Privacy-Preserving Authenticated Key-Exchange Over Internet

Andrew Chi-Chih Yao and Yunlei Zhao

***Abstract*— Key-exchange, in particular Diffie–Hellman key-exchange (DHKE), is among the core cryptographic mechanisms for ensuring network security. For key-exchange over the Internet, both security and privacy are desired. In this paper, we develop a family of privacy-preserving authenticated DHKE protocols named deniable Internet key-exchange (DIKE), both in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity and practical (online) efficiency. They provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard. To the best of our knowledge, our protocols are the first provably secure DHKE protocols that additionally enjoy all the following privacy protection advantages: 1) forward deniability, actually concurrent non-malleable statistical zero-knowledge, for both protocol participants simultaneously; 2) the session transcript and session-key can be generated merely from DH-exponents (together with some public values), which thus cannot be traced to the pair of protocol participants; and 3) exchanged messages do not bear peer's identity, and do not explicitly bear player role information.**

***Index Terms*— Authentication, Diffie–Hellman, key exchange, security, privacy, deniability, restricted random oracle.**

I. Introduction

THE Internet Key-Exchange (IKE) protocols [27], [29] are the *core* cryptographic protocols to ensure Internet security, which specify key exchange mechanisms used to establish shared keys for use in the Internet Protocol Security (IPsec) standards [30]. The IPsec and IKE are intended to protect messages communicated in the IP layer, i.e., "layer 3" of ISO-OSI, which process the transmission of messages using the network addresses *possibly without knowing end-user peers' identities*. The IKE and IPsec can in turn be used to offer confidentiality, authentication and privacy for communication protocols in the higher layers of ISO-OSI.

Manuscript received December 4, 2012; revised May 31, 2013; accepted November 9, 2013. Date of publication December 3, 2013; date of current version December 23, 2013. This work was supported in part by the National Basic Research Program of China (973 Program) under Grants 2007CB807900, 2007CB807901, and 2014CB340600, in part by the National Natural Science Foundation of China under Grants 61033001, 61061130540, 61070248, 61332019, and 61272012, in part by the Innovation Project of Shanghai Municipal Education Commission under Grant 12ZZ013, and in part by the Joint Project of SKLOLS. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. C.-C. Jay Kuo. *(Corresponding author: Y. Zhao.)*

A. C.-C. Yao is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China (e-mail: andrewcyao@tsinghua.edu.cn).

Y. Zhao is with the Software School, Fudan University, Shanghai 200433, China, and also with Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education (Wuhan University) and State Key Laboratory of Information Security (Beijing), China (e-mail: ylzhao@fudan.edu.cn).

Digital Object Identifier 10.1109/TIFS.2013.2293457

The standard of IKE has gone through two generations. The first generation IKEv1 [27] uses public-key encryption as the authentication mechanism. The second generation IKEv2 [29] uses signatures as the authentication mechanism, with the SIGMA protocol [31] serving as the basis.

The IKEv2 protocol is based on DHKE [20], and works in the "post-specified peer" setting [29], where the information of who the other party is does not necessarily exist at the session initiation stage and is learnt by the party only after the protocol run evolves (even just in the last round). Actually, this is quite a common case for KE protocols in practice, particularly for the purpose of preserving players' privacy. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation; Or, a party may respond to a request (for a KE session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smart-card that authenticates the legitimacy of the card-reader before disclosing its own identity) [13].

For key-exchange protocols, both security and privacy are desired. Actually, providing a certain level of privacy protection serves as one of the major criteria underlying the evolution of a list of important industrial standards of KE protocols, which is particularly witnessed by the evolution of IKE [29] that is based on the SIGMA protocol [31]. Among privacy concerns, deniability is an essential privacy property, and has always been a central concern in personal and business communications, with off-the-record communication serving as an essential social and political tool [18], [19], [21], [23], [39]. Given that many of these interactions now happen over digital media (email, instant messaging, web transactions, virtual private networks), it is of critical importance to provide these communications with "off-the-record" or deniability capability to protocol participants.

Traditional deniability only considers the privacy of the honest prover against a possibly malicious verifier, and requires that the interactions between them be computationally simulatable, i.e., computational zero-knowledge (ZK) [25]. That is, given a session transcript, the malicious verifier cannot prove that the honest prover was ever involved in the conversation. However, as clarified by Di Raimondo et al. in [18], there are scenarios in which deniability is actually a concern to the receiver's privacy as well. What we would like to happen is that if the prover acts honestly during the protocol, it also should not be able at a later stage to claim the messages are authentic in order to violate the privacy of the verifier. This property is called *forward deniability*, as it has some affinity to the notion of forward secrecy. It is shown in [18] that computational ZK does not guarantee forward deniability, but statistical ZK does.

Whenever deniability of messages is desired, in general, we can just run a deniable authentication protocol [23] for each message to be sent. However, the beauty of using deniable key-exchange is that if the key-exchange protocol is deniable, then all the transactions (of *public* messages) using the session-key produced by the key-exchange protocol can be deniable for both the protocol participants. Moreover, for the IKE protocol that is the core cryptographic protocol to ensure Internet security, offering deniability by IKE running at the IP layer within the IPsec standard [30] is much more desirable, because it enables various privacy services to be offered at the higher layers with uncompromised quality. Note that a privacy problem at the IP layer can cause irreparable privacy damage at the application layer. For example, an identity connected to an IP address, if not deniable, certainly nullifies an anonymous property offered by a fancy cryptographic protocol running at the application level. (If deniability is not desired, for some cases, then a non-repudiable proof, e.g., a signature, can always be issued at the application level.)

Despite its seemingly conceptual simplicity, designing "*sound*" and "*right*" key-exchange protocols turns out to be extremely error prone and can be notoriously subtle. Also, the analysis of even a simple cryptographic protocol in intricate adversarial settings like the Internet can be a luxury and dauntingly complex task [10], [32]. The reader is referred to the comprehensive book by Boyd and Mathuria [7] for a good survey about key-exchange protocol constructions and various attacks. The reason for this is the high system complexity and enormous number of subtleties in protocol design and analysis (surrounding the various trade-off among security, privacy and protocol efficiency).

A. Our Contributions

In this work, develop a family of privacy-preserving (particularly, *deniable*) authenticated DHKE protocols, named deniable Internet key-exchange (DIKE), in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity, practical (online) efficiency, provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard [27], [29] and the SIGMA protocol [31].

The security of DIKE is analyzed in accordance with the Canetti-Krawczyk framework (CK-framework) [12] *with post-specified peers* in the random oracle (RO) model. We also make discussions on a list of concrete yet essential security properties of DIKE, *most of which are beyond the CK-framework*. We then define CNMSZK for DHKE , along with detailed clarifications and justifications. To our knowledge, our formulation of CNMSZK for DHKE stand for the strongest definition of deniability, to date, for key-exchange protocols. The CNMSZK property of our protocols is analyzed in the restricted random oracle model [45], under an extension of the knowledge-of-exponent assumption [17] named concurrent knowledge-of-exponent (CKEA) that might be of independent interest.

II. Preliminaries

If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots ; \rho)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins ρ. We let $y \leftarrow A(x_1, x_2, \cdots ; \rho)$ denote the experiment of picking ρ at random and letting y be $A(x_1, x_2, \cdots ; \rho)$. If $\mathcal{S}$ is a finite set then $|\mathcal{S}|$ is its cardinality, and $(x_1, \cdots, x_\nu) \leftarrow \mathcal{S}$, $\nu \geq 1$, is the operation of picking ν elements uniformly and independently from $\mathcal{S}$ (that is, each value x_i, $1 \leq i \leq \nu$, is taken uniformly and independently from the set $\mathcal{S}$). If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. By $\Pr[E : R_1; \cdots ; R_n]$ we denote the probability of event E, after the *ordered* execution of random processes $R_1, \cdots, R_n$. A string means a binary one, and for arbitrary strings e_1 and e_2, $e_1||e_2$ denotes the concatenation of e_1 and e_2. For any hash function H, we usually write $H(x_1||x_2|| \cdots ||x_k)$ simply as $H(x_1, x_2, \cdots, x_k)$.

Let G' be a finite Abelian group of order N, and $G = \langle g \rangle$ be a unique subgroup of G', generated by the generator g, of prime order q. Denote $Z_q = \{0, 1, \cdots, q-1\}$ and $Z_q^* = \{1, 2, \cdots, q-1\}$. Denote by 1_G the identity element of G' and by $G \setminus 1_G = G - \{1_G\}$ the set of elements of G except 1_G. In the specification of this paper, w.l.o.g., we assume G' is the multiplicative group Z_p^* of order $N = p - 1$ for a large prime p, and G is the unique subgroup of order q for some prime divisor q of $N = p - 1$. Typically, the length of q, denoted $|q| = k$ (usually presented in unary as 1^k), is treated as the *security* parameter. The value $t = (p-1)/q$ is called the *cofactor*. The specification can be trivially applicable to the groups based on elliptic curves. In elliptic curve systems, G' is the group of points $E(L)$ on an elliptic curve E defined over a finite field L, and G is a subgroup of $E(L)$ of prime order q. For elliptic curve based groups, the cofactor t is typically very small.

Definition 2.1: Letting G be a cyclic group of prime order q generated by an element g, for two elements $X = g^x$, $Y = g^y$ in G, where $x, y \in Z_q$, we denote by $CDH(X, Y) = g^{xy \bmod q} \bmod p$ (the *mod* operation is usually omitted for presentation simplicity). An algorithm is called a CDH solver for G if it takes as input any elements $(X, Y) \in G^2$ (and also the system parameters p, g, q) and its goal is to output the value of $CDH(X, Y)$. We say the computational Diffie-Hellman (CDH) assumption holds in G if for any probabilistic polynomial-time (PPT) CDH solver, the probability that on any pair of random elements $(X = g^x, Y = g^y) \leftarrow G^2$ (i.e., each of x and y is taken uniformly at random from Z_q), the solver computes the correct value $CDG(X, Y)$ is negligible (in k). The probability is taken over the random coins of the solver, and the choice of X, Y uniformly at random in G (and also the random choice of the system parameters (p, g, q)). □

The gap DH assumption (GDH) [41] essentially says that in the group G, computing $CDH(X, Y)$, for $(X, Y) \leftarrow G^2$, is strictly harder than deciding if $Z = CDH(U, V)$ for an arbitrary triple $(U, V, Z) \in G^3$.

Definition 2.2: (Gap Diffie-Hellman (GDH) Assumption [41]): Let G be a cyclic group generated by an element g

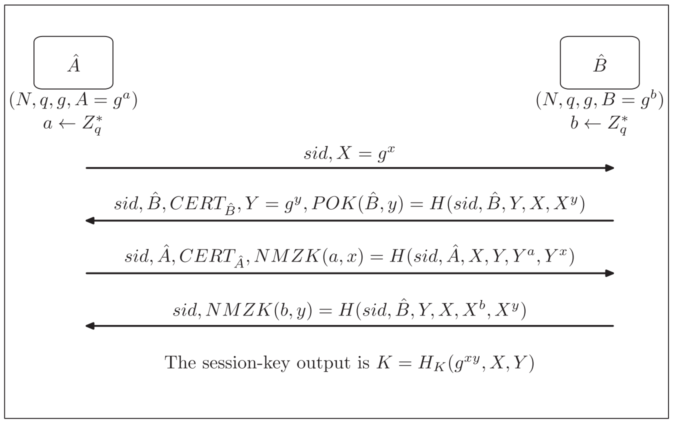
$\hat{A}$
$(N, q, g, A = g^a)$
$a \leftarrow Z_q^*$
$\hat{B}$
$(N, q, g, B = g^b)$
$b \leftarrow Z_q^*$
$sid, X = g^x$
$sid, \hat{B}, CERT_{\hat{B}}, Y = g^y, POK(\hat{B}, y) = H(sid, \hat{B}, Y, X, X^y)$
$sid, \hat{A}, CERT_{\hat{A}}, NMZK(a, x) = H(sid, \hat{A}, X, Y, Y^a, Y^x)$
$sid, NMZK(b, y) = H(sid, \hat{B}, Y, X, X^b, X^y)$
The session-key output is $K = H_K(g^{xy}, X, Y)$

Fig. 1. Deniable Internet key-exchange (the main mode).

of order q, and a decision predicate algorithm $\mathcal{O}$ be a (*full*) Decisional Diffie-Hellman (DDH) Oracle for the group G and generator g such that on input (U, V, Z), for *arbitrary* $(U, V) \in G^2$, oracle $\mathcal{O}$ outputs 1 if and only if $Z = CDH(U, V)$. We say the GDH assumption holds in G if for any PPT CDH solver for G, the probability that on a pair of random elements $(X, Y) \leftarrow G^2$ the solver computes the correct value $CDH(X, Y)$ is negligible (in k), even when the algorithm is provided with the (full) DDH-oracle $\mathcal{O}$. □

III. DIKE Implementation and Advantageous Features

Let $(A = g^a, a)$ (resp., $(X = g^x, x)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of the initiator $\hat{A}$, and $(B = g^b, b)$ (resp., $(Y = g^y, y)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of the responder player $\hat{B}$, where a, x, b, y are taken randomly and independently from Z_q^*. Let $H, H_K : \{0,1\}^* \rightarrow \{0,1\}^{|q|}$ be hash functions, which are modeled as random oracles in security analysis. Here, for presentation simplicity, we have assumed H, H_K are of the same output length. In practice, they may be of different output lengths.

