Computationally-Fair Group and Identity-Based Key-Exchange*

Andrew C. Yao[1] and Yunlei Zhao[2]

[1] Tsinghua University, Beijing, China
[2] Fudan University, Shanghai, China

Abstract. In this work, we re-examine some fundamental group key-exchange and identity-based key-exchange protocols, specifically the Burmester-Desmedet group key-exchange protocol [7] (referred to as the BD-protocol) and the Chen-Kudla identity-based key-exchange protocol [9] (referred to as the CK-protocol). We identify some new attacks on these protocols, showing in particular that these protocols are not computationally fair. Specifically, with our attacks, an adversary can do the following damages:

– It can compute the session-key output with much lesser computational complexity than that of the victim honest player, and can maliciously nullify the contributions from the victim honest players.
– It can set the session-key output to be some pre-determined value, which can be efficiently and publicly computed without knowing any secrecy supposed to be held by the attacker.

We remark these attacks are beyond the traditional security models for group key-exchange and identity-based key-exchange, which yet bring some new perspectives to the literature of group and identity-based key-exchange. We then present some fixing approaches, and prove that the fixed protocols are computationally fair.

1 Introduction

Key-exchange (KE) protocols are basic to modern cryptography and to secure systems in general. KE protocols are used to generate a common secret-key among a set of users for encryption, authentication and for enforcing access-control policies. Among them, the Diffie-Hellman key-exchange (DHKE) protocol marked the birth of public-key cryptography, and serves as the basis for most key-exchange protocols.

Usually, key-exchange (particularly DHKE) protocols are considered in the two party setting under a public-key infrastructure (PKI). Two important extension dimensions are group key-exchange (GKE) and identity-based key-exchange (IBKE). Group key-exchange extends the standard two-party KE protocol to the multiple-party case. The Burmester-Desmedet group key-exchange protocol [7]

* This work is supported in part by NSFC grants No. 61033001 and No. 61070248. Preliminary version of this work appears in the PCT patent file [20].

(referred to as the BD-protocol) is an extension of DHKE into the group setting, which is one of the most fundamental group key-exchange protocols and serves as a basis for many group key-exchange protocols in the literature. Identity-based key-exchange simplifies public-key certificate management in traditional PKI-based key-exchange, where users' identities themselves can serve as the public-keys (but at the price of introducing a trusted authority called private key generator that generates the secret-keys for all users). A list of identity-based key-exchange protocols have been developed in the literature [8], among which the Chen-Kudla identity-based key-exchange protocol [9] (referred to as the CK-protocol) is one of the most efficient IBKE protocols.

In this work, we re-examine the BD-protocol and the CK-protocol. We identify some new attacks on these protocols, showing in particular that these protocols are not computationally fair. Specifically, with our attacks, an adversary can do the following damages:

– It can compute the session-key output with much lesser computational complexity than that of the victim honest player, and can maliciously make the contributions from the victim honest players be of no effect.
– It can set the session-key output to be some pre-determined value, which can be efficiently and publicly computed without knowing any secret value supposed to be held by the attacker.

We note these attacks are beyond the traditional security models for group key-exchange and identity-based key-exchange, which yet bring some new perspectives to the literature of group and identity-based key-exchange. We then present some fixing approaches, and prove that the fixed protocols are computationally fair.

2 Preliminaries

If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots; r)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. If S is a finite set then $x \leftarrow S$, sometimes also written as $x \in_{\mathrm{R}} S$, is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < \frac{1}{\lambda^c}$ for all $\lambda > \lambda_c$.

Let G' be a finite Abelian group of order N, G be a cyclic subgroup of prime order q in G'. Denote by g a generator of G, by 1_G the identity element, by $G \setminus 1_G = G - \{1_G\}$ the set of elements of G except 1_G. Throughout this paper, unless explicitly specified, for presentation simplicity we assume G is a multiplicative group, and use multiplicative notation to describe the group operation in G'. (When G' is defined w.r.t. elliptic curves over finite fields, usually addition notation is used for the group operation in G'.)

