Noname manuscript No.
(will be inserted by the editor)

One-Round Deniable Key Exchange with Perfect Forward Security

Weiqiang Wen · Libin Wang · Min Xie

Received: date / Accepted: date

Abstract In response to the need for secure one-round authenticated key exchange protocols providing both perfect forward secrecy and full deniability, we put forward a new paradigm for constructing protocols from a Diffie-Hellman type protocol plus a non-interactive designated verifier proof of knowledge (DV-PoK) scheme. We define the notion of DV-PoK which is a variant of non-interactive zero-knowledge proof of knowledge, and provide an efficient DV-PoK scheme as a central technical building block of our protocol. The DV-PoK scheme possesses nice properties such as unforgeability and symmetry which help our protocol to achieve perfect forward secrecy and full deniability respectively. Moreover, the security properties are formally proved in the Canetti-Krawczyk model under the Gap Diffie-Hellman assumption. In sum, our protocol offers a remarkable combination of salient security properties and efficiency, and the notion of DV-PoK is of independent interests.

Keywords Authenticated Key Exchange · Perfect Forward Secrecy · Full Deniability · Non-Interactive Zero-Knowledge · Proof of Knowledge

Mathematics Subject Classification (2000) 68M12 Network protocols · 94A60 Cryptography · 94A62 Authentication and secret sharing

1 Introduction

Motivation. Authenticated key exchange (AKE) is a cryptographic primitive that allows two entities to agree on a secure session key in public network. The design and analysis of AKE protocols continues to be a topic of active research after the seminal work of Diffie and Hellman [1], many works [2–5] have been dedicated to define stronger security models and provide novel protocols with many desired security properties.

Recently, the property of "deniability" is getting more and more attractive since it can protect personal privacy which we often concern in the real life or business activities. Motivated by the work of deniable authentication [6], Raimondo *et al* formally define the property of deniability for key exchange (KE) protocols [7], which is a central concern in KE protocols nowadays. The goal of deniability is to prevent a (possibly dishonest and malicious) receiver

Weiqiang Wen
School of Computer Science, South China Normal University, Guangzhou 510631, P.R. China
State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, P.R. China)
E-mail: weiqwen@gmail.com

Libin Wang
School of Computer Science, South China Normal University, Guangzhou 510631, P.R. China
State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, P.R. China)
Tel.: +86 020 85211352 ext. 406
Fax: +86 020 85215418
Corresponding author (E-mail: lbwang@scnu.edu.cn)

Min Xie
School of Computer Science, South China Normal University, Guangzhou 510631, P.R. China
State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, P.R. China)
E-mail: min.xie1988@gmail.com

convincing a third party that a given sender authenticated a given message. Using the simulation paradigm, the notion of deniability is characterized by constructing an effective machine (called the *simulator*) without possessing the secret key of the sender to simulate the view of any receiver. This natural and appealing definition characterizes the essential idea that the transcript of the protocol held by the receiver cannot be used to trace back to a specific sender, since it could be produced by a simulator instead of the actual sender.

Perfect forward secrecy (PFS) is another important security property that ensures damage confinement in the case of secrecy leakages, and was originally introduced by Diffie *et al* to describe a property of the Station-to-Station protocol [8]. Informally, an AKE protocol is said to be secure with PFS if disclosure of the long-term secret key of a party does not compromise the security of session keys established by that party before the disclosure occurred. Due to a general conclusion that PFS cannot be achieved by any implicitly authenticated 2-message or one-round protocol [3], many works which have been dedicated to design one-round AKE protocols with PFS [9–12] share a common design principle that uses explicit authentication techniques, e.g. digital signature or message authentication code (MAC) to thwart active attacks.