The deniable Internet key-exchange protocol, for the main mode of IKEv2 [27], [29], [30], is depicted in Figure 1 (page 127), where $CERT_{\hat{A}}$ (resp., $CERT_{\hat{B}}$) is the public-key certificate of $\hat{A}$ (resp., $\hat{B}$) issued by some trusted Certificate Authority (CA) within the underlying public-key infrastructure, and sid is the session-identifier that is assumed to be set by some "higher layer" protocol that "calls" the KE protocol and ensures no two sessions run at a party are of identical session-identifier [13]. Throughout this work, we assume no proof-of-knowledge/possession (POK/POP) of secret-key is mandated during public-key registration, but the CA will check the non-identity sub-group (i.e., $G \setminus 1_G$) membership of registered public-keys. Also, each party checks the $G \setminus 1_G$ membership of the DH-component from its peer.

A. Some Advantageous Features of DIKE

The DIKE protocol enjoys remarkable privacy protection for both protocol participants. Note that all the authentic messages, $POK(\hat{B}, y)$ and $NMZK(b, y)$ (resp., $NMZK(a, x)$), from $\hat{B}$ (resp., $\hat{A}$) can be computed *merely* from its peer's DH-exponent x (resp., y) and one's own public messages. Furthermore, one party sends the authentic messages involving its secret-key only after being convinced that its peer does "know" the corresponding DH-exponent. This ensures deniability for *both* of the protocol participants simultaneously. To the best of our knowledge, this is the first provably secure DHKE protocol in the literature that enjoys deniability for both protocol participants simultaneously. Indeed, we can view the authentic message $POK(\hat{B}, y)$ as a proof-of-knowledge (POK) of the DH-exponent y for the DH-component Y sent by $\hat{B}$, that is in turn bounded to the identity $\hat{B}$. In this sense, the first three round message can be viewed as a non-malleable zero-knowledge (NMZK) [22], [25] for proving the *joint* knowledge of both a and x, and the combination of the first, the third and the fourth rounds can be viewed as an NMZK for proving the joint knowledge of both b and y. IKEv2 and SIGMA do not enjoy these privacy properties, due to the underlying signatures used. Note also that the DIKE protocol works in the post-specified-peer setting, and the messages from one party do not bear the information of its peers's ID and public-key.

Besides some hashing operations and the validation of peer's public-key certificate, the player $\hat{A}$ computes (Y^q, Y^a, Y^x) and (X, B^x), and the player $\hat{B}$ computes (X^q, X^b, X^y) and (Y, A^y). Note that the computation of (Y^q, Y^a, Y^x) (resp., (X^q, X^b, X^y)) *in parallel* actually amounts to roughly 1.5 modular exponentiations. The DH-component X (resp., Y) can always be off-line pre-computed by $\hat{A}$ (resp., $\hat{B}$). Moreover, if the peer's identity is pre-specified, $\hat{A}$ (resp., $\hat{B}$) can further off-line pre-compute the value B^x (resp., A^y). That is, the total computation involved at each player side is about 3.5 exponentiations with parallel computation (resp., 5 exponentiations without parallel computation), and the on-line computation involved at each player side can be only 1.5 exponentiations with parallel computation (resp., 3 exponentiations without parallel computation). We note that if the underlying signatures used in SIGMA are implemented with the Digital Signature Standard (DSS) [24], the total computation involved at each player side in SIGMA is about 4.5 exponentiations with the simultaneous exponentiation (SE) technique [37][1] (resp., 5 exponentiations without the SE technique) at each player side in total, and the online computation involved at each player side is about 2.5 exponentiations with SE (resp., 3 exponentiations without SE). For communication complexity, by waiving the use and exchanges of signatures, our deniable IKE enjoys improved communication complexity in comparison with that of SIGMA.

Our DIKE protocol is of well compatibility with IKEv2/SIGMA and the (H)MQV protocols [32], [34]. By compatibility with SIGMA/IKEv2, we mean that in case some players are not of discrete logarithm (DL) public-keys, they still can use the Sign-then-MAC mechanism of

[1]By simultaneous exponentiations, we mean that the product of two exponentiations can be evaluated in about 1.5 exponentiations (rather than 2 separate exponentiations) [37].

SIMGA/IKEv2 to authenticate messages from them. In more details, in this case, any one of the last two messages in our deniable IKE (both for the main mode and for the aggressive mode) can be replaced by the corresponding message flow in SIGMA/IKEv2. By compatibility with (H)MQV, we mean that both (H)MQV and DIKE work for players of DL public-keys, and can be of the same system parameters.

IV. Analysis of DIKE in the Canetti-Krawczyk Post-Specified Peer Framework

A. Brief Description of the CK-Framework

The following brief description of the CK-framework is almost verbatim from [32] (for full details the reader is referred to [12], [13]).

A key-exchange (KE) protocol is run in a network of interconnected parties where each party can be activated to run an instance of the protocol called a session. Within a session a party can be activated to initiate the session or to respond to an incoming message. As a result of these activations, and according to the specification of the protocol, the party creates and maintains a session state, generates outgoing messages, and eventually completes the session by outputting a session-key and erasing the session state. A session may also be aborted without generating a session key. A session may also be expired, and for an expired session the session-key is also erased. A KE session is associated with its holder or owner (the party at which the session exists), a peer (the party with which the session key is intended to be established), and a unique session identifier. For presentation simplicity, for DHKE protocols, the following two assumptions are made in the basic CK-framework: (i) the activation of a session at a party always specifies the name of the intended peer (i.e., working in the "pre-specified peer model"); and (ii) a session is defined by a tuple $(\hat{A}, \hat{B}, X, Y)$, where $\hat{A}$ is the identity of the holder of the session who sends the DH-component X, $\hat{B}$ the peer who sends the DH-component Y. The peer that sends the first message in a session is called the initiator and the other the responder. Usually, the peers to a session are denoted by $\hat{A}$ and $\hat{B}$; either one may act as initiator or responder. The session $(\hat{B}, \hat{A}, Y, X)$ (if it exists) is said to be matching to the session $(\hat{A}, \hat{B}, X, Y)$.

A note on session identifiers in reality. It is suggested by Canetti et al. in [12] that the application invoking KE protocol instances should supply the unique session identifier sid for each session, and the setting mechanism of session identifiers is beyond the CK-framework. In practice and particularly in IKEv2, two participants first exchange random nonces prior to the actual protocol session run, and then use the concatenation of these nonces as the corresponding session identifier. This is a common approach, as suggested by Choo et al. in [14], for setting session identifiers for two-party protocols like IKEv2 and the DIKE protocol developed in this work. An alternative would be to use an externally generated SID, such as a counter, but the use of such an SID would be inconvenient [14]. However, when it comes to multi-party protocols (e.g., group key exchange) *without broadcasting messages*, the situation is not so clear, as demonstrated by Choo et al in [14]. The reader is referred to [14] for the subtleties of setting up unique session identifiers for group KE in reality.

Attacker model. The attacker, denoted $\mathscr{A}$, is an active concurrent man-in-the-middle (CMIM) adversary with full control of the communication links between parties. $\mathscr{A}$ can intercept and modify messages sent over these links, and can delay or prevent their delivery, inject its own messages, interleave messages from different sessions, etc. (Formally, it is $\mathscr{A}$ to whom parties hand their outgoing messages for delivery.) $\mathscr{A}$ also schedules all session activations and session-message delivery, and can register a list of public-keys (for players already controlled by $\mathscr{A}$ at the onset of its attack).

In addition, in order to model potential disclosure of secret information, the attacker is allowed to have access to secret information via session exposure attacks of four types: state-reveal queries, session-key queries, secret-key queries, and party corruptions. A state-reveal query is directed at a single session while still incomplete (i.e., before outputting the session key), and its result is that the attacker learns the session state for that particular session (which may include, for example, the DH-exponent corresponding to the DH-component). A session-key query can be performed against an individual session after completion but before expiration, and the result is that the attacker learns the corresponding session-key. A secret-key query can be performed against any uncorrupted party, and the result is that the attacker learns the static secret-key of that party. Finally, party corruption means that the attacker learns *all* information in the memory of that party, including the long-term static secret-key (corresponding to the public-key) as well as session states and session keys stored at the party; in addition, from the moment a party is corrupted all its actions may be controlled by the attacker.[2] A session is called exposed, if this session or its matching session suffers from *any* of the above four types of session-exposure attacks.

The security of session keys generated in unexposed sessions is captured via the inability of the attacker $\mathscr{A}$ to distinguish the session key of a test session, chosen by $\mathscr{A}$ among all *complete* sessions of the protocol between *uncorrupted* players, from a random value. When $\mathscr{A}$ chooses the test session it is provided with a value v which is chosen as follows: Firstly, a random bit b is tossed. If $b = 0$ then v is the real value of the session key, otherwise v is a random value chosen under the same distribution of session keys produced by the protocol but independent of the value of the real session key. After receiving v the attacker may continue with the regular actions against the protocol; at the end of its run $\mathscr{A}$ outputs a bit b'. The attacker succeeds in its distinguishing attack if (1) the test-session is not exposed, and (2) the probability that $b = b'$ is non-negligibly larger than 1/2.

Definition 4.1 (Secure KE (SK-security)): A polynomial-time attacker with the above capabilities is called a

[2]We remark that, the basic CK-framework does not distinguish between static secret-key reveal and party corruption. In this sense, our actual formalization is w.r.t. an enhanced version of the CK-framework.

KE-attacker. A key-exchange protocol π is called secure [12] if for any KE-attacker $\mathcal{A}$ running against π it holds:

1) If two uncorrupted parties complete matching sessions in a run of protocol π under attacker $\mathcal{A}$ then, except for a negligible probability, the session key output in these sessions is the same.
2) $\mathcal{A}$ succeeds (in its test-session distinguishing attack) with probability not more than that 1/2 plus a negligible function. □

Adapting SK-security to the post-specified peer setting. Recall that the CK-framework [12] assumes: a party that is activated with a new session knows already at activation the identity of the intended peer to the session. By contrast, in the "post-specified peer" setting (particularly for the IPsec and IKE protocols [27], [29], [30]), the information of who the other party is does not necessarily exist at the session initiation stage. It is actually learnt by the party only after the protocol run evolves. Fortunately, as shown by Canetti et al. in [13], adapting the SK-security [12] to the post-specified peer setting only requires some minor modifications (related to the mechanism of defining matching sessions): (1) To distinguish concurrent sessions (run at each party), each session bears a unique session-identifier sid; (2) Supposing $(\hat{A}, sid, \hat{B})$ be a completed session (at the party $\hat{A}$) with peer $\hat{B}$, the session $(\hat{B}, sid)$ is called the matching session of $(\hat{A}, sid, \hat{B})$, if either $(\hat{B}, sid)$ is not completed or $(\hat{B}, sid)$ is completed with peer $\hat{A}$. For more details, the reader is referred to [13].

B. SK-Security Analysis of DIKE With Post-Specified Peers

We consider a version of the DIKE protocol (depicted in Figure 1) with *exposable DH-exponents and pre-computed DH-components*. Specifically, for a state-reveal query against an incomplete session at a party, the value returned to the attacker $\mathcal{A}$ is specified to be the corresponding DH-exponent generated by the party for that session. Furthermore, we assume each uncorrupted player pre-computes (and stores) DH-components, which can be exposed to the attacker prior to the actual sessions in which these pre-computed DH-components are to be used.

In this section we prove that the DIKE protocol *with exposable DH-exponents and pre-computed DH-components* is SK-secure in the CK-framework with post-specified peers, under the gap Diffie-Hellman (GDH) assumption in the random oracle model. At the core of the proof is the following lemma, which essentially says that the attacker can successfully finish an *unexposed* session in the name of some uncorrupted player only if that uncorrupted player (impersonated by the attacker) does indeed send the authenticated value, say, $NMZK(a, x)$ or $NMZK(b, y)$, in the corresponding matching session. In the following analysis, we denote by $\widetilde{NMZK}$ or $\widetilde{POK}$ the authentic messages sent by the attacker, and by $NMZK$ or POK the authentic message sent by an uncorrupted player.

Lemma 4.1: For the DIKE protocol $\langle \hat{A}, \hat{B} \rangle$ (depicted in Figure 1) *with pre-computed and exposed DH components and exponents*, where the players $\hat{A}$ and $\hat{B}$ may be the same, the probability of the following events is negligible under the GDH assumption in the random oracle model assuming that $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$ is a RO (recall that $k = |q|$):

- **Event-1.** The attacker $\mathcal{A}$ successfully finishes an *unexposed* session of session identifier sid with an uncorrupted player $\hat{B}$ in the name of uncorrupted player $\hat{A}$ (actually, any uncorrupted player), where the session is referred to as $(\hat{B}, sid, \hat{A})$, in which $\hat{B}$ sends $Y = g^y$ in the second round and $\mathcal{A}$ sends $\widetilde{NMZK}(a, x)$ in the third round of this session, while the uncorrupted player $\hat{A}$ did not send $\widetilde{NMZK}(a, x)$ in any session.
- **Event-2.** The attacker $\mathcal{A}$ successfully finishes an *unexposed* session of session identifier sid with an uncorrupted player $\hat{A}$ in the name of uncorrupted player $\hat{B}$ (actually, any uncorrupted player), where the session is referred to as $(\hat{A}, sid, \hat{B})$, in which $\hat{A}$ sends $X = g^x$ in the first round and $\mathcal{A}$ sends $\widetilde{NMZK}(b, y)$ in the fourth round, while the uncorrupted player $\hat{B}$ did not send $\widetilde{NMZK}(b, y)$ in any session.