Let $(A = g^a \in G, a)$ (resp., $(X = g^x \in G, x)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{A}$, and $(B = g^b \in G, b)$ (resp., $(Y = g^y \in G, y)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{B}$, where a, x, b, y are taken randomly and independently from Z_q^*. The (basic version of) DHKE protocol [11] works as follows: after exchanging their DH-components X and Y, player $\hat{A}$ (resp., $\hat{B}$) computes the session-key $K = Y^x = g^{xy}$ (resp., $K = X^y = g^{xy}$). The security of DHKE relies on the computational Diffie-Hellman (CDH) assumption over G, which says that given $X = g^x, Y = g^y \leftarrow G$ (i.e., each of x and y is taken uniformly at random from Z_q) no efficient (say, probabilistic polynomial-time) algorithm can compute $CDH(X, Y) = g^{xy}$.

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a Diffie-Hellman key-exchange protocol $\langle \hat{A}, \hat{B} \rangle$ are run concurrently over an asynchronous network like the Internet. To distinguish concurrent sessions, each session run at the side of an uncorrupted player is labeled by a tag, which is the concatenation, in the order of session initiator and then session responder, of players' identities/public-keys and DH-components available from the session transcript. For identity-based key-exchange, we also include the public-key of the private-key generator (that is a trusted authority) into the session-tag. A session-tag is complete if it consists of a complete set of all these components.

Admissible Pairing: Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing [2, 6], where G is a cyclic multiplicative (or additive) group of order q generated by an element g. Here, an admissible pairing $\hat{e}$ satisfies the following three properties:

- Bilinear: If $g_1, g_2 \in G$, and $x, y \in Z_q$, then $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1_{G_T}$, where 1_{G_T} is the identity element in G_T. In particular, $\hat{e}(g, g)$ is the generator of G_T in case G_T is also a cyclic group of the same order q.
- Computable: If $g_1, g_2 \in G$, $\hat{e}(g_1, g_2) \in G_T$ can be computed in polynomial-time.

2.1 Non-malleably Independent Dominant-Operation Values, and Session-Key Computational Fairness

In this section, we review and discuss the notion of session-key computational fairness recently introduced by Yao, et al [21].

For any complete session-tag Tag of a key-exchange protocol among n users $\{U_1, \cdots, U_n\}$ where $n \geq 2$, we first identify *dominant-operation values* w.r.t. Tag and each user U_i, $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m, m \geq 2$, which are specified to compute the session-key K by the honest player U_i for a complete session specified by the complete session-tag Tag, where G_k, $1 \leq k \leq m$ is the range of V_k^i. Specifically, $K = F_K(V_1^i, \cdots, V_m^i, Tag)$, where K is the session-key output by user U_i, F_K is some polynomial-time computable function (that is defined by the session-key computation specified for honest players). We remark that dominant operations are specific to protocols, where for different key-exchange protocols the dominant operations can also be different.

Then, roughly speaking, we say that a key-exchange protocol enjoys session-key computational fairness *w.r.t some pre-determined dominant operations*, if for any complete session-tag Tag, the session-key computation *involves* the same number of *non-malleably independent* dominant-operation values for each user U_i, $1 \leq i \leq n$, whether it is honest or malicious.

Definition 1 (non-malleable independence). *For the dominant-operation values,* $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m$, $m \geq 2$ *and* $1 \leq i \leq n$, *w.r.t. a complete session-tag* Tag *on any sufficiently large security parameter* λ, *we say* $V_1^i, \cdots, V_m^i$ *are computationally (resp., perfectly) non-malleably independent, if for any polynomial-time computable (resp., any power unlimited) relation/algorithm* $\mathcal{R}$ *(with components drawn from* $G_1 \times \cdots \times G_m \times \{0,1\}^*$*) it holds that the following quantity is negligible in* λ *(resp., just* 0*):*

$$\left|\Pr[\mathcal{R}(V_1^i, \cdots, V_m^i, Tag) = 1] - \Pr[\mathcal{R}(R_1, \cdots, R_m, Tag) = 1]\right|,$$

where $R_i, 1 \leq i \leq m$ *is taken uniformly and independently from* G_i, *and the probability is taken over the random coins of* $\mathcal{R}$ *(as well as the choice of the random function in the random oracle model [3]).*

Remark: As clarified in [21], the above Definition 1 is defined w.r.t. any complete session-tag and w.r.t. some pre-determined dominant operations, which does not explicitly take the malicious player's ability into account. But, this definition ensures that, by the birthday paradox, for any successfully finished session among a set of (malicious and honest) players, no matter how the malicious players collude, it holds that: for any $i, 1 \leq i \leq n$ and for any values $(\alpha_1, \cdots, \alpha_m) \in G_1 \times \cdots \times G_m$, the probability that $\Pr[V_k^i = \alpha_i]$ is negligible for any $k, 1 \leq k \leq m$ and any $i, 1 \leq i \leq n$.