However, there is not any one-round AKE protocol providing full deniability as well as PFS so far. Addressing to the problem, we plan to construct a protocol achieves this goal. The obstacle to our plan is that we would like the protocol can counter active attacks and then achieves PFS without using traditional explicit authentication techniques, which will breach the deniability in general. To fulfill this need, we put forward a new paradigm for constructing protocols from the a Diffie-Hellman (DH) type protocol plus a non-interactive **d**esignated **v**erifier **p**roof **o**f **k**nowledge (DV-PoK) scheme which is a variant of non-interactive proof of knowledge (PoK) [13] and enjoys two additional properties, unforgeability and symmetry. Unforgeability guarantees that a valid proof could not be generated without trapdoors of the prover and the verifier, while symmetry allows an honest verifier generating a valid proof without the witness and thus ensures that the proof generated by the prover could not leave any traces.

Then, we provide an efficient DV-PoK scheme for the discrete logarithm language, its properties are formally defined and proved. Furthermore, we present a one-round **d**eniable **AKE** (named as DAKE) protocol with perfect forward secrecy by using DV-PoK as a central technical building block. By relying on the salient features of DV-PoK, the security properties can be formally proved in the Canetti-Krawczyk (CK) model [14] under the Gap Diffie-Hellman (GDH) assumption.

Related work. Informal treatment of deniability issues for KE protocols have been extensively studied [15–17]. Raimondo *et al* extended the definitional work of Dwork *et al* [18] from deniable authentication to deniable KE protocols, and formally prove that SKEME is full deniable and SIGMA is partial deniable in the sense of simulation paradigm. Jiang and Safavi-Naini [19] also proposed an efficient deniable key exchange protocol under a formal security model, which is a combination of Bellare-Rogaway KE protocol model [20] and the work of Dwork *et al* [18]. Deniable key exchange with a formal proof was obtained in [21, 22] as well. These work do not concern PFS at the same time.

The desire to achieve efficient perfect forward secrecy in one-round AKE protocols has been currently addressed by using explicit authentication, e.g., signature or MAC, which requires more communications complexity and computational complexity [9, 10]. Signature is a popular mean of authenticating the author of a message. An important consequence of signature is that anyone who holds the public key could verify the related signature, and then prove to a third party the signer of a message without signer's cooperation. In other words, the signer is unable at a later time to disclaim authorship of a message that she signed. Obviously, this property accounts for loss of deniability in AKE protocols which use signature as a building block. Though Cremers and Feltz [10] show that it is possible to establish deniability and perfect forward secrecy in one-round AKE protocol by using signature, the proposed scheme only provides peer-and-time deniability, which is weaker than full deniability.

In contrast to signature, MAC is a symmetric mechanism that requires the sender and the verifier sharing a common secret key to authenticate and verify. Since any third party without the secret MAC key could not generate valid transcripts, AKE protocols relying on MAC as a building block thwart the active attacks and thus provide perfect forward secrecy. Boyd and Nieto [9] construct a generic framework of AKE protocols with perfect forward secrecy by using a MAC scheme. However, this construction requires participants sharing a extra secret MAC key which adds additional cost, and until now it has been unknown if the construction guarantees full deniability.

Contributions. Firstly, we rephrase the definition of deniability for AKE protocols proposed by Raimondo *et al* [7]. To capture different abilities of an adversary, we formalize *full deniability* by using a simulation which is divided into two phases depending on whether the adversary possesses the secret key of the specific receiver or not.

Secondly, we define a notion of non-interactive designated verifier proof of knowledge holding two distinguishing properties, namely, unforgeability and symmetry. In addition, we propose an efficient DV-PoK scheme and show that the construction satisfies the security definition in the random oracle model under the GDH assumption.

Thirdly, we present a one-round AKE protocol combining a DV-PoK scheme with a Diffie-Hellman type KE protocol, and prove that the protocol satisfies perfect forward secrecy as well as full deniability in the CK model under the GDH assumption.

Organization. We first recall the definition of key exchange security and define the full deniability notion in Section 2. In Section 3 we introduce the notion of DV-PoK and present an efficient concrete DV-PoK scheme for the discrete logarithm language. The DAKE protocol and its security proof are presented in Section 4. Finally, further discussion and concluding remarks are made in Section 5.