Notice that, for Lemma 4.1, we only assume H to be a random oracle. In contrast, the function H_K can be any secure key derivation function (KDF).

Proof (of Lemma 4.1). In the following security analysis, for presentation simplicity and without loss of generality, we assume $\mathcal{A}$ is involved with at most s many sessions, and simply assume the session-identifier $sid \in \{1, \cdots, s\}$, where s is polynomial in $k = |q|$.

Proof of Event-1. Specifically, supposing Event-1 occurs with non-negligible probability, we construct another efficient algorithm S' that, on inputs $(A, Y) \leftarrow G^2$ (where each of (A, Y) is taken randomly and independently from G) and with access to the DDH-oracles $\mathcal{O}_A$ and $\mathcal{O}_Y$, outputs $CDH(A, Y)$ also with non-negligible probability, which violates the GDH assumption.[3] Here, $\mathcal{O}_A$ (resp., $\mathcal{O}_Y$) stands for a special DDH-oracle, which on queries of the form $(A, \gamma, \delta) \in G^3$ (resp., $(Y, \gamma, \delta) \in G^3$) outputs whether $\delta = CDH(A, \gamma)$ (resp., $\delta = CDH(Y, \gamma)$) or not. The algorithm S' runs the attacker $\mathcal{A}$ as a subroutine, and works as follows:

S' generates and sets the public-key and secret-key pairs for all uncorrupted players except the uncorrupted player $\hat{A}$. For the uncorrupted player $\hat{A}$, S' just sets its public-key to be A (i.e., the value given to S' as input). Note that S' does not know the secret-key of $\hat{A}$, i.e., the discrete logarithm of A. S' chooses j uniformly at random from $\{1, \cdots, s\}$, and sets the DH-component to be sent in the second round of the j-th session to be Y (i.e., the value given to S' as input), while DH-components (to be generated by uncorrupted players) in all other sessions will be generated by S' itself. In the session run, in case $\mathcal{A}$ makes a query (of the four types of session-

[3]The input of the public-key A to S' is for presentation simplicity. In general, the input to S' should be the certificate $CERT_{\hat{A}}$ of the uncorrupted player $\hat{A}$ (besides the randomly chosen Y). Notice that we are assuming a very weak system setup assumption, where POP/POK of secret-key is not mandated during public-key registration, which is an advantageous property of our protocol. In this case, from public-key A the simulator S' can actually get the certificate $CERT_{\hat{A}}$ by himself (in the name of $\hat{A}$). An alternative approach is to let S' incorporate the secret-key of the certificate authority (CA) and generate players' certificates by S' itself. With this approach, S' is still of polynomial time, violating the GDH assumption.

exposure attacks) against the j-th session, or the j-th session is not of the form $(\hat{B}, j, \hat{A})$ for some uncorrupted player $\hat{B}$, S' just aborts (recall we have assumed that $sid \in \{1, \cdots, s\}$). However, all the queries made by $\mathscr{A}$ against any session other than the j-th session will be answered by S'. Specifically, S' can give the attacker $\mathcal{A}$ the secret-keys of all uncorrupted players *other than the uncorrupted player* $\hat{A}$, and all the (pre-computed) DH-components and DH-exponents to be used in all sessions other than the chosen j-th session. Also, as S' generates DH-exponents and DH-components honestly in all the sessions other than the j-th session (that is supposed to be unexposed), it can answer session-key queries against these sessions, no matter what the key derivation function H_K is.

For each session $(\hat{A}, i, \hat{B})$ run by uncorrupted $\hat{A}$, where $1 \leq i \neq j \leq s$ and $\hat{A}$ plays the session initiator and the attacker $\mathscr{A}$ plays the role of the session responder in the name of $\hat{B}$ in this (say, the i-th) session, S' works just as the honest player $\hat{A}$ does in the first, the second and the fourth round (by using the DH-exponent x_i generated by itself for this session). But in the third round of this session (that is dependent on the secret-key a of $\hat{A}$ which S', however, does not know), S' simulates as follows. Denote by $X_i = g^{x_i}$ the DH-component sent by S' in the first round, and by Y_i the DH-component sent by the attacker $\mathscr{A}$ in the second round in the name of $\hat{B}$. S' ensures the consistency of RO-answers related to $NMZK(a, x_i) = H(i, \hat{A}, X_i, Y_i, CDH(A, Y_i), CDH(X_i, Y_i))$, with the assistance of the DDH-oracle $\mathcal{O}_A$, as follows (recall that x_i is generated by S' itself): S' first checks whether the RO, H, has been previously defined on the point $(i, \hat{A}, X_i, Y_i, CDH(A, Y_i), CDH(X_i, Y_i))$. This is done by checking all RO-queries to H of the form $(i, \hat{A}, X_i, Y_i, \alpha_i, Y_i^{x_i})$ for some $\alpha_i \in G$, and testing whether $\alpha_i = CDH(A, Y_i)$ with the assistance of the DDH-oracle $\mathcal{O}_A$. If H has been defined on the point $(i, \hat{A}, X_i, Y_i, CDH(A, Y_i), CDH(X_i, Y_i))$, then S' just sets $NMZK(a, x_i)$ to be the already defined value $H(i, \hat{A}, X_i, Y_i, CDH(A, Y_i), CDH(X_i, Y_i))$; otherwise, S' just sets $NMZK(a, x_i)$ to be a value taken uniformly at random from $\{0, 1\}^k$. S' sends $\{i, \hat{A}, CERT_{\hat{A}}, NMZK(a, x_i)\}$ to $\mathscr{A}$ as the third-round message of the i-th session. In case that $NMZK(a, x_i)$ is set to be a random value in $\{0, 1\}^k$, from this point on, whenever the attacker makes a RO-query to H of the form $(i, \hat{A}, X_i, Y_i, \alpha'_i, (Y_i)^{x_i})$ for some $\alpha'_i \in G$, S' checks whether $\alpha'_i = CDH(A, Y_i)$ with the assistance of the DDH-oracle $\mathcal{O}_A$. If this is the case, S' just defines the RO answer, i.e., $H(i, \hat{A}, X_i, Y_i, CDH(A, Y_i), CDH(X_i, Y_i))$, to be the already fixed $NMZK(a, x_i)$ (i.e., the random value in $\{0, 1\}^k$ set previously).

For each session $(\hat{B}, i, \hat{A})$ run by uncorrupted $\hat{B}$, where $1 \leq i \neq j \leq s$ and $\hat{B}$ plays the session responder and the attacker $\mathscr{A}$ plays the role of the session initiator in the name of $\hat{A}$, we further distinguish two cases according to whether $\hat{B} = \hat{A}$ or not. For the case of $\hat{B} \neq \hat{A}$, as the secret-key of $\hat{B}$ is set by S', S' can perfectly simulate this session just as the honest player $\hat{B}$ does. Below, we focus on the case of $\hat{B} = \hat{A}$, where the simulator S' does not know the secret-key $b = a$ of $\hat{B} = \hat{A}$. For this case, S' works just as the honest player $\hat{B}$ does in the first three rounds of this session, by using the DH-exponent y_i generated by S' itself for this session. Denote by X_i the DH-component sent by $\mathscr{A}$ in the first round of this session, and recall that $\hat{A} = \hat{B}$ and $A = B$ for this case. But in the fourth round of this session, S' needs to send the authentication value $NMZK(b, y_i) = NMZK(a, y_i) = H(i, \hat{B}, Y_i, X_i, CDH(A, X_i), X_i^{y_i})$ while S' actually does not know the secret-key a. S' ensures the consistency of the RO answers of H related to the authentication value $NMZK(b, y_i)$ with the assistance of the DDH-oracle $\mathcal{O}_A$, which is similar to the simulation of the session $(\hat{A}, i, \hat{B})$ as specified above.

For the session $(\hat{B}, j, \hat{A})$ (recall that if the j-th session is not of this form or $\mathscr{A}$ made any exposure query against the j-th session, S' simply aborts), S' works as follows. After receiving (j, X_j) from $\mathscr{A}$ in the first round of the j-th session, in the second round of the j-th session S' just sets the DH-component Y_j to be the value Y given as its input. Note that the discrete logarithm y_j of $Y_j = Y$ is unknown to S'. Then, S' sets the value $POK(\hat{B}, y_j)$ and ensures the consistency of the RO-answers of H related to $POK(\hat{B}, y_j) = H(j, \hat{B}, Y, X_j, CDH(Y, X_j))$, *with the assistance of the DDH-oracle* $\mathcal{O}_Y$ similar to the simulation of the session $(\hat{A}, i, \hat{B})$ above. Specifically, with the assistance of $\mathcal{O}_Y$, S' first checks whether the random oracle H has been defined on the point $(j, \hat{B}, Y, X_j, CDH(Y, X_j))$ or not. If yes, S' simply sets $POK(\hat{B}, y_j)$ to be the already defined value. Otherwise, S' sets $POK(\hat{B}, y_j)$ to be a value taken uniformly at random from $\{0, 1\}^k$, and from then on for all the RO queries made by $\mathscr{A}$, S' ensures the consistency of RO-answers related to $POK(\hat{B}, y_j)$ with the assistance of the DDH-oracle $\mathcal{O}_Y$. Finally, if $\mathscr{A}$ successfully sends $(j, \hat{A}, CERT_{\hat{A}}, \widetilde{NMZK}(a, x_j))$ in the third round of the j-th session, S' checks whether $\mathscr{A}$ queried the RO, H, with $(j, \hat{A}, X_j, Y, CDH(Y, A), CDH(Y, X_j))$. That is, for each RO query by $\mathscr{A}$ to H of the form $(j, \hat{A}, X_j, Y, \alpha, \beta)$, where $(\alpha, \beta) \in G^2$, S' checks whether $\alpha = CDH(Y, A)$ with the assistance of the DDH-oracle $\mathcal{O}_Y$. If so, S' stops and outputs $CDH(A, Y)$. In all other cases, S' aborts.

It is straightforward to calculate that, supposing the attacker $\mathscr{A}$ made q_H queries to the random oracle H, the number of queries by S' to its DDH-oracle $\mathcal{O}_A$ or $\mathcal{O}_Y$ is at most q_H. Specifically, all the RO queries made by $\mathscr{A}$ to H can be classified into disjoint subsets according to their leading items that correspond to different session-identifiers. For each RO query in each classified subset, S' makes at most one DDH-test according to the specification of S'. As we assume $\mathscr{A}$ runs in polynomial time, we have that S' works in polynomial time. It is easy to see that, conditioning on S' correctly guessed the target unexposed session $(\hat{B}, j, \hat{A})$, the simulation by S' is perfect from the view of $\mathscr{A}$. In particular, the attacker $\mathscr{A}$ is disallowed to issue a state-reveal query to learn the discrete logarithm y of Y. Suppose $\widetilde{\text{Event-1}}$ occurs, i.e., $\mathscr{A}$ successfully sent the correct value $\widetilde{NMZK}(a, x_j)$ in the j-th session, which, however, was not sent by the uncorrupted player $\hat{A}$ in any session. Up to the third round of the j-th session, the correct value of $\widetilde{NMZK}(a, x_j) = H(j, \hat{A}, X_j, Y, CDH(Y, A), CDH(Y, X_j))$ was actually not sent by any uncorrupted player, and thus the simulator S', in any session (particularly due to the uniqueness of the session-identifier of each session). In this case, to send a correct $\widetilde{NMZK}(a, x)$ in the third round of the j-th session, in the RO model except for probability $\frac{1}{2^k}$ $\mathscr{A}$ must have queried the RO H with $(j, \hat{A}, X_j, Y, CDH(Y, A), CDH(Y, X_j))$. We conclude that, supposing Event-1 occurs with non-negligible probability p, with also non-negligible probability at least $\frac{p}{s} - \frac{1}{2^k}$ S' correctly computes $CDH(A, Y)$, which violates the GDH assumption.

Proof of Event-2. The proof for Event-2 is similar to, but *more involved than*, that for Event-1. Specifically, for Event-2, the correct value of $\widetilde{NMZK}(b, y)$ should be $H(sid, \hat{B}, Y, X, CDH(B, X), CDH(Y, X))$, which cannot be sent by the uncorrupted player $\hat{A}$ (even if $\hat{A} = \hat{B}$) in any other session (due to the uniqueness of the session-identifiers of the left-sessions). However, supposing the attacker $\mathscr{A}$ sets $\hat{B} = \hat{A}$ and $Y = X$, it may be the case that $\widetilde{NMZK}(b, y) = \widetilde{NMZK}(a, x)$ could just be sent by $\hat{A}$ in the third round of the unexposed target session $(\hat{A}, sid, \hat{B})$. In the following analysis for Event-2, we further distinguish two cases according to whether $Y = X$ or not in the target session.