Definition 2 ((session-key) computational fairness). *We say a key-exchange protocol enjoys session-key computational fairness w.r.t. some pre-determined dominant operations, if for any complete session-tag* Tag *on any sufficiently large security parameter* λ, *the session-key computation involves the same number of (perfectly or computationally)* non-malleably independent *dominant-operation values for any user* U_i, $1 \leq i \leq n$.

Remark: Note that the notion of "(session-key) computational fairness" is defined w.r.t. some predetermined dominant operations that are uniquely determined by the protocol specification. We remark that it is the task of the protocol designer to specify the dominant operations (for which computational fairness can be provably proved), which should also be natural and common to the literature. Though computational fairness is defined w.r.t. some pre-determined dominant operations, as clarified, this property holds for arbitrary efficient adversaries. Also note that, for presentation simplicity, session-key computational fairness is defined w.r.t. either perfect or computational non-malleable independence. In general, we can define perfect (resp., computational) session-key computational fairness w.r.t. perfect (resp., computational) non-malleable independence. Here, we would like to point out that, session-key computational fairness refers to the fairness in *computing* session-key, while the word "computational" in "computational non-malleable independence" refers to indistinguishability between probability distributions.

More detailed discussions and clarifications, on the formulation of non-malleable independence and session-key computational fairness, are referred to [21].

3 Re-examination of the Burmester-Desmedet Group Key-Exchange Protocol

In this section, we re-examine the Burmester-Desmedet GKE protocol [7], referred to as the BD-protocol for presentation simplicity. We show an attack against the BD-protocol, where some malicious players can collude to nullify the effects of victim honest players, and discuss the consequences of the identified attack. We then present a fixed protocol called computationally-fair BD-protocol, and show the fixed protocol is computationally-fair (while the original BD-protocol is not).

3.1 Brief Review of the Burmester-Desmedet Group Key-Exchange Protocol

Suppose $U_1, U_2, \cdots, U_n, n \geq 2$, be a group of parties who want to share a common group key among them. Let G be a cyclic group of order q generated by a generator g. The BD-protocol works as follows:

– Each $U_i, 1 \leq i \leq n$, takes x_i uniformly at random from Z_q^*, computes $X_i = g^{x_i}$, and finally broadcasts X_i to all other users.
– After receiving X_j, for all j where $1 \leq j \neq i \leq n$, each user U_i computes and broadcasts $Z_i = (X_{i+1}/X_{i-1})^{x_i}$, where the indices are taken in a cycle (i.e., $\mod n$).
– Finally, each user U_i, $1 \leq i \leq n$, computes the shared session-key $K = (X_{i-1})^{nx_i} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2}$. Note that the session-key generated by all the users is the same, specifically $K = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$.

The tag for a complete BD-protocol session is defined to be $(U_1, U_2, \cdots, U_n, X_1, X_2, \cdots, X_n)$.

3.2 An Attack against the BD-Protocol

We demonstrate an attack against the BD-protocol. We illustrate our attack for the case $n = 3$, where two malicious users U_1 U_2 collude to be against an honest user U_3.[1]. The attack works as follows: U_2 sets X_2 to be X_1^{-1} (i.e., $x_2 = -x_1$), then the shared DH-secret is $K_1 = K_2 = K_3 = g^{-x_1^2}$, no matter what DH-exponent x_3 is chosen by the honest U_3.