2 Security Definition

2.1 Security Notions

Computational Diffie-Hellman (CDH) Assumption: Let k be a security parameter and $\mathbb{G}$ be a group generated by g with security parameter k, where the order of $\mathbb{G}$ is prime p and $|p| = k$. For two elements $U = g^u$, $V = g^v$ in $\mathbb{G}$, we denote the result of applying Diffie-Hellman computation to U and V by $\mathsf{CDH}(U, V)$. CDH Assumption means that for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, which is called a CDH solver, there exists a negligible function μ with security parameter k such that

$$\Pr[\mathcal{A}(g, U, V) = \mathsf{CDH}(U, V)] \leq \mu(k),$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a, b \in \mathbb{Z}_p$, and the random bits of $\mathcal{A}$.

Gap Diffie-Hellman (GDH) Assumption: The GDH assumption holds if no PPT CDH solver exists to solve the CDH problem, even if the CDH solver is equipped with a Decisional Diffie-Hellman (DDH) oracle for the group $\mathbb{G}$ and the generator g, where on arbitrary input $(U, V, W) \in \mathbb{G}^3$ the DDH oracle outputs 1 if and only if $W = \mathsf{CDH}(U, V)$.

2.2 Security Model

In this section we briefly describe the CK model needed in the rest of the paper, and emphasize how to capture the perfect forward secrecy in the CK model. The CK model describes a very realistic adversary which basically controls all communication in the network. The adversary $\mathcal{A}$ presents parties with incoming messages via **Send** queries, obtains the outgoing messages of the parties, and makes decisions about their delivery, thereby controlling all communications between parties.

- **Send**($Message$): The message has one of the following forms: $(\hat{U}_i, \hat{U}_j)$, $(\hat{U}_i, \hat{U}_j, out)$ or $(\hat{U}_i, \hat{U}_j, in, out)$. The adversary is given the session's response according to the protocol and the variables In, Out are initialized and updated (by concatenation) accordingly.

The peer that sends the first message in a session is called the *initiator* (denoted as $\mathcal{I}$) and the other the *responder* (denoted as $\mathcal{R}$). We usually denote the peers to a session by $\hat{A}$ and $\hat{B}$; either one may act as initiator or responder. An initiator session identifier is a five-tuple $(\mathcal{I}, \hat{A}, \hat{B}, Out, In)$ where $\hat{A}$ is the identity of the holder of the session and $\hat{B}$ is the peer. The session $(\mathcal{R}, \hat{B}, \hat{A}, Y, X)$ (if it exists) is said to be *matching* to the session $(\mathcal{I}, \hat{A}, \hat{B}, X, Y)$.

The adversary $\mathcal{A}$ can register arbitrary public keys of its choice, including public keys equal to keys of some honest parties in the system, on behalf of adversary-controlled parties. Additional, to capture information leakage $\mathcal{A}$ is allowed to make the following queries:

- **Session State Reveal**($\hat{U}$, sid): The adversary obtains the session-specific secret information (specified by a protocol) of a session sid being executed by $\hat{U}$.
- **Session key Reveal**($\hat{U}$, sid): The adversary obtains the session key computed by $\hat{U}$ in a completed session sid.
- **Corrupt**($\hat{U}$): By making this query the adversary takes full control of a user $\hat{U}$ and obtains the static key of $\hat{U}$ as well as the ephemeral secret information of all current sessions executed by $\hat{U}$.
- **Expire**(sid): This query takes a session sid as input and deletes the related session key (and any related session state).

If the adversary issues **Session State Reveal**($\hat{U}$, sid), or **Session key Reveal**($\hat{U}$, sid), or **Corrupt**($\hat{U}$) before **Expire**(sid), then sid is said to be locally exposed. If neither sid nor its matching session are locally exposed, sid is fresh. At any stage during its execution the adversary is allowed to make one special query **Test**(sid), where sid is an unexpired and fresh session. The goal of the adversary is to guess whether the challenge is a true session key or a randomly selected key. When the adversary terminates, it outputs a bit b'. The adversary wins the experiment if $b' = b$, where b is the random bit chosen in the **Test** query. The advantage **Adv** of the adversary $\mathcal{A}$ in breaking the security of a protocol Σ is defined as:

$$\mathbf{Adv}(\mathcal{A}) = \Pr[b = b'] - \frac{1}{2}.$$

Definition 1 ***(CK Security)*** *A key exchange protocol Σ is called CK secure if the following conditions hold:*