Specifically, supposing Event-2 occurs with non-negligible probability, we construct another efficient algorithm S' that, on inputs $(B, X) \leftarrow G^2$ (where each of (B, X) is taken uniformly at random from G) and with access to the DDH-oracles $\mathcal{O}_B$ and $\mathcal{O}_X$, outputs $CDH(B, X)$ or $CDH(X, X)$ also with non-negligible probability, which violates the GDH assumption. Here, we note that the hardness of computing $CDH(X, X)$ from $X \leftarrow G$ is equivalent to that of the CDH problem [36], [38]. Again, $\mathcal{O}_B$ (resp., $\mathcal{O}_X$) stands for a special DDH-oracle, which on queries of the form $(B, \gamma, \delta) \in G^3$ (resp., $(X, \gamma, \delta) \in G^3$) outputs whether $\delta = CDH(B, \gamma)$ (resp., $\delta = CDH(X, \gamma)$). The algorithm S' runs the attacker $\mathscr{A}$ as a subroutine, and works as follows.

S' generates and sets the public-key and secret-key pairs for all uncorrupted players other than the uncorrupted player $\hat{B}$. For the uncorrupted player $\hat{B}$, S' just sets its public-key to be B (i.e., the value given to S' as input). S' chooses j uniformly at random from $\{1, \cdots, s\}$, and sets the DH-component to be sent in the first round of the j-th session to be X (i.e., the value given to S' as input), while DH-components (to be generated by uncorrupted players) in all other sessions will be generated by S' itself. In particular, it implies that S' can also answer session-key queries against all the sessions other than the j-th session (that is assumed to be unexposed), no matter what the key derivation function H_K is. In the run, in case $\mathscr{A}$ makes a query (of the four types of session-exposure attacks) against the j-th session, or the j-session is not of the form $(\hat{A}, j, \hat{B})$ between two uncorrupted players $\hat{A}$ and $\hat{B}$, S' just aborts. However, all the queries made by $\mathscr{A}$ against any session other than the j-th session can be perfectly answered by S'.

For each session $(\hat{A}, i, \hat{B})$ run by uncorrupted $\hat{A}$, where $1 \leq i \neq j \leq s$ and $\hat{A}$ plays the session initiator and the attacker $\mathscr{A}$ plays the role of the session responder in the name of $\hat{B}$ in this (say, the i-th) session, we further distinguish two cases according to whether $\hat{A} = \hat{B}$ or not. For the case of $\hat{A} \neq \hat{B}$, as the secret-key of $\hat{A}$ is just set by S' itself, S' can perfectly simulate this session just as the honest player $\hat{A}$ does. Below, we focus on the case of $\hat{A} = \hat{B}$, where the simulator S' does not know the secret-key $a = b$ of $\hat{A} = \hat{B}$. For this case, S' works just as the honest player $\hat{A}$ does in the first, the second and the fourth rounds, by using the DH-exponent x_i generated by S' itself for this session. Denote by Y_i the DH-component sent by $\mathscr{A}$ in the second round of this session, and recall that $\hat{A} = \hat{B}$ and $A = B$. But in the third round of this session, S' needs to send the authentication value $NMZK(a, x_i) = NMZK(b, x_i) = H(i, \hat{A}, X_i, Y_i, CDH(B, Y_i), Y_i^{x_i})$ while S' actually does not know the secret-key b. S' ensures the consistency of the RO answers of H related to the authentication value $NMZK(b, x_i)$ with the assistance of the DDH-oracle $\mathcal{O}_B$, similar to the simulation of the session $(\hat{A}, i, \hat{B})$ in the proof of Event-1.

For each session $(\hat{B}, i, \hat{A})$ run by uncorrupted $\hat{B}$, where $1 \leq i \neq j \leq s$ and $\hat{B}$ plays the session responder and the attacker $\mathscr{A}$ plays the role of the session initiator in the name of $\hat{A}$, S' works just as the honest player $\hat{B}$ does in the first three rounds (by using the DH-exponent y_i generated by itself for this session). But in the fourth round of the this session (that is dependent on the secret-key b of $\hat{B}$ which S', however, does not know), S' simulates as follows. Denote by $X_i = g^{x_i}$ the DH-component sent by the attacker $\mathscr{A}$ in the first round (in the name of $\hat{A}$), and by $Y_i = g^{y_i}$ the DH-component sent by S' in the second round. Similarly, S' can ensure the consistency of RO-answers related to $NMZK(b, y_i) = H(i, \hat{B}, Y_i, X_i, CDH(B, X_i), X_i^{y_i})$, with the assistance of the DDH-oracle denoted $\mathcal{O}_B$.

For the session $(\hat{A}, j, \hat{B})$ (recall that if the j-th session is not of this form or $\mathscr{A}$ made any exposure query against the j-th session, S' simply aborts), S' just sets the DH-component to be sent in the first round to be X (i.e., the value given as its input). Denote by Y_j the DH-component sent by the attacker $\mathscr{A}$ in the second round (in the name of $\hat{B}$), we further distinguish two cases according to whether $Y_j = X$ or not.

Case-1: $Y_j = X$**.** For this case, if $\mathscr{A}$ successfully sends $(j, \hat{B}, CERT_{\hat{B}}, \widetilde{POK}(\hat{B}, y_j))$ in the second round of the j-th session, S' checks whether $\mathscr{A}$ has queried the RO, H, with $(j, \hat{B}, X, X, CDH(X, X))$. In more detail, for all RO-queries of the form $(j, \hat{B}, X, X, \alpha)$, where $\alpha \in G$, made by $\mathscr{A}$ to H, S' checks whether $\alpha = CDH(X, X)$ with the assistance of the DDH-oracle $\mathcal{O}_X$. If so, S' stops and outputs $CDH(X, X)$. In all other cases, S' aborts. Note that the correct value of $\widetilde{POK}(\hat{B}, y_j)$ in this case should just be $(j, \hat{B}, X, X, CDH(X, X))$.

Case-2: $Y_j \neq X$**.** For this case, the correct value of $\widetilde{POK}(\hat{B}, y_j)$ should be $(j, \hat{B}, Y_j, X, CDH(X, Y_j))$. After receiving $\widetilde{POK}(\hat{B}, y_j)$, similar to Case-1, S' checks whether $\mathscr{A}$ has queried the RO, H, with $(j, \hat{B}, Y_j, X, CDH(X, Y_j))$, with the assistance of the DDH oracle $\mathcal{O}_X$. If not, S' simply aborts; otherwise, S' gets the value $CDH(X, Y_j)$. After getting $CDH(X, Y_j)$, and as S' sets the secret-key a of $\hat{A}$, S' can perfectly simulate the third round message.

Finally, if $\mathscr{A}$ successfully sends $(j, \hat{B}, \widetilde{NMZK}(b, y_j))$ in the fourth round of the j-th session, for all RO-queries of the form $(j, \hat{B}, Y_j, X, \alpha, \beta)$, where $(\alpha, \beta) \in G^2$, made by $\mathscr{A}$ to H, S' checks whether $\alpha = CDH(B, X)$ with the DDH-oracle $\mathcal{O}_X$. If so, S' stops and outputs $CDH(B, X)$. In all other cases, S' aborts.

It's easy to see that S' works in polynomial time. In particular, supposing the attacker $\mathscr{A}$ makes q_H queries to the random oracle H, the number of queries by S' to its DDH-oracle $\mathcal{O}_B$ or $\mathcal{O}_X$ is at most q_H. Conditioning on S' correctly guessed the target unexposed session $(\hat{B}, j, \hat{A})$, the simulation by S' is perfect from the view of $\mathscr{A}$. We have that, supposing Event-2 occurs with non-negligible probability p, then with also non-negligible probability at least $\frac{p}{s} - \frac{1}{2^k}$ S' correctly computes $CDH(X, X)$ or $CDH(B, X)$, which violates the GDH assumption, where $\frac{1}{2^k}$ is the probability that $\mathscr{A}$ correctly sends $\widetilde{POK}(\hat{B}, y_j)$ or $\widetilde{POK}(b, y_j)$ without making the corresponding RO query. □

Theorem 4.1: The DIKE protocol (depicted in Figure 1), *with pre-computed and exposed DH components and exponents*, is SK-secure in the CK-framework with post-specified peers, under the GDH assumption in the random oracle model, where H is assumed to be a programmable random oracle while H_K to be a *non-programmable* RO.

Proof (outline). We present the proof with respect to an *unexposed* and successfully finished test-session run by $\hat{A}$ (with peer $\hat{B}$), denoted $(\hat{B}, j, \hat{A})$, where $1 \leq j \leq s$ and $\hat{A}$ and $\hat{B}$ are both uncorrupted players and *can be identical* (but $\hat{A}$ may be impersonated by the attacker $\mathscr{A}$). The proof for the case of an unexposed test-session $(\hat{A}, j, \hat{B})$ run by $\hat{A}$ (with peer $\hat{B}$) is similar.

For the unexposed test-session $(\hat{B}, j, \hat{A})$, denote by $X = g^x$ (resp, $Y = g^y$) the DH-component sent by $\hat{A}$ (resp., $\hat{B}$), and by $NMZK(a, x) = H(j, \hat{A}, X, Y, CDH(A, Y), CDH(X, Y))$ the authentication value sent by $\hat{A}$ (maybe impersonated by $\mathscr{A}$) in the third round of the test-session. By Lemma 4.1 (for Event-1), we have that, with overwhelming probability, the uncorrupted player $\hat{A}$ does indeed send $NMZK(a, x)$ in some session that is matching to $(\hat{B}, j, \hat{A})$. That is, the test-session has the matching session $(\hat{A}, j)$ in which $\hat{A}$ sends X in the first round and $NMZK(a, x)$ in the third round.

As the session-key is computed as $H_K(X, Y, g^{xy})$ where H_K is a (non-programmable) random oracle, there are only two possible strategies for the adversary $\mathscr{A}$ to distinguish $H_K(X, Y, g^{xy})$ from a random value:

- **Key-replication attack:** $\mathscr{A}$ succeeds in forcing the establishment of a session (other than the test-session or its matching session) that has the same session-key output as the test-session. In this case, $\mathscr{A}$ can learn the session-key of test-session by simply querying that unmatching session to get the same key (without having to expose the test-session or its matching session).
- **Forging attack:** At some point in its run, $\mathscr{A}$ queries the RO, H_K, with the values (X, Y, g^{xy}).

The possibility of the key-replication attack is trivially ruled out with overwhelming probability in the RO model, by observing that X is only sent by $\hat{A}$ in the test-session and that Y is only sent by $\hat{B}$ in the matching session in accordance with Lemma 4.1. Recall that the session-key of the text-session, as well as that of its matching session, is $H_K(X, Y, CDH(X, Y))$.

The success of the forging attack says that $\mathscr{A}$ can successfully output $(X, Y, CDH(X, Y))$. By Lemma 4.1, with overwhelming probability, X and Y are only sent *by the uncorrupted players* in the test-session and its matching session. As both the test-session and its matching session are assumed to be unexposed, $\mathscr{A}$ does not know the DH-exponent x or y. Similar to the proof of Lemma 4.1, we can exploit the assumed ability of $\mathscr{A}$ in performing the successful forging attack to construct another efficient algorithm S' who breaks the CDH assumption with the assistance of a DDH-oracle $\mathcal{O}$ (actually, $\mathcal{O}$ is only used to deal with queries including X or Y).

Specifically, on inputs (X, Y) and with access to the DDH-oracle $\mathcal{O}$, S' runs $\mathscr{A}$ as a subroutine and does the following: (1) S' sets up the public-keys and secret-keys for all uncorrupted players, including the players $\hat{A}$ and $\hat{B}$. (2) Then, S' guesses (with non-negligible success probability $\frac{1}{s}$) the test-session $(\hat{B}, j, \hat{A})$ and its matching session $(\hat{A}, j)$, for which the forging attack succeeds. (3) S' perfectly emulates the uncorrupted players, and answers any query made by the adversary $\mathscr{A}$, in all the sessions other than the test-session and its matching session. (4) S' sends X in the first round of the matching session $(\hat{A}, j)$ (in the name of $\hat{A}$), and Y in the second round of the test-session (in the name of $\hat{B}$). For all the authentic messages (exchanged in the test-session and its matching session) with $CDH(X, Y)$ as an input, similar to the proof of Lemma 4.1, S' ensures the consistency of RO answers with the assistance of the DDH-oracle $\mathcal{O}$. In particular, if before the completion of the test-session or its matching session, $\hat{A}$ has already queried the random oracle H with input $CDH(X, Y)$, S' simply outputs $CDH(X, Y)$ and stops. Otherwise, after the completion of the test-session, supposing the forging attack is successful, S' outputs $CDH(X, Y)$ from the query made by $\mathscr{A}$ to the non-programmable random oracle H_K. □

A note on the key derivation function H_K**.** For presentation simplicity, we have assumed H and H_K can be the same random oracle. But the actual uses of H and H_K are quite different in the security analysis. Specifically, we actually do *not* make any assumption on the key derivation function (KDF), H_K, in the proof of Lemma 4.1 as well as in the CNMSZK analysis (cf. Theorem 6.1 in Section VI-D). We only assume H_K to be a *non-programmable* RO in the proof of Theorem 4.1. However, a closer investigation shows that, the RO assumption on H_K is not essential in the proof of Theorem 4.1. What we actually need is the following property of H_K, which is relatively reasonable and is much weaker than the RO assumption.

- Strong pseudorandomness. Specifically, given $(X, Y) \leftarrow G^2$, the distribution of $H_K(g^{xy}, X, Y)$ and the uniform distribution U_k over $\{0, 1\}^k$ are computationally indistinguishable, by any efficient distinguisher algorithm *equipped with a full DDH-oracle*.