Consequence of the Attack: Note that, as x_1 may be maliciously generated by U_1 (i.e., x_1 can be an arbitrary value in Z_q), the shared DH-secret $g^{-x_1^2}$ can be

[1] The attack can be easily extended to the general case of $n > 3$, where some malicious players collude to be against sets of honest players

an arbitrary value in G with no guarantee on its randomness and independence. Furthermore, suppose the colluding U_1 and U_2 use the same X_1 and $X_2 = X_1^{-1}$ in different sessions, then the shared session-keys among different sessions are the same, i.e., always $g^{-x_1^2}$, no matter what efforts are made desperately by the third victim honest player. This is clearly *unfair* to the honest player U_3. We note that even the universally composable (UC) version of the BD-protocol, proposed in [15, 16], still does not frustrate the above attack (specifically, the fairness issue and particularly the above attack were not captured by the UC framework there).

The above concrete attack shows some unfairness, in generating group session-key, between honest victim players and malicious colluding players, and such an unfairness issue can cause essential damages.

3.3 Computationally-Fair Group Key-Exchange

We present a variant of the BD-protocol, computationally-fair BD-protocol (referred to as the fBD-protocol for presentation simplicity). The fBD protocol works as follows:

- Each $U_i, 1 \leq i \leq n$, takes x_i uniformly at random from Z_q^*, computes $X_i = g^{x_i}$, and finally broadcasts X_i to all other users.
- After receiving X_j, for all j where $1 \leq j \neq i \leq n$, each user U_i computes and broadcasts $Z_i = X_{i+1}^{x_i h(U_i, x_i, x_{i+1})} / X_{i-1}^{x_i h(U_{i-1}, x_{i-1}, x_i)}$, where the indices are taken in a cycle, and $h : \{0,1\}^* \rightarrow Z_q^*$ is a hash function that is modeled to be a random oracle in security analysis. Note that, as the indices are taken in a cycle of n (i.e., $\mod n$), $Z_n = X_1^{x_n h(U_n, x_n, x_1)} / X_{n-1}^{x_n h(U_{n-1}, x_{n-1}, x_n)}$.
- Finally, each user U_i, $1 \leq i \leq n$, computes the shared session-key $K = (X_{i-1})^{n x_i h(U_{i-1}, X_{i-1}, X_i)} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2}$. Note that the session-key output by all the users is the same, specifically

$$K = g^{x_1 x_2 h(U_1, X_1, X_2) + x_2 x_3 h(U_2, X_2, X_3) + \cdots + x_n x_1 h(U_n, X_n, X_1)}.$$

We note that the above fBD protocol can be converted into an authenticated group KE by the general technique of [16], and password-based group KE by the technique of [1]. It's easy to check that our fBD-protocol ensures the following properties in the random oracle model: (1) For any value $\alpha \in G/1_G$ and any i, $1 \leq i \leq n$, as long as U_i is honest, i.e., x_i is distributed uniformly at random over Z_q^*, it is guaranteed that the probability that the shared session-key K is equal to α is negligible, no matter how the rest players collude against it. Formally, we have:

For the fBD-protocol and any complete session-tag Tag, the dominant-operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1 x_2 h(U_1, X_1, X_2)}, V_2^i = g^{x_2 x_3 h(U_2, X_2, X_3)}, \cdots, V_n^i = g^{x_n x_1 h(U_n, X_n, X_1)}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$.

For the original BD-protocol and any complete session-tag Tag, the dominant operation values for user U_i can be specified as, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2}, V_2^i = g^{x_2x_3}, \cdots, V_n^i = g^{x_nx_1}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$.

Theorem 1. *In the random oracle model where the hash function h is assumed to be a random oracle, the fBD-protocol is session-key computationally fair, while the original BD-protocol is not, w.r.t. the above specified dominant operations.*

Proof (sketch). For both the BD-protocol and the fBD-protocol, the dominant-operation (involved in session-key computation) is defined to be modular exponentiation. A complete session-tag Tag consists of $(U_1, U_2, \cdots, U_n, X_1, X_2, \cdots, X_n)$.

For the fBD-protocol and any complete session-tag Tag, the dominant-operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2h(U_1,X_1,X_2)}, V_2^i = g^{x_2x_3h(U_2,X_2,X_3)}, \cdots, V_n^i = g^{x_nx_1h(U_n,X_n,X_1)}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$. Let $G_1 = G_2 = \cdots = G_n = G \backslash 1_G$, it is clear that, *in the random oracle model*, the distribution of $(V_1^i, V_2^i, \cdots, V_n^i)$ is identical to the distribution of $(R_1, R_2, \cdots, R_n)$, where each $R_k, 1 \leq k \leq n$ is taken uniformly at random from $G \backslash 1_G$. That is, $(V_1^i, V_2^i, \cdots, V_n^i)$ are perfectly non-malleably independent, and each user involves computing the same number (say n) of non-malleably independent dominant operations values. Thus, the fBD-protocol enjoys session-key computational fairness.