- *Users who complete matching sessions compute the same session key.*
- *For any PPT adversary $\mathcal{A}$ with the above capabilities running against Σ,* $\mathbf{Adv}(\mathcal{A})$ *is negligible.*

Perfect forward security is another important notion that provides a security guarantee of key exchange sessions in the case that the adversary has learned the private keys of some parties. This property is captured via the notion of *session key expiration* which represents the erasure of a session key from memory. The adversary against key exchange protocol is able to obtain the long-term keys of peers to the test session after the session key expired.

Definition 2 ***(CK security with perfect forward secrecy)*** *The protocol Σ is CK secure with perfect forward secrecy if the PPT adversary $\mathcal{A}$ is allowed to corrupt actor and peer to the test session after the session key expired, and*

- *Users who complete matching sessions compute the same session key.*
- *For any PPT adversary $\mathcal{A}$ with the above capabilities running against Σ,* $\mathbf{Adv}(\mathcal{A})$ *is negligible.*

2.3 Deniability of Key exchange protocols

We present an extended definition of deniability which is a variant of the definition proposed by Raimondo *et al* [7]. Aiming to explicitly specify the abilities of the adversaries and then to facilitate the security proof, we divide the definition into two parts with respect to two different kinds of adversaries, who possess the secret keys of the specific parties or not but still share a sole goal that is to accuse a given party had involved in a protocol execution. According to the definition which follows the traditional simulation paradigm, in the security proof we construct a two-phase simulation depending on whether the adversary corrupts honest parties or not. In the first phase, the adversary does not corrupt any specific parties, and the simulator is required to possess the same limited power as the adversary, which only runs on the public keys of the honest parties and the auxiliary information (denoted as aux) as inputs. In the second phase, the adversary does corrupt some specific parties and holds their secret keys, and the simulator is also allowed to possess the secret keys.

Consider an adversary $\mathcal{M}$ who runs on input an arbitrary number of public keys $\overrightarrow{\mathrm{pk}} = (\mathrm{pk}_1, \cdots, \mathrm{pk}_n)$, randomly generated by a key generation algorithm KG and associated with the honest parties in the network. The adversary initiates an arbitrary number of executions of the key exchange protocol with the honest parties, some as an initiator, others as a responder. These executions can be arbitrarily scheduled and interleaved by $\mathcal{M}$. The adversary's *view* consists of the transcript of the entire interaction and the session keys computed in all the protocols in which $\mathcal{M}$ participated (if the session does not complete, the session key is defined as an error value), together with the internal coin tosses of $\mathcal{M}$. We denote this view as $\mathrm{View}_{\mathcal{M}}(\overrightarrow{\mathrm{pk}}, \mathrm{aux})$. The definition of deniability is formalized as follows.

Definition 3 (Full Deniability) *Let $\Sigma = (\mathrm{KG}, \Sigma_I, \Sigma_R)$ be a KE protocol defined by a key generation algorithm* KG*, and interactive machines Σ_I, Σ_R specifying the roles of the (honest) initiator (the party who sends the first message) and responder, respectively. We say that Σ is a* fully deniable *key exchange protocol if for any adversary $\mathcal{M}$, for any input of public keys $\overrightarrow{\mathrm{pk}} = (\mathrm{pk}_1, \cdots, \mathrm{pk}_n)$ of the parties and any auxiliary input* aux*, there exist a simulator* SIM *that produces a simulated view which is indistinguishable from the real view of $\mathcal{M}$. The simulator* SIM *and the adversary $\mathcal{M}$ share the same input including the random coins.*