In practice, we can instantiate H_K by using a pseudorandom function (PRF), e.g., HMMC, with g^{xy} as the randomness seed (i.e., the key of PRF). That is, $H_K(g^{xy}, X, Y) = PRF_{g^{xy}}(X, Y)$. Notice that, if the distinguisher algorithm is not equipped with DDH oracles, the indistinguishability between $PRF_{g^{xy}}(X, Y)$ and U_k can be established as follows. By the DDH assumption, the distribution of $PRF_{g^{xy}}(X, Y)$ and that of $PRF_R(X, Y)$ are indistinguishable, where R is an element taken uniformly at random from G. Then, according to the pseudorandomness of PRF, the distribution of $PRF_R(X, Y)$ and that of U_k are computationally indistinguishable. However, if the distinguisher is equipped with DDH oracles, assuming the indistinguishability between $PRF_{g^{xy}}(X, Y)$ and U_k is non-standard, though it still seems reasonable. In particular, the strong pseudorandomness assumption is stronger than the GDH assumption.

We briefly notice that the proof of Theorem 4.1 can be directly reduced to the strong pseudorandomness assumption of H_K. The proof is similar to that of ruling out the "forging attack" in Theorem 4.1. Specifically, we consider an algorithm S'' whose task, on input (X, Y, v) where v is $H_K(g^{xy}, X, Y)$ or a random value in $\{0, 1\}^k$, is to distinguish which case the given value v is. S'' mimics the algorithm S' in Theorem 4.1 (and is thus equipped with DDH-oracles), with the following modifications. If S', and thus S'', has already gotten $CDH(X, Y)$ from the RO-queries to H by the attacker $\mathscr{A}$ before the completion of the test-session or its matching session, S'' just uses $CDH(X, Y)$ to distinguish v. Otherwise, S'' presents v to the attacker $\mathscr{A}$, and then uses the assumed ability of $\mathscr{A}$ (in distinguishing $H_K(g^{xy}, X, Y)$ from a random value) to finish its task.

V. Discussions about Resistance Against Some Concrete Attacks

In this section, we further discuss a list of concrete yet essential security properties of the DIKE protocol (depicted in Figure 1), *most of which are beyond the CK-framework.*

Resistance against "cutting-last-message" attack. Supposing the responder player $\hat{B}$ sends the last message in the run of a DHKE protocol $\langle \hat{A}, \hat{B} \rangle$, the "cutting-last-message" attack, from which SIGMA and IKEv2 suffer, works as follows [35]: A man-in-the-middle attacker $\mathscr{A}$ interacts with the uncorrupted $\hat{B}$ in the name of $\hat{A}$ in a session (referred to as the test-session), while concurrently interacting with the uncorrupted $\hat{A}$ in the name of $\hat{M} \neq \hat{B}$ in another session (referred to as the matching session). $\mathscr{A}$ relays messages between $\hat{A}$ and $\hat{B}$ in these two sessions, but aborts the matching session after receiving the last message from $\hat{B}$ in the test-session. Such a simple attack results in authentication failure as follow: $\hat{B}$ is perfectly fooled to believe that it has shared a session key with $\hat{A}$ in the test-session, while $\hat{A}$ thinks it only ever took part in an aborted session with $\hat{M}$ in the matching session.[4]

Such an attack is simply ruled out for our DIKE protocol, particularly by the mechanism of $POK(\hat{B}, y)$. Specifically, after receiving the second-round message $(\hat{B}, Y = g^y, POK(\hat{B}, y))$ from $\hat{B}$ in the test-session, the attacker $\mathscr{A}$ cannot correctly compute and send to $\hat{A}$ the message of $(\hat{M}, Y = g^y, POK(\hat{M}, y))$ in the name of $\hat{M} \neq \hat{B}$ in the matching session.

[4]As suggested in [13], this "cutting-last-message-attack" can be prevented by adding an additional fifth round of "acknowledgement" from $\hat{A}$ to $\hat{B}$, but increasing the round and system complexity.

Resistance against unknown key share (UKS) attack. Informally speaking, by a successful UKS attack, an attacker $\mathscr{A}$ can successfully make two uncorrupted parties, say, $\hat{A}$ and $\hat{B}$, compute the same session-key in two sessions but have different views of who the peer to the exchange was, even if the adversary actually does not know the corresponding session-key.

Observe that the session-key of DIKE protocol is derived from $H_K(X, Y, g^{xy})$. If the two sessions are of the same session-key, with overwhelming probability in the RO model, these two sessions must be of the same DH-components, i.e., (X, Y) generated by the two uncorrupted players, and furthermore, in the same (initiator and responder) order. As uncorrupted players generate DH-components randomly and independently, with overwhelming probability, in addition to the two sessions suffering from the UKS attack there exist no other sessions of the same (ordered) DH-components (X, Y). Then, a successful UKS attack implies that $\mathscr{A}$ can at least malleate an authentic message, e.g., $NMZK(a, x) = H(sid, \hat{A}, X, Y, CDH(A, Y), CDH(X, Y))$ into $NMZK(a', x) = H(sid, \hat{A}, X, Y, CDH(A', Y), CDH(X, Y))$ for a different player $\hat{A}' \neq \hat{A}$ of public key A', which is impossible in the RO model without knowing $CDH(X, Y)$.

Perfect forward secrecy (PFS). Informally, a key-exchange protocol is PFS secure, if the leakage of the static secret-key of an uncorrupted player does not compromise the security of the session-keys established by the player for unexposed yet expired sessions, which have been erased from memory before the leakage occurred. In other words, once an unexposed session is expired and the session-key is erased from its holder's memory, then the session-key cannot be learned by the attacker even if the player is subsequently corrupted.

The PFS property of our DIKE protocol is from the following observations: (1) the computation of the session-key $H_K(X, Y, g^{xy})$ does not involve players' secret-keys. Thus, if the attacker $\mathscr{A}$ does not know x or y, under the CDH assumption it cannot derive the session-key even if both the two players' secret-keys are compromised. (2) If X is just generated by the attacker, it nevertheless cannot successfully finish the session with $\hat{B}$ (in the name of $\hat{A}$) by providing a correct authentic value $NMZK(a, x)$ (in particular, $\mathcal{A}$ cannot correctly compute $CDH(A, Y)$ w.r.t. the DH-component challenge Y from $\hat{B}$). Note that, for PFS security, we assume the secret-key of $\hat{A}$ is not compromised during the session run. The similar argument holds if Y is generated by the attacker itself.

Resistance against key-compromise impersonation (KCI) attack. Informally, security against KCI attacks says that: the knowledge of the secret-key of an uncorrupted player (e.g., a of $\hat{A}$) does not help an attacker $\mathscr{A}$ to impersonate another uncorrupted player (e.g., $\hat{B}$) to this uncorrupted *yet secret-key compromised* player (i.e., $\hat{A}$).

Our DIKE protocol is KCI secure, by the following observations: the authentic message $NMZK(a, x)$ (resp., $NMZK(b, y)$), which can be viewed as non-malleable zero-knowledge proof of the *joint* knowledge (a, x) (resp., (b, y)), is bounded to sender identity and is only w.r.t. the DH-component challenge sent by its peer. Under the CDH assumption in the RO model, the knowledge of a is useless to generate correct $NMZK(b, y)$ for $B \neq A$ w.r.t. the DH-component $X = g^x$ (for which the attacker doesn't know x).

On a complementary analysis of DIKE in [44]. Recently, we are referred to [44]. The privacy advantage of DIKE, i.e., the session-key transcript can be generated merely from DH-exponents x and y (and some public values), is, however, *re-stated* in [44] as a security vulnerability, by saying that the leakage of x can allow an attacker to finish the exposed session without knowing the static secret-key b and that this "violates" the principle of security against the leakage of DH-exponents. However, such a claim made in [44] is somewhat misleading. On the one hand, this is inherent to any deniable DHKE where the session transcript can be generated merely from DH-components (which we view a much preferable privacy advantage, as the session transcript cannot be traced to the pair of protocol participants). On the other hand, this does not breach the security in accordance with the CK-framework. Actually, if this is viewed as an "attack," most DHKE protocols just like IKEv2 and SIGMA, where the session-key is derived from g^{xy} (and some public values), are "insecure," as the leakage of x will cause the exposure of session-key and thus no security of the exposed session can be guaranteed. By "security resilient to the leakage of DH exponents" we mean the principle that such a leakage should have no bearing on the security of other (non-matching) sessions. Also, this is related to how to balance security and privacy in general, and our DIKE protocol is aimed to maximizing privacy protection while still providing robust security guarantees as analyzed above.

VI. Overview of CNMSZK Formulation and Analysis

A. Formulating (Privacy-Preserving) CNMSZK for DHKE

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a DHKE 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. A session-tag is complete if it consists of a complete set of all these components, e.g., $(\hat{A}, A, X, \hat{B}, B, Y)$. WLOG, we assume player's identity bears its public-key certificate.

We assume all communication channels, among all the concurrent sessions of $\langle \hat{A}, \hat{B} \rangle$, are unauthenticated and controlled by a PPT concurrent man-in-the-middle (CMIM) adversary $\mathscr{A}$. This means that the honest player instances cannot directly communicate with each other, since all communication messages are done through the adversary. The CMIM adversary $\mathscr{A}$ (controlling all communication channels) can do whatever it wishes. In particular, $\mathscr{A}$ can *concurrently* interact with polynomial number of instances of the initiator player $\hat{A}$ in the name of any player playing the role of the responder; such sessions are called the left-sessions. At the same time, $\mathscr{A}$ can *concurrently* interact with polynomial number of instances of the responder player $\hat{B}$ in the name of any player playing the role of the initiator; such sessions are called the right-sessions. For presentation simplicity, we assume the number of left-sessions is equal to that of right-sessions, which is $s(k)$ for some positive polynomial $s(\cdot)$ where k is the security parameter. The CMIM adversary $\mathscr{A}$ also takes some arbitrary auxiliary input $z \in \{0, 1\}^*$, which captures arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from the executions of arbitrary (*possibly different*) protocols prior to its actual session interactions with the instances of $\hat{A}$ or $\hat{B}$.

We denote by $view_{\mathscr{A}}(1^k, \hat{A}, A, \hat{B}, B, z)$ the random variable describing the view of $\mathscr{A}$ in its concurrent interactions with the instances of $\hat{A}$ and $\hat{B}$, which includes the input $(1^k, \hat{A}, A, \hat{B}, B, z)$, $\mathscr{A}$'s random tape, all the messages received in the $s(k)$ left sessions and the $s(k)$ right sessions, and the resultant session-keys for all successfully finished sessions (for an aborted or incomplete session, the session-key is defined to be "$\perp$"). For protocols in the RO model, $\mathscr{A}$'s view also includes the RO (see [2] for more details).

Definition 6.1: A DHKE protocol, $\langle \hat{A}, \hat{B} \rangle$, is called concurrent non-malleable statistical zero-knowledge (CNMSZK) *for both protocol participants simultaneously*, if for any PPT CMIM adversary $\mathscr{A}$ there exists a PPT simulator/extractor S such that for any sufficiently large k, any pair of uncorrupted players $\hat{A}$ and $\hat{B}$ (of public-key A and B respectively), and any auxiliary string $z \in \{0, 1\}^*$, the output of $S(1^k, \hat{A}, A, \hat{B}, B, z)$ consists of two parts (str, sta), where the distribution of the first output of S, i.e., str, is denoted by $S_1(1^k, \hat{A}, A, \hat{B}, B, z)$, such that the following hold:

Statistical simulatability. The following ensembles are statistically indistinguishable:

$$\begin{cases} \{view_{\mathscr{A}}(1^k, \hat{A}, A, \hat{B}, B, z)\}_{n, \hat{A}, A, \hat{B}, B, z} \\ \{S_1(1^k, \hat{A}, A, \hat{B}, B, z)\}_{n, \hat{A}, A, \hat{B}, B, z} . \end{cases}$$

Simultaneous knowledge extraction. The second output of S, i.e., sta, consists of a set of $2s(k)$ strings, $\{\tilde{w}_1^l, \tilde{w}_2^l, \cdots, \tilde{w}_{s(k)}^l, \tilde{w}_1^r, \tilde{w}_2^r, \cdots, \tilde{w}_{s(k)}^r\}$, satisfying:

- For any i, $1 \leq i \leq s(k)$, if the i-th left-session (resp., right-session) in str is aborted or with a tag identical to that of one of the right-sessions (resp., left-sessions), then $\tilde{w}_i^l = \perp$ (resp., $\tilde{w}_i^r = \perp$).
- Otherwise, i.e., the i-th left-session (resp., right-session) in str is successfully completed and it has a session-tag different from those of all right-sessions (resp., left-sessions), then $\tilde{w}_i^l = (\tilde{b}_i^l, \tilde{y}_i^l)$ (resp., $\tilde{w}_i^r = (\tilde{a}_i^r, \tilde{x}_i^r)$), where $\tilde{b}_i^l$ (resp., $\tilde{a}_i^r$) is the discrete-logarithm of the public-key $\tilde{B}_i^l$ (resp., $\tilde{A}_i^r$) set and alleged by the CMIM adversary $\mathscr{A}$ for the i-th left-session (resp., right-session) in the name of $\hat{\tilde{B}}_i^l$ (resp., $\hat{\tilde{A}}_i^r$), and $\tilde{y}_i^l$ (resp., $\tilde{x}_i^r$) is the discrete-logarithm of the DH-component $\tilde{Y}_i^l$ (resp., $\tilde{X}_i^r$) set and sent by the CMIM adversary $\mathscr{A}$ in the i-th left-session (resp., right-session).