For the original BD-protocol and any complete session-tag Tag, the dominant operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2}, V_2^i = g^{x_2x_3}, \cdots, V_n^i = g^{x_nx_1}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$. Clearly, with $n = 3$ as the illustration example, our above attack shows that the distribution of (V_1^i, V_2^i, V_3^i) under our attack is $(g^{-x_1^2}, g^{-x_1x_3}, g^{x_1x_3})$, which is clearly different from the uniform independent distribution (R_1, R_2, R_3). Thus, the original BD-protocol is not of session-key computational fairness. $\square$

Computational Fairness vs. Contributiveness. A related notion, called *contributiveness*, was also introduced in the literature of group key-exchange (see e.g., [4,5,10,19]. Roughly speaking, the notion of contributiveness for group key-exchange says that a subset of players cannot pre-determine the session-key output. But, contributiveness says nothing about computational fairness in computing the session-key output. As clarified in Section 2.1, computational fairness says that each player needs to compute the same number of *non-malleably independent* dominant-operation values in generating the session-key output. (To our knowledge, the notion of non-malleably independent dominant-operation values was not previously considered in the literature of group key-exchange.) If we view each non-malleably independent dominant-operation value as a proof-of-knowledge of the corresponding secrecy, our notion of computational fairness ensures that a subset of malicious players cannot set the session-output to be some value that can be publicly computed from the session transcript. From these observations, we can see that computational fairness and contributiveness are two fundamentally different notions.

4 Re-examination of the Chen-Kudla Identity-Based Key-Exchange Protocol

In this section, we re-examine the Chen-Kudla identity-based key-exchange protocol [9], referred to as the CK-protocol for presentation simplicity. We show an attack against the CK-protocol (for the case of $\hat{A} = \hat{B}$), where an attacker can successfully finish a session with a victim honest player but without knowing any secrecy supposed to be known by it. Moreover, the attacker can set the session-key output to be some predetermined value with computational complexity significantly lesser than that of the victim honest player. We then present a fixed protocol called computationally-fair CK-protocol, and show that the fixed protocol is computationally-fair (while the original CK-protocol is not).

4.1 Brief Review of the Chen-Kudla Identity-Based Key-Exchange Protocol

Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing, where G is a cyclic multiplicative (or additive) group of order q generated by an element g. For presentation simplicity, below we assume G is a cyclic multiplicative group. The (basic version of) Chen-Kudla protocol (with escrow) works as follows [9]:

- **Setup:** The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. PKG also specifies a map-to-point hash function $H_1 : \{0,1\}^* \rightarrow G$ and a key-derivation function KDF. The public parameters are: $(G, G_T, \hat{e}, g, S, H_1, KDF)$.
- **User secret-key extract:** For a user with identity $\hat{A}$, the public-key is given by $A = H_1(\hat{A})$, and the PKG generates the associated secret-key of the user as $S_A = A^s$. Similarly, a user of identity $\hat{B}$ is of public-key $B = H_1(\hat{B})$ and secret-key $S_B = B^s$.
- **Key agreement between two users $\hat{A}$ and $\hat{B}$:**
  1. $\hat{A}$ picks $x \in Z_q^*$ at random, computes $X = A^x$ and sends X to $\hat{B}$.
  2. $\hat{B}$ picks $y \in Z_q^*$ at random, computes $Y = B^y$ and sends Y to $\hat{A}$.
  3. $\hat{A}$ computes $K_{\hat{A}} = \hat{e}(S_A, YB^x)$. Similarly, $\hat{B}$ computes $K_{\hat{B}} = \hat{e}(XA^y, S_B)$. Note that if $\hat{A}$ and $\hat{B}$ follow the protocol, they will compute the same shared secret: $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A, B)^{s(x+y)}$. Then, the actual session-key is derived from $K = KDF(K_{\hat{A}}) = KDF(K_{\hat{B}})$.