That is, consider the following two probability distributions where $\overrightarrow{\mathrm{pk}} = (\mathrm{pk}_1, \cdots, \mathrm{pk}_n)$ is the set of public keys of the parties, k is the security parameter:

$$\mathrm{Real}(k, \mathrm{aux}) = [(\mathrm{sk}_i, \mathrm{pk}_i) \leftarrow \mathrm{KG}(1^k); (\mathrm{aux}, \overrightarrow{\mathrm{pk}}, \mathrm{View}_{\mathcal{M}}(\overrightarrow{\mathrm{pk}}, \mathrm{aux}))]$$
$$\mathrm{Sim}(k, \mathrm{aux}) = [(\mathrm{sk}_i, \mathrm{pk}_i) \leftarrow \mathrm{KG}(1^k); (\mathrm{aux}, \overrightarrow{\mathrm{pk}}, \mathrm{SIM}_{\mathcal{M}}(\overrightarrow{\mathrm{pk}}, \mathrm{aux}))]$$

then for every PPT machine $\mathcal{D}$ running on the same input as $\mathcal{M}$, there exists a negligible function μ with security parameter k such that:

$$|\Pr_{x \in \mathrm{Real}(k, \mathsf{aux})}[\mathcal{D}(x) = 1] - \Pr_{x \in \mathrm{Sim}(k, \mathsf{aux})}[\mathcal{D}(x) = 1]| \leq \mu(k) \tag{1}$$

According to the adversary corrupts the honest parties or not, we partition the full deniability into two disjoint cases as follows.

Passive Deniability. *We say that Σ is* passive deniable *when the adversary $\mathcal{M}$ does not corrupt any honest parties and the equation 1 is satisfied.*

Active Deniability. *We say that Σ is* active deniable *when the adversary $\mathcal{M}$ does corrupt some honest parties and the equation 1 is satisfied.*

Remarks on the definition. We stress that there is not essential difference between our definition and Raimondo *et al*'s. In both of the definitions, the adversary has the same capabilities, and the simulator shares the same inputs with the adversary, which is a natural and reasonable assumption. However it is also worth highlighting the difference between the two definitions. In our definition, we explicitly define two different kinds of deniability according to the adversary corrupts the honest parties or not (holds the secret keys of the honest parties or not). In the Raimondo *et al*'s definition (and the subsequent security proof), the two situations are not explicitly defined and analyzed separately.

In the work of Raimondo *et al*'s, the deniablity of SKEME [17] is proved in the situation that the adversary has the party's secret key (see [7, Section 3.2, Theorem 3]). It claims that the deniability with respect to the adversary corrupting honest parties implies the deniability with respect to eavesdroppers (see [7, Section 2]), therefore the deniability with respect to eavesdroppers (corresponding to the case of *passive deniability* in our definition) is not analyzed. In fact, since the protocol SKEME uses public key encryption as a core mechanism of authentication, a perfect simulation with respect to a passive adversary attacking the deniability can be easily constructed.

In contrast to that, we organize a two-case definition since we focus on an AKE protocol based on a DV-PoK scheme, the adversary with or without party's secret key makes a significant difference in the process of simulation. The key point is that the outgoing message of a protocol execution includes a knowledge proof which relies on the sender's secret key, thus the simulation with respect to a passive adversary, in which the simulator does not hold the sender's secret key, can not be trivially obtained. We demonstrate a computationally indistinguishable simulation for this case in the proof of Theorem 3 (Section 4).

3 Non-Interactive Designated Verifier Proof of Knowledge

3.1 Definition

We formally define the notion of non-interactive designated verifier proof of knowledge which provides five properties including completeness, validity, adaptive zero-knowledge, unforgeability and symmetry.

The completeness and special soundness properties are basic requirements of a proof of knowledge. The completeness requirement means that if a prover does know the witness, it succeeds in convincing the honest verifier. We say that the proof is valid when the verifier accepts the proof. The special soundness property shows that if a given proof is valid, the prover who generated this proof really does know the witness. The special soundness property is captured by using an extractor who interacts with a prover to extract the witness that the prover claims to know. A proof of knowledge would be known as a zero-knowledge proof of knowledge when no additional knowledge is released. It requires that any verifier does not learn anything except that a proof is valid. Adaptive zero-knowledge is formalized by constructing a PPT simulator to generate a simulated proof which is indistinguishable from the view of the verifier in a real execution.