Furthermore, we say the protocol $\langle \hat{A}, \hat{B} \rangle$ is *privacy-preserving* CNMSZK, if it additionally satisfies: (1) the transcript of each session can be generated merely from the DH-exponents (together with some public system parameters, e.g., players' public-keys and identities); (2) messages from one party do not bear the identity and public-key information of its peer. □

SK-security vs. CNMSZK for DHKE. We make some brief comparisons between the SK-security in accordance with the CK-framework and our formulation of CNMSZK for DHKE.

- At a high level, the SK-security essentially says that a party that completes a session has the following guarantees [12]: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the *unexposed* peer completes a matching session then the two parties have the same shared key.
  Roughly speaking, besides others, CNMSZK for DHKE ensures the enhanced guarantee of the above (2): if the *possibly malicious* peer completes a matching session, then not only the two parties have the same shared key, *but also and more importantly, the (possibly malicious) peer does "*know*" both the DH-exponent (and thus the shared session-key) and the secret-key corresponding to the DH-component and public-key sent and alleged by it in the test-session.* We suggest this kind of security guarantee is very essential to DHKE protocols, particularly when they are run concurrently over the Internet, which is, however, beyond the traditional SK-security.
- The CNMSZK formulation for DHKE follows the simulation approach [43] of tag-based CNMZK, which can actually be viewed as an extended and much strengthened version of the latter (more clarifications are given in Section VI-E). In particular, CNMSZK implies concurrent *forward* deniability for both the protocol initiator and the responder simultaneously. The SK-security definition follows the indistinguishability approach, which does not take deniability into account.
- Notice that the CNMSZK formulation is w.r.t. any efficient CMIM adversary of *arbitrary* auxiliary input $z \in \{0,1\}^*$. In particular, the adversary's auxiliary input can be dependent on players' public keys; for instance, z consists of a CDH triple (X, B, g^{xb}) or just the secret-key b. That is, the CNMSZK formulation implicitly captures the adversarial leakage of static secret-keys of uncorrupted players. However, static secret-key exposure *against uncorrupted players* was not captured by the SK-security in accordance with the basic CK-framework [12], where security against static secret-key exposure was separately considered beyond the CK-framework. On the other hand, the CNMSZK formulation does not take into account the following abilities of the CMIM adversary in the CK-framework: exposing ephemeral private state for incomplete sessions, exposing session-keys for completed sessions, and party corruption.

From the above clarifications, the CNMSZK security and the SK-security can be viewed complementary, and thus *it is much desirable to have DHKE protocols that enjoy both the SK-security and the CNMSZK security simultaneously.*

B. The Concurrent Knowledge-of-Exponent Assumption

For the formal analysis of CNMSZK of the DIKE protocol, we extend the knowledge-of-exponent assumption (KEA) into the *concurrent interactive* setting, which is referred to as concurrent knowledge-of-exponents assumption (CKEA).

The KEA assumption was introduced in [17] and then used in a large number of subsequent works. The KEA assumption was originally introduced to argue the non-malleability of public-key encryption that is a non-interactive cryptographic primitive [17] by nature. However, when arguing the security of *interactive* protocols running *concurrently* against CMIM adversaries, we notice that, in many scenarios (*particularly for DH-based authentication and key-exchange, as is the focus of this work*), the KEA assumption is insufficient. The reason is that, in such concurrent interactive settings, the CMIM adversary can potentially get access to a list of (polynomially many) DDH-oracles, with each being w.r.t. an element taken randomly and independently in G by an honest player instance.

For example, consider a two party protocol $\langle \hat{A}, \hat{B} \rangle$, where $\hat{A}$ generates and sends $X = g^x \in G$ and $\hat{B}$ generates and sends $Y = g^y \in G$. After or during the exchange of X and Y, each party uses the shared DH-secret g^{xy} to authenticate some public values (e.g., by MAC or simply hashing as in DIKE), and aborts in case the authentication from its peer is deemed to be invalid. Now, consider a CMIM adversary who, on the security parameter 1^k, concurrently interacts with $s(k)$ instances of $\hat{A}$ (by playing the role of $\hat{B}$) and $s(k)$ instances of $\hat{B}$ (by playing the role of $\hat{A}$) simultaneously, where $s(\cdot)$ is a positive polynomial. On an arbitrary value $Z \in G$, a random element X_i generated by an instance of the honest party $\hat{A}$ or $\hat{B}$, $1 \leq i \leq s(k)$, and another arbitrary element $Y \in G$ where Y may also be one of the random elements generated by $\hat{A}$ or $\hat{B}$, the CMIM adversary $\mathcal{A}$ can simply use Z (as the supposed DH-secret) to authenticate a value to the honest party who sends X_i in that session. Then, if that party aborts (that indicates the authentication using Z is invalid), $\mathcal{A}$ concludes $Z \neq CDH(X_i, Y)$, otherwise it concludes $Z = CDH(X_i, Y)$. This simple protocol example demonstrates that, in the *concurrent interactive* settings, the CMIM adversary can actually get access to polynomially many DDH-oracles.

These observations motivate us to introduce the following concurrent knowledge-of-exponents assumption (CKEA), for arguing the security of *interactive* cryptographic schemes against *concurrent* man-in-the-middle adversaries.

Definition 6.2: Suppose G is a cyclic group of prime order q generated by an element g, 1^k is the system parameter, $p(\cdot)$ and $q(\cdot)$ are positive polynomials. Define a decision predicate algorithm $\mathcal{O}_{\mathcal{C}}$ to be a DDH-Oracle for the group G and generator g w.r.t. the random challenge set $\mathcal{C} = \{C_1 = g^{c_1}, \cdots, C_{p(k)} = g^{c_{p(k)}}\}$, where c_i, $1 \leq i \leq p(k)$, is taken uniformly at random from Z_q^*. On a query of the form (X, Y, Z), for arbitrary $(X, Y) \in G^2$, the oracle $\mathcal{O}_{\mathcal{C}}$ outputs 1 if and only if $X \in \mathcal{C}$ and

$Z = CDH(X, Y)$. Consider an algorithm $\mathcal{A}$ with oracle access to $\mathcal{O}_{\mathcal{C}}$, denoted $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$, which, on input a triple $(g, \mathcal{C}, z)$, outputs a set of triples $\{(X_1, Y_1, Z_1), \cdots, (X_{q(k)}, Y_{q(k)}, Z_{q(k)})\} \subseteq (G^3)^{q(k)}$, where $z \in \{0,1\}^*$ is an arbitrary string that is generated independently of $\mathcal{C}$. (Specifically, we can consider an experiment where the DH-components in the set $\mathcal{C}$ are generated only after the auxiliary string z is fixed.) Such an algorithm $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$ is said to be a CKEA algorithm if, with non-negligible probability (over the choice of $g, c_1, \cdots, c_{p(k)}$ and $\mathcal{A}$'s random coins), $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}(g, \mathcal{C}, z)$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(k)}, Y_{q(k)}, Z_{q(k)})\} \subseteq (G^3)^{q(k)}$ satisfying $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(k)$.

We say that the CKEA assumption holds over G, if for every PPT CKEA-algorithm $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$ there exists another efficient PPT algorithm $\mathcal{K}$, referred to as the CKEA-extractor, such that for any polynomials $p(\cdot), q(\cdot)$ and sufficiently large k the following property holds except for a negligible probability: Let $(g, \mathcal{C}, z)$ be the input to $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$, ρ a vector of random coins for $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$ and ϖ a vector of answers given by $\mathcal{O}_{\mathcal{C}}$ on queries made by $\mathcal{A}^{\mathcal{O}_{\mathcal{C}}}$, on which $\mathcal{A}$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(k)}, Y_{q(k)}, Z_{q(k)})\} \subseteq (G^3)^{q(k)}$ satisfying $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(k)$. Then, on the same inputs and random coins and oracle answers, $\mathcal{K}(g, \mathcal{C}, z, \rho, \varpi)$ outputs $\{(X_1, Y_1, Z_1, y_1), \cdots, (X_{q(k)}, Y_{q(k)}, Z_{q(k)}, y_{q(k)})\}$ where $Y_i = g^{y_i}$ for all i, $1 \leq i \leq q(k)$. □

We note that the CKEA assumption can be viewed as the non-black-box counterpart of the gap Diffie-Hellman assumption, while the original KEA assumption is that of the computational Diffie-Hellman assumption. As we shall show in this work, the CKEA-based approach is a useful paradigm, and is powerful, for achieving highly practical DH-based cryptographic protocols provably secure against CMIM adversaries in concurrent settings like the Internet.

C. Zero-Knowledge Simulation With Restricted RO

When employing the simulation paradigm for proving the security of cryptographic protocols in the RO model, the RO is usually programmed by the simulator (i.e., the simulator provides random answers to RO queries). But a subtlety here is: ZK simulation with programmable RO loses deniability in general [40], [42], [45]. That is, a zero-knowledge simulation with programmable RO is meaningless for ensuring deniability in general. More specifically, the problem lies in the ability of the simulator in defining (i.e., programming) the RO on queries *first* made by the simulator itself (on behalf of the simulated honest parties) in order to simulate the actions of honest parties of private inputs. Specifically, the simulator runs the underlying adversary as a subroutine and mimics honest parties in its simulation. Typically, honest parties possess some private inputs (say, static secret-keys for DHKE), and get access to a non-programmable RO in reality. The simulator (in its simulation) has to take the advantage of its ability in programming the RO (to be more precise, programming the RO on queries first made by the simulated honest parties) in order to successfully simulate messages generated by the honest parties. However, such simulated honest-party messages (by using programmable RO) may not necessarily be able to be generated by a polynomial-time machine with the non-programmable RO in reality. For example, a Schnorr's signature can be a non-interactive zero-knowledge in the (programmable) RO model, but it is definitely undeniable. This is precisely the reason for the problems, particularly the loss of deniability, observed in [40], [42], [45] for zero-knowledge simulation with programmable RO.

To overcome the deniability loss problem of ZK simulation with (programmable) RO, Yung et al. [45] proposed the *restricted* RO model. In the restricted RO model, all the parties (particularly, all honest parties and the simulator) *except the adversary* get access to a non-programmable RO, but the adversary (who runs in polynomial time and possesses no private inputs) is still allowed to have access to a programmable RO. We can simply view that the restricted RO model is identical to the original RO model, except that the simulator is confined to programming the RO only on queries first made by the adversary (run by the simulator as its subroutine). The restricted RO model allows efficient interactive protocol implementations, while still reasonably avoiding the loss of deniability caused by simulation with fully programmable RO.

D. Analysis Overview of CNMSZK

Theorem 6.1: The DIKE protocol depicted in Figure 1 is *privacy-preserving* CNMSZK in the restricted RO model, assuming H is a (restricted) random oracle (but H_K can be any secure key derivation function), under the GDH assumption and the CKEA assumption over the group G.

Though the CNMSZK property of the DIKE protocol seems to be quite intuitive and straightforward, its formal analysis is quite lengthy and tedious. Below, we present the overview of the CNMSZK analysis, with focus on the tricks of using CKEA assumption and restricted RO in the CNMSZK analysis. The reader is referred to [46] for the complete proof.

High-level discussion. Consider a left-session between an instance of the honest initiator player $\hat{A}$ and a malicious responder player $\hat{B}$ (actually impersonated by the CMIM adversary $\mathscr{A}$). The simulator S generates $X = g^x$ by itself in the first-round message. After receiving $POK(\hat{B}, y) = H(sid, \hat{B}, Y, X, X^y)$ in the second round from $\mathscr{A}$, with overwhelming probability $\mathscr{A}$ has queried the RO with $(sid, \hat{B}, Y, X, X^y)$. Then, by the CKEA assumption, the value y can be extracted, based on which the authentic message $NMZK(a, x)$ (to be sent in the third round) can be generated by the simulator using a non-programmable RO (without knowing the static secret-key a). Furthermore, after receiving $NMZK(b, y)$ in the fourth round (i.e., in case $\mathscr{A}$ successfully finishes the session), with overwhelming probability $\mathscr{A}$ has queried the RO with $(sid, \hat{B}, Y, X, X^b, X^y)$, from which the secret-key b will also be extracted by the CKEA assumption.

Now, consider a right-session between an instance of the honest responder player $\hat{B}$ and a malicious initiator player $\hat{A}$ (actually impersonated by the CMIM adversary $\mathscr{A}$). After receiving X in the first round from $\mathscr{A}$, the simulator generates $Y = g^y$ and $POK(\hat{B}, y)$ by itself in the second round.

After receiving $NMZK(a, x)$ in the third round from $\mathscr{A}$ (i.e., in case $\mathscr{A}$ successfully finishes the session), with overwhelming probability $\mathscr{A}$ has queried the RO with (sid, X, Y, Y^a, Y^x), from which both the secret-key a and the DH-exponent x can be extracted by the CKEA assumption. Then, using the extracted DH-exponent x, the simulator can generate $NMZK(b, y)$ to be sent in the fourth round using a non-programmable RO (without knowing the secret-key b of the honest responder player $\hat{B}$).

