>
<td>$2M+1P$</td>
<td>Yes</td>
<td>Yes No</td>
</tr>
<tr>
<td>Xie [51] #1</td>
<td>$1M+1P$</td>
<td>$2M+1P$</td>
<td>Yes (No KGC-FS)</td>
<td>Yes Broken [44]</td>
</tr>
<tr>
<td>Xie [51] #2</td>
<td>$1M+1P$</td>
<td>$2M+1P$</td>
<td>Yes</td>
<td>Yes Broken [44]</td>
</tr>
<tr>
<td>UMP-ID</td>
<td>$1M$</td>
<td>$1M+1P$</td>
<td>Yes (with KGC-FS)</td>
<td>No BR model</td>
</tr>
<tr>
<td>McCullagh &amp; Barreto [36] #1</td>
<td>$1M+1P$</td>
<td>$1M$</td>
<td>Yes (No KGC-FS)</td>
<td>No Restricted BR model</td>
</tr>
<tr>
<td>Wang, Cao &amp; Cao [50]</td>
<td>$1M+1P$</td>
<td>$2M$</td>
<td>Yes (with KGC-FS)</td>
<td>No No</td>
</tr>
<tr>
<td>BJM-ID</td>
<td>$1P$</td>
<td>$1M+1P$</td>
<td>No</td>
<td>Yes BR model</td>
</tr>
<tr>
<td>Chen &amp; Kudla [14] #2</td>
<td>$1P$</td>
<td>$2M$</td>
<td>No</td>
<td>Yes Restricted BR model</td>
</tr>
<tr>
<td>Chen &amp; Kudla [14] #2'</td>
<td>$1M+1P$</td>
<td>$2M$</td>
<td>No</td>
<td>Yes Restricted BR model</td>
</tr>
<tr>
<td>Smart [45]</td>
<td>$1P$</td>
<td>$2M+1P$</td>
<td>No</td>
<td>Yes No</td>
</tr>
<tr>
<td>Shim [43]</td>
<td>$1P$</td>
<td>$2M$</td>
<td>No</td>
<td>No Broken [46]</td>
</tr>
</tbody>
</table>

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