Unforgeability and symmetry are two distinguishing properties in our definition. In the case of unforgeability, we require that any PPT adversary without secret keys of the prover and the verifier cannot forge a valid proof though he knows the witness. In the case of symmetry, we require that a valid proof can be generated not only by the prover but also by the honest designated verifier which gets the witness.

Definition 4 (Non-Interactive Designated Verifier Proof of Knowledge (DV-PoK)) *Let* pp *be the public parameters,* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}})$ *and* $\mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$ *be the secret/public key pairs of the prover and the verifier respectively. A proof system* $\Pi = (\mathrm{Setup}, \mathrm{Gen}, \mathrm{P}, \mathrm{V})$ *is a non-interactive DV-PoK for the language* $L \in \mathcal{NP}$ *(with corresponding* $W_L(x)$*) as the set of all witnesses for public value* x *and witness relation* R_L*, if* Setup*,* Gen*,* P *and* V *are all PPT algorithms, and the following conditions hold:*

- **Completeness:** *For every* $x \in L$*,* $w \in W_L(x)$ *and public parameter* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$*, we have*

$$\Pr[\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V}); \pi \leftarrow \mathrm{P}(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}) : \mathrm{V}(x, \pi, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}) = 1] = 1,$$

  where ρ *denotes* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$.
- **Special Soundness:** *There exists a PPT algorithm* $\mathcal{E}$*, a knowledge extractor, with access to a prover oracle* $\mathrm{P}_{\mathrm{sk}_{\hat{\mathrm{P}}}}(\cdot)$ *such that for every* $x \in L$*, public parameter* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$ *and* $\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V})$*, and for every* $\hat{P}$ *for which* $p_x = \Pr[\pi \leftarrow \mathrm{P}(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}) : \mathrm{V}(x, \pi, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}) = 1]$*, we have*

$$\Pr[\mathcal{E}^{\mathrm{P}_{\mathrm{sk}_{\hat{\mathrm{P}}}}(\cdot)}(x) \in W_L(x)] \geq poly(p_x),$$

  where ρ *denotes* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$.

– **Adaptive Zero-Knowledge:** *For every non-uniform PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, and for every* $x \in L$*,* $w \in W_L(x)$ *and public parameter* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$*, there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that the following result is negligible*

$$|\Pr[\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V}); (x, w) \leftarrow \mathcal{A}_1(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}}) : \mathcal{A}_2^{\mathrm{P}(x,w,\mathrm{pp},\mathrm{sk}_{\hat{\mathrm{P}}},\mathrm{pk}_{\hat{\mathrm{P}}},\mathrm{pk}_{\hat{\mathrm{V}}})}(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}}) = 1] -$$

$$\Pr[(\tau, \rho) \leftarrow \mathcal{S}_1(\mathrm{pp}, \hat{P}, \hat{V}); (x, w) \leftarrow \mathcal{A}_1(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}}) : \mathcal{A}_2^{\mathcal{S}_2(x,\tau,\mathrm{pp},\rho)}(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}}) = 1]|$$

where ρ *denotes* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$*, and* τ *denotes the auxiliary information used by* $\mathcal{S}_2$*.*

– **Unforgeability:** *For every PPT adversary* $\mathcal{A}$ *access to a verifier oracle* $\mathrm{V}_{\mathrm{SIM}}(\cdot)$ *and a prover oracle* $\tilde{\mathrm{P}}_{\mathrm{SIM}}(\cdot)$*, and for every* $x \in L$*,* $w \in W_L(x)$ *and public parameter* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$*, there exists a negligible function* μ *such that*

$$\Pr[\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V}); (x, \pi) \leftarrow \mathcal{A}^{\mathrm{V}_{\mathrm{SIM}}(\cdot),\mathrm{P}_{\mathrm{SIM}}(\cdot)}(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}}) : \mathrm{V}(x, \pi, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}) = 1] \leq \mu(|x|),$$

where ρ *denotes* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$*.*

– **Symmetry:** *For every* $x \in L$*,* $w \in W_L(x)$ *and public parameter* $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$*, there exists a PPT algorithm* $\mathcal{B}$ *such that*

$$\Pr[\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V}); \pi \