The session-tag for a complete session of the CK-protocol is $Tag = (S, \hat{A}, \hat{B}, X, Y)$.

4.2 An Attack on the CK-protocol for $\hat{A} = \hat{B}$

In some scenarios, a party may want to establish a secure channel with itself (i.e., $\hat{A} = \hat{B}$ in this case). For example, a mobile user that communicates to its desktop computer, while both the mobile device and the desktop have the same identity [17]. Below, we show an attack on the CK-protocol, by which an adversary $\mathcal{A}$ can successfully finish a session with $\hat{A}$ in the same name of $\hat{A}$ (i.e., impersonating $\hat{B} = \hat{A}$) but without knowing the corresponding DH-exponent y (i.e., the discrete logarithm of Y) or the secret-key S_A. The attack works as follows:

After receiving $X = A^x$ from $\hat{A}$, the adversary $\mathcal{A}$ (impersonating $\hat{B} = \hat{A}$) randomly selects $\alpha \in Z_q$ and sends back $Y = g^{\alpha} X^{-1}$ in the same name $\hat{B} = \hat{A}$. *Note that, denote $Y = A^y = B^y$ (recall $A = B$), $\hat{A}$ does not know the secret exponent y.* Finally, $\mathcal{A}$ computes $K_{\hat{B}} = \hat{e}(S, A)^{\alpha}$, and then derives the session-key from $K_{\hat{B}}$. Note that, as $B = A$ (and thus $S_A = S_B$) and $Y = A^y = B^y$ and $XY = g^{\alpha}$, $\hat{e}(S, A)^{\alpha} = \hat{e}(g^{\alpha}, S_A) = \hat{e}(XY, S_A) = \hat{e}(XA^y, S_B) = K_{\hat{B}}$. This shows that $\mathcal{A}$ successfully finishes the session but without knowing either the DH-exponent y or the secret-key S_A. Moreover, suppose $\alpha = 0$, then $K_{\hat{B}} = K_{\hat{A}} = 1_{G_T}$, where 1_{G_T} is the identity element in G_T. In general, α can be a small number in Z_q. This clearly indicates the unfairness between the attacker $\mathcal{A}$ and the honest player $\hat{A}$ in computing the session-key. The attacker can predicate the session-key output and can only expend constant time (for the case $\alpha = 0$) or one paring and one *small* exponentiation (for the case of small non-zero α), while the honest player $\hat{A}$ has to compute at least one pairing and one *full* exponentiation.

4.3 Computational Fair Identity-Based Key-Exchange

In this section, we present a variant of the CK-protocol, referred to as computationally-fair CK-protocol (fCK-protocol) for presentation simplicity. The only difference between the fCK-protocol and the original CK-protocol is the way of computing $K_{\hat{A}}$ and $K_{\hat{B}}$. Specifically, in the fCK-protocol, the values $K_{\hat{A}}$ and $K_{\hat{B}}$ are set to be: $K_{\hat{A}} = \hat{e}(S_A, B^{xc} Y^d)$ and $K_{\hat{B}} = \hat{e}(X^c A^{yd}, S_B)$, where $c = h(S, \hat{A}, X)$ and $d = h(S, \hat{B}, Y)$ and S is the public-key of PKG and $h : \{0,1\}^* \to Z_q^*$ is a hash function that is modeled to be a random oracle in security analysis.

For both the CK-protocol and the fCK-protocol, the dominant operation (involved in session-key computation) is defined to be modular exponentiation in the group G_T. A complete session-tag Tag consists of $(S, \hat{A}, \hat{B}, X, Y)$. For the fCK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for user $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sxc}, V_2 = \hat{e}(A, B)^{syd}\}$, while for the original CK-protocol, the dominant operation values specified for player $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sx}, V_2 = \hat{e}(A, B)^{sy}\}$.

Theorem 2. *In the random oracle model where the hash function $h : \{0,1\}^* \to Z_q^*$ is assumed to be a random oracle, the fCK-protocol is of session-key computational fairness, while the original CK-protocol is not, w.r.t. the dominant operations specified above.*

Proof. Note that for fCK-protocol, $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A, B)^{sxc+syd}$, where $c = h(S, \hat{A}, X)$ and $d = h(S, \hat{B}, Y)$. Recall that $X = A^x$ and $Y = B^y$. Recalll that for both the CK-protocol and the fCK-protocol, the dominant operation (involved in session-key computation) is defined to be modular exponentiation in the group G_T.