Overview of the actual CNMSZK analysis. In the formal CNMSZK analysis, the polynomial-time simulator S generates DH-components and DH-exponents by itself by emulating the honest player instances. However, being different from honest player instances, S uses the DH-exponents (generated by S itself) merely for DDH-tests in its simulation. To this end, S maintains a DDH-test list, denoted $\mathcal{L}_{DDH}$, and stores all DDH-test records into $\mathcal{L}_{DDH}$. The key observation is: what can be done by the simulator S can also be done by another efficient oracle machine $S^{\mathcal{O}_\mathcal{C}}$ on the same common input and the random coins of S *except the coins used to generate the DH-components*, where $\mathcal{O}_\mathcal{C}$ is a DDH-oracle and $\mathcal{C} = \{X^l_1, \cdots, X^l_{s(k)}, Y^r_1, \cdots, Y^r_{s(k)}\}$ is the set of all the DH-components generated by S. Specifically, $S^{\mathcal{O}_\mathcal{C}}$ works just as S does, with the following modifications: (1) $S^{\mathcal{O}_\mathcal{C}}$ just sets the DH-component for the i-th left-session (resp., the j-th right-session) to be the value $X^l_i \in \mathcal{C}$ (resp., $Y^r_j \in \mathcal{C}$), $1 \leq i, j \leq s(k)$, rather than generating them by itself as S does. (2) Whenever $S^{\mathcal{O}_\mathcal{C}}$ needs to perform a DDH-test w.r.t. a DH-component in $\mathcal{C}$, it queries the DDH-test to its oracle $\mathcal{O}_\mathcal{C}$ and stores the record of the DDH-test into $\mathcal{L}_{DDH}$. Whenever $S^{\mathcal{O}_\mathcal{C}}/S$ needs to extract the DH-exponent and/or secret-key corresponding to the DH-component and/or public-key sent and alleged by the CMIM adversary $\mathscr{A}$, $S^{\mathcal{O}_\mathcal{C}}/S$ runs the CKEA-extractor $\mathcal{K}$ on the same common input, the random coins of $S^{\mathcal{O}_\mathcal{C}}$ *that just correspond to the coins of S except the coins used to generates the DH-components* X^l_i*'s and* Y^r_j*'s*, and $\mathcal{L}_{DDH}$ that corresponds to the vector of records of DDH-tests performed by $\mathcal{O}_\mathcal{C}$. By the CKEA assumption, $\mathcal{K}$ will successfully extract the corresponding DH-exponents and/or secret-keys with overwhelming probability. Notice that, for each successfully finished session (of different session-tag), from the extracted secret-key and DH-exponent the corresponding session-key can be computed, no matter what the key derivation function H_K is. That is, *we do not assume* H_K *to be an RO in the CNMSZK analysis.*

For the use of restricted RO, whenever S needs to send one of the values $NMZK(\hat{B}, y)$, $NMZK(a, x)$ and $NMZK(b, y)$, it first checks whether the value, denoted τ, has been defined by checking all the RO queries made by $\mathscr{A}$ and performing corresponding DDH-tests. If the value to be sent (i.e., τ) has already been defined (by $\mathscr{A}$'s RO query), it is set to be the already defined one; otherwise, S sets τ to be a random value $r \leftarrow \{0, 1\}^k$. If τ is set to be a random value r, from this point on whenever $\mathscr{A}$ makes an RO query, S checks whether the previously sent random value r is the answer to the RO query (again, by performing DDH-tests). Notice that, in the later case (i.e., τ is set to be a random value r that is however undefined by the RO), S does not try to use its knowledge of DH-exponents (generated by itself) to honestly compute the value τ, to ensure that those DH-exponents are used merely for DDH-tests in order to comply with the CKEA assumption. If $\mathscr{A}$ never makes an RO query with the previously sent random value r as the RO answer, the RO on this point remains undefined. In particular, S never defines it on its own, which ensures S works in the restricted RO model. By the above tricks, the simulator S works in *strict* polynomial time and its simulation is *straight-line* (i.e., without rewinding $\mathscr{A}$).

E. Comparisons With Related Works

Comparisons with deniable DHKE from [5]. Though deniable authentication was theoretically introduced by Dolev et al. in [22], to the best of our knowledge, the *first* practical deniable DHKE protocol, referred to as BMP-protocol, was proposed by Boyd et al. in the pioneer work [5] that opens the door for achieving fully-deniable practical DHKE. In a sense, the development of our DIKE protocol was inspired by, and based on, the BMP-protocol. Below, we briefly present differences between the BMP-protocol and our DIKE protocol.

- Our protocol is deniable (actually, CNMSZK) for both protocol participants simultaneously in the restricted RO model (without requiring POK/POP of secret-key during key generation). For the BMP-protocol [5], the interaction between an honest responder $\hat{B}$ and a malicious initiator $\hat{A}$ can be undeniable. Specifically, consider a malicious initiator $\hat{A}$ whose public-key A and DH-component X are externally given by (or generated interactively with) a third party (e.g., police) such that $\hat{A}$ (or anyone) actually does not necessarily know the secret-key a or the DH-exponent x, and the malicious $\hat{A}$ just aborts after receiving the authentic message from $\hat{B}$ in the second round. As the second-round authentic message involves the value A^b, which thus leaves an undeniable witness (to the malicious initiator or the third party) of the fact that $\hat{B}$ has gotten involved in the interaction. However, we notice that the deniability loss of the BMP-protocol in this case can be mitigated by additionally requiring POK/POP of secret-key during key generation. But we do not know how to formally prove (in the restricted RO model) the deniability of the honest responder against such a malicious aborting initiator, even if POK/POP of secret-key is also required during key generation.

  By comparison, for our DIKE protocol against such an aborting initiator, the authentic message included in the aborted transcript is only $POK(\hat{B}, y)$ that can be generated by a PPT machine using y and a non-programmable RO (without using any static secret-key), and is thus deniable. Moreover, the honest responder $\hat{B}$ proves the knowledge of its secret-key b in the fourth round, only if the malicious initiator has successfully proved the knowledge, via the authentic value $NMZK(a, x)$, of both secret-key a and DH-exponent x in the third round.

- The transcript of a session run of the BMP-protocol can be traced to the pair of protocol participants, as the value g^{ab} is involved in the generation of session transcript.
- The exchanged messages in the protocol proposed in [44] explicitly bear player role and peer's identity information.
- As already noticed by Boyd et al. in [6], the deniable DHKE protocol proposed in [5] does not provide KCI resistance, which is inherent to any KE protocol using statically keyed authenticators. However, this does not violate the security within the CK-framework.

CNMSZK for DHKE vs. traditional CNMSZK based approaches. Our CNMSZK formulation for DHKE is based on the traditional CNMSZK formulation (see, e.g., [22], [43]), but with some essential differences. On the one hand, traditional CNMSZK formulation considers a pair of players of *fixed roles*, specifically, one prover and one verifier. By comparison, our CNMSZK simulation for DHKE considers a pair of players of *interchanged roles*, i.e., each player plays both the role of ZK prover and that of ZK verifier. On the other hand, *privacy-preserving* CNMSZK proposes additional privacy requirements for the session messages of DHKE being exchanged concurrently over Internet. This, in particular, means that to achieve (privacy-preserving) CNMSZK is much more complicated, or harder, than to achieve traditional CNMSZK.

On identity hiding and unlinkability. A deniable KE protocol essentially ensures: given a session transcript between a pair of players $(\hat{A}, \hat{B})$, as well as the result session-key, each player $\hat{A}$ or $\hat{B}$ can deny the fact that it was ever involved in the generation of the session transcript and the session-key. That is, the session transcript and the session-key can be efficiently simulated from public values (e.g., players' certificates) *without knowing the knowledge of any static secret-key*. In other words, the session transcript and the session-key cannot serve as a valid witness to the fact either $\hat{A}$ or $\hat{B}$ was involved in the generation of the presented session transcript and session-key, as they can be simulated by some efficient algorithm.

But a deniable KE protocol does not aim to hide the information related to players' identities, affiliations or certificates, which is, however, the case in affiliation-hiding KE (see, e.g., [28], credential based KE [9] and unlinkable secure channel establishment [8]. Also, as the player uses the same certificate/identity in all sessions, it cannot preserve the anonymity property (as typically considered in group signature) or the unlinkability property.

Roughly speaking, a protocol is identity-hiding and unlinkable, if the multiple sessions run by the same party cannot be linked. Unlinkability particularly implies that protocol transcript does not bear players' identities in plain. We notice that, like IKEv2 and SIGMA, our DIKE protocols, and their implications as presented in the full version, can be extended to offer unlinkable secure channel establishment. The idea is to use the derived session-key to encrypt, via an authenticated encryption scheme, all the exchanged messages (except the session identifier sid) starting from the second round. Formal treatment of unlinkable DIKE protocols is left to a subsequent separate paper.

VII. Identity-Based Deniable Internet Key-Exchange

Identity-based key-exchange (IBKE) 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 the users). A list of identity-based key-exchange protocols have been developed in the literature, and the reader is referred to [15] for a good survey. However, to the best of our knowledge, deniable IBKE (for both the initiator and the responder simultaneously) with post-specified peers is still unknown. In this section, we present an identity-based variant of the DIKE protocol depicted in Figure 1, which is referred to as ID-DIKE for presentation simplicity.

Admissible pairing: Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing [1], [4], where G is a cyclic multiplicative group of order q generated by an element g. Here, an admissible pairing $\hat{e}$ satisfies the following three properties:

- Bilinear: If $x, y \in Z_q$, then $\hat{e}(g^x, g^y) = \hat{e}(g, g)^{xy}$.
- 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.

Definition 7.1 (Bilinear DH (BDH) assumption [4]*):* Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing as defined above. For any three elements $X = g^x$, $Y = g^y$ and $Z = g^z$ in G, where $x, y, z \in Z_q$, we denote by $BDH(X, Y, Z) = \hat{e}(g, g)^{xyz}$. An algorithm is called a BDH solver for $\hat{e}$ if it takes as input of any three elements $(X, Y, Z) \in G^3$ and its goal is to output the value of $BDH(X, Y, Z)$. We say the BDH assumption holds for $\hat{e}$ if for any PPT BDH solver, the probability that on input $(X, Y, Z) \leftarrow G^3$ (i.e., each of x, y and z is taken uniformly at random from Z_q), the solver computes the correct value $BDH(X, Y, Z)$ is negligible (in $k = |q|$). The probability is taken over the random coins of the solver, the choice of X, Y, Z uniformly at random in G (and also the random choice of the system parameters for (g, q, G, G_T)). □

The gap BDH (GGDH) assumption [33] essentially says that, for the pairing $\hat{e}$ defined over (g, q, G, G_T), computing $BDH(X, Y, Z)$, for $X, Y, Z \leftarrow G$, is strictly harder than deciding whether $U = BDH(X', Y', Z')$ for an arbitrary tuple $(X', Y', Z', U) \in G^3 \times G_T$.

Definition 7.2 (Gap BDH (GBDH) Assumption [33]): Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing defined over (g, q, G, G_T). Let a decision predicate algorithm $\mathcal{O}_B$ be a *(full)* Decisional Bilinear Diffie-Hellman (DBDH) Oracle for $\hat{e}$ such that on any input $(X', Y', Z', U) \in G^3 \times G_T$, oracle $\mathcal{O}_B$ outputs 1 if and only if $U = BDH(X', Y', Z')$. We say the GBDH assumption holds over (G, G_T), if for any PPT BDH solver the probability that on input of random

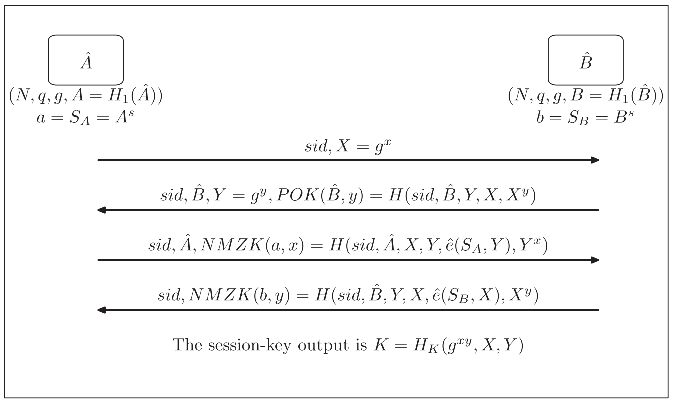
$\hat{A}$
$(N, q, g, A = H_1(\hat{A}))$
$a = S_A = A^s$
$\hat{B}$
$(N, q, g, B = H_1(\hat{B}))$
$b = S_B = B^s$
$sid, X = g^x$
$sid, \hat{B}, Y = g^y, POK(\hat{B}, y) = H(sid, \hat{B}, Y, X, X^y)$
$sid, \hat{A}, NMZK(a, x) = H(sid, \hat{A}, X, Y, \hat{e}(S_A, Y), Y^x)$
$sid, NMZK(b, y) = H(sid, \hat{B}, Y, X, \hat{e}(S_B, X), X^y)$
The session-key output is $K = H_K(g^{xy}, X, Y)$

Fig. 2. Identity-based deniable IKE.

elements $(X, Y, Z) \leftarrow G^3$ the solver computes the correct value $BDH(X, Y, Z)$ is negligible (in k), even when the algorithm is provided with the (full) DBDH-oracle $\mathcal{O}_B$. □

The ID-DIKE works as follows:

Setup: The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. Besides the functions H and H_K (that are the same in the description of the DIKE protocol in Section III), PKG also specifies a map-to-point hash function $H_1 : \{0,1\}^* \to G$. The public parameters are: $(G, G_T, \hat{e}, g, S, H_1, h, H_K)$. We assume the master secret-key s cannot be compromised.

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 $\hat{B}$ has public-key $B = H_1(\hat{B})$ and secret-key $S_B = B^s$.

ID-based DIKE between two users $\hat{A}$ and $\hat{B}$. The four-round ID-based DIKE protocol, in accordance with the main mode DIKE protocol depicted in Figure 1, is depicted in Figure 2.

Most advantageous features of the DIKE protocols described in Section III are inherited by the above ID-DIKE protocols. Below, we explicitly highlight some features of the ID-DIKE protocols.

- Deniability. The major difference between DIKE and ID-DIKE is that the value $CDH(A, Y)$ (resp., $CDH(B, X)$) in DIKE is now replaced by $\hat{e}(S_A, Y)$ (resp., $\hat{e}(S_B, X)$) in ID-DIKE. Observe that all the authentic values $POK(\hat{B}, y)$, and $NMZK(b, y)$ (resp., $NMZK(a, x)$) in ID-DIKE can still be computed merely from peer's DH-exponent x (resp., y). In particular, $\hat{e}(S_A, Y) = \hat{e}(A, S)^y$ (resp., $\hat{e}(S_B, X) = \hat{e}(B, S)^x$).
- Online efficiency. In case of pre-specified peer identity, $\hat{A}$ (resp., $\hat{B}$) can offline pre-compute $\hat{e}(B, S)$ and $\hat{e}(B, S)^x$ (resp., $\hat{e}(A, S)$ and $\hat{e}(A, S)^y$). That is, the online computation involved at each user side is essentially 1 pairing and 1 modular exponentiation.

Security analysis of ID-DIKE in the CK-framework, based on the GBDH assumption in the RO model, is similar to (and actually simpler than) that of Theorem 4.1, by the following observations: Though GBDH assumption implies that the CDH problem is still hard in the group G (even with a full DBDH oracle), the DDH problem becomes easy in G due to the underlying pairing operations. Below, we briefly discuss the differences between the analysis of ID-DIKE and that of DIKE. Denote $A = H_1(\hat{A}) = g^a$ and $B = H_1(\hat{B}) = g^b$.

In the proof of Lemma 4.1, the simulator does not need now to resort to a DDH oracle to ensure the consistency of RO answers related to $CDH(X, Y)$, as the DDH problem is easy in G now. By running the assumed attacker $\mathscr{A}$ as a subroutine, the simulator's goal now is to compute $\hat{e}(S_A, Y) = \hat{e}(g, g)^{yas} = BDH(Y, A, S)$ for Event-1 or $\hat{e}(S_B, X) = \hat{e}(g, g)^{xbs} = BDH(X, B, S)$ for Event-2. Toward this goal, the simulator resorts to a DBDH oracle to ensure the consistency of RO answers related to $\hat{e}(S_A, Y)$ and $\hat{e}(S_B, X)$. Note that we assume the master secret-key s of PKG cannot be compromised in the system, which is particularly unknown to both $\mathscr{A}$ and the simulator.

In the proof of Theorem 4.1, in order to rule out the forging attack, the weaker CDH assumption (implied by the GBDH assumption) is sufficient. Specifically, in order to compute $CDH(X, Y)$ from $X, Y \leftarrow G$, the simulator sets X and Y just to be the DH-components to be exchanged in the test-session. In the simulation, the simulator performs pairing operations (rather than resorting to a DDH oracle as in the proof of Theorem 4.1) to ensure the consistency of RO queries related to $CDH(X, Y)$.

Corollary 7.1: The ID-DIKE protocol (depicted in Figure 2), *with pre-computed and exposed DH components and exponents*, is SK-secure in the CK-framework with post-specified peers, under the GBDH assumption in the RO model.

Notice that the CKEA assumption over the group G is degenerated to a simplified version, where the DDH-oracle $\mathcal{O}_\mathcal{C}$ can be dispensed with as DDH is easy in the bilinear groups (G, G_T). Informally speaking, the KEA assumption over the bilinear groups (G, G_T) (originally introduced in [26]) says that: given a DH-challenge $X \leftarrow G$, the ability of coming up with $(B, S, Z) \in G^2 \times G_T$, where $Z = BDH(X, B, S) = e(g, g)^{bsx} = \hat{e}(S_B, X)$, implies the knowledge of $CDH(B, S) = S_B$. The CKEA assumption can be defined analogously over the bilinear groups (G, G_T). Here, the key differences are: (1) the DDH oracle in Definition 6.2 is replaced by a DBDH oracle $\mathcal{O}_\mathcal{C}$, which on a query of the form $(X, Y, S, Z) \in G^3 \times G_T$ outputs 1 iff $X \in \mathcal{C}$ and $Z = BDH(X, Y, S)$. (2) The algorithm $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ will output a list of values $\{(X_1, P_1, S, Z_1), \cdots, (X_{q(k)}, P_{q(k)}, S, Z_{q(k)})\} \in (G^3 \times G_T)^{q(k)}$, and the knowledge extracted by the extractor $\mathcal{K}$ will be $\{CDH(P_1, S), \cdots, CDH(P_{q(k)}, S)\}$. The CNMSZK analysis of the DIKE depicted in Figure 1 can be straightforwardly adapted to the ID-DIKE protocol, and we have the following corollary:

Corollary 7.2: The ID-DIKE protocol depicted in Figure 2 is *privacy-preserving* CNMSZK in the restricted random oracle model, under the GBDH assumption over the bilinear groups (G, G_T) and the CKEA assumption over both the group G and the bilinear groups (G, G_T). Here, the CKEA assumption over (G, G_T) (resp., G) is used to extract the corresponding secret-key S_A or S_B (resp., the DH-exponent x or y).

ACKNOWLEDGMENT

We are grateful to Prof. Colin Boyd for many very helpful and insightful comments, clarifications and suggestions, as well as related work references, which have much improved the presentation of this work. We thank the anonymous referees for their very helpful comments, Boru Gong for many helpful discussions and editing assistance.

REFERENCES

[1] S. Al-Riyami and K. Paterson, "Certificateless public-key cryptography," in *Proc. Asiacrypt 2003*, pp. 452–473.
[2] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. CRYPTO 1993*, pp. 273–289.
[3] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM CCS 1993*, pp. 62–73.
[4] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. CRYPTO 2001*, pp. 213–229.
[5] C. Boyd, W. Mao, and K. G. Paterson, "Deniable authenticated key establishment for Internet protocols," in *Proc. SPW 2003*, pp. 255–271.
[6] C. Boyd, W. Mao, and K. G. Paterson, "Key agreement using statically keyed authenticators," in *Proc. ACNS 2004*, pp. 248–262.
[7] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*. New York, NY, USA: Springer-Verlag, 2003.
[8] C. Brzuska, N. P. Smart, B. Warinschi, and G. J. Watson, "An analysis of the EMV channel establishment protocol," in *Proc. ACM CCS*, 2013, pp. 373–386.
[9] J. Camenisch, N. Casati, T. Gross, and V. Shoup, "Credential authenticated identification and key exchange," in *Proc. CRYPTO 2010*, pp. 255–276.
[10] R. Canetti, "Security and composition of cryptographic protocols: A tutorial," *SIGACT News*, vol. 37, no. 3, pp. 67–92, 2006.
[11] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proc. STOC 1996*, pp. 639–648.
[12] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Eurocrypt 2001*, pp. 289–307.
[13] R. Canetti and H. Krawczyk, "Security analysis of IKE's signature-based key-exchange protocol," in *Proc. CRYPTO 2002*, pp. 143–161.
[14] K. K. Choo, C. Boyd, Y. Hitchcock, and G. Maitland, "On session identifiers in provably secure protocols," in *Proc. SCN 2004*, pp. 351–366.
[15] M. C. Gorantla, R. Gangishetti, and A. Saxena, "A survey on ID-based cryptographic primitives," IACR (The International Association for Cryptologic Research), San Diego, CA, USA, Tech. Rep. 2005/094, 2005.
[16] C. J. F. Cremers, "Formally and practically relating the CK, CK-HMQV, and eCK security models for authenticated key exchange," IACR (The International Association for Cryptologic Research), San Diego, CA, USA, Tech. Rep. 2009/253, 2009.
[17] I. Damgård, "Towards practical public-key systems secure against chosen ciphertext attacks," in *Proc. CRYPTO 1991*, pp. 445–456.
[18] M. Di Raimondo and R. Gennaro, "New approaches for deniable authentication," in *Proc. ACM CCS 2005*, pp. 112–121.
[19] M. Di Raimondo, R. Gennaro, and H. Krawczyk, "Deniable authentication and key exchange," in *Proc. ACM CCS 2006*, pp. 466–475.
[20] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.
[21] Y. Dodis, J. Katz, A. Smith, and S. Walfish, "Composability and on-line deniability of authentication," in *Proc. TCC 2009*, pp. 146–162.
[22] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," *SIAM J. Comput.*, vol. 30, no. 2, pp. 391–437, 2000.
[23] C. Dwork, M. Naor, and A. Sahai, "Concurrent zero-knowledge," in *Proc. STOC 1998*, pp. 409–418.
[24] *Digital Signature Standard (DSS)*, FIPS Standard 186-2, Jan. 2000.
[25] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. STOC 1985*, pp. 291–304.
[26] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," in *Proc. Asiacrypt 2010*, pp. 321–340.
[27] D. Harkins and D. Carreal, "The Internet key-exchange (IKE)," IETF (The Internet Engineering Task Force), New York, NY, USA, Tech. Rep. 2409, Nov. 1998.
[28] S. Jarecki, J. Kim, and G. Tsudik, "Beyond secret handshakes: Affiliation-hiding authenticated key agreement," in *Proc. CT-RSA 2008*, pp. 352–369.
[29] C. Kaufman, "Internet key exchange (IKEv2) protocol," The Internet Engineering Task Force, London, U.K., Tech. Rep. 4306, Dec. 2005.
[30] S. Kent and R. Atkinson, "Security architecture for the Internet protocol," IACR (The International Association for Cryptologic Research), San Diego, CA, USA, Tech. Rep. 2401, 1998.
[31] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE-protocols," in *Proc. CRYPTO 2003*, pp. 400–425.
[32] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in *Proc. CRYPTO 2005*, pp. 546–566.
[33] C. Kudla and K. Paterson, "Modular security proofs for key agreement protocols," in *Proc. Asiacrypt 2005*, pp. 549–565.
[34] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Des., Codes Cryptogr.*, vol. 28, no. 2, pp. 119–134, 2003.
[35] W. Mao, *Modern Cryptography: Theory and Practice*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.
[36] U. Maurer and S. Wolf, "Diffie-Hellman oracles," in *Proc. CRYPTO 1996*, pp. 268–282.
[37] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.
[38] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," *J. ACM*, vol. 1, no. 2, pp. 231–262, 2004.
[39] K. Neupane, R. Steinwandt, and A. S. Corona, "Scalable deniable group key establishment," in *Proc. FPS 2012*, pp. 365–373.
[40] J. B. Nielsen, "Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case," in *Proc. CRYPTO 2002*, pp. 111–126.
[41] T. Okamoto and D. Pointcheval, "The gap-problems: A new class of problems for the security of cryptographic schemes," in *Proc. PKC 2001*, pp. 104–118.
[42] R. Pass, "On deniabililty in the common reference string and random oracle models," in *Proc. CRYPTO 2003*, pp. 316–337.
[43] R. Pass and A. Rosen, "New and improved constructions of non-malleable cryptographic protocols," in *Proc. STOC 2005*, pp. 533–542.
[44] A. P. Sarr and P. E. Vincent, "A complementary analysis of the (s)YZ and DIKE Protocols," in *Proc. Africacrypt 2012*, pp. 203–220.
[45] M. Yung and Y. Zhao, "Interactive zero-knowledge with restricted random oracles," in *Proc. TCC 2006*, pp. 21–40.
[46] A. C. Yao and Y. Zhao, "Deniable Internet key-exchange," IACR (The International Association for Cryptologic Research), San Diego, CA, USA, Tech. Rep. 2011/035, Jan. 2011.

Andrew Chi-Chih Yao is the Dean of the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing. He received the B.S. degree in physics from National Taiwan University in 1967, the Ph.D. degree in physics from Harvard University in 1972, and the Ph.D. degree in computer science from the University of Illinois in 1975. After serving on the faculty at MIT, Stanford, UC Berkeley and Princeton University, he left Princeton in 2004 to join Tsinghua Univeristy, Beijing. He is a Distinguished Professor-at-Large with the Chinese University of Hong Kong. He is a recipient of the Prestigious A. M. Turing Award in Year 2000 for his contributions to the theory of computation, including pseudorandom number generation, cryptography, and communication complexity. He is a member of the U.S. National Academy of Sciences and the Chinese Academy of Sciences.

Yunlei Zhao received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2004. He joined Hewlett-Packard European Research Center, Bristol, U.K., as a Post-Doctoral Researcher, in 2004. Since 2005, he has been with Fudan University, and is currently an Associate Professor with the Software School, Fudan University. His research interests include the theory and applications of cryptography, information security, and the interplay between cryptography and complexity theory.