For the fCK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for user $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sxc}, V_2 = \hat{e}(A, B)^{syd}\}$. The function F_K is specified to be $F_K(V_1, V_2, Tag) = V_1 \cdot V_2$. Let $G_1 = G_2 = G_T \setminus 1_{G_T}$, it is clear that, *in the random oracle model*, the distribution of (V_1, V_2) is identical to the distribution of (R_1, R_2), where each $R_1, 1 \leq k \leq 2$ is taken uniformly at random from $G_T \setminus 1_{G_T}$. That is, (V_1, V_2) are perfectly non-malleably independent, and each user involves computing the same number (say 2) of non-malleably independent dominant operations values. Thus, the fCK-protocol enjoys session-key computational fairness.

For the original CK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for player $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sx}, V_2 = \hat{e}(A, B)^{sy}\}$. The function F_K is specified to be $F_K(V_1, V_2, Tag) = V_1 \cdot V_2$. Let $\mathcal{R}_{1_{G_T}}$ be the $\mathcal{NP}$-relation that $\mathcal{R}_{1_{G_T}}(V_1, V_2, Tag) = 1$ if $V_1 \cdot V_2 = 1_{G_T}$. Let $\mathcal{R}_\alpha$, for a value $\alpha \in Z_q$, be the $\mathcal{NP}$-relation that $\mathcal{R}_{1_{G_T}}(V_1, V_2, Tag) = 1$ if $V_1 \cdot V_2 = \hat{e}(S, A)^\alpha$. Then, our above attack shows that the original CK-protocol does not enjoy session-key computational fairness (particularly with respect to the relations $\mathcal{R}_{1_{G_T}}$ and $\mathcal{R}_\alpha$). □

References

1. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-Based Group Key Exchange in a Constant Number of Rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Bresson, E., Manulis, M.: Securing Group Key Exchange Against Strong Corruptions. In: ASIACCS 2008, pp. 249–260. ACM (2008)
5. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure Group Key Establishment Revisited. International Journal of Information Security 6(4), 243–254 (2007)
6. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
8. Choudary Gorantla, M., Gangishetti, R., Saxena, A.: A Survey on ID-Based Cryptographic Primitives. Cryptology ePrint Archive, Report No. 2005/094 (2005)
9. Chen, L., Kudla, C.: Identity Based Key Agreement Protocols From Pairings. In: IEEE Computer Security Foundations Workshop, pp. 219–233 (2002); Full version available at: Cryptology ePrint Archive, Report 2002/184 (2002)

10. Desmedt, Y., Pieprzyk, J., Steinfeld, R., Wang, H.: A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)
11. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transaction on Information Theory 22(6), 644–654 (1976)
12. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource Fairness and Composability of Cryptographic Protocols. Journal of Cryptology 24(4), 615–658 (2011)
13. Goldwasser, S., Lindell, Y.: Secure Computation without Agreement. Journal of Cryptology 18(3), 247–287 (2005)
14. Gordon, D.M.: A Survey of Fast Exponentiation Methods. Journal of Algorithms 27(1), 129–146 (1998)
15. Katz, J., Shin, J.: Modeling Insider Attackss on Group Key Exchange. In: ACM CCS 2005, pp. 180–189 (2005)
16. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
17. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
18. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, pp. 617–619. CRC Press (1995)
19. Mitchell, C.J., Ward, M., Wilson, P.: Key Control in Key Agreement Protocols. Electronic Letters 34(10), 980–981 (1998)
20. Yao, A.C., Zhao, Y.: Method and Structure for Self-Sealed Joint Proof-of-Knowledge and Diffie-Hellman Key-Exchange Protocols. PCT Patent, No.PCT/CN2008/072794 (August 2008); Online available from Global Intellectual Property Office (GIPO)
21. Yao, A.C., Zhao, Y.: A New Family of Practical Non-Malleable Diffie-Hellman Protocols CoRR abs/1105.1071 (2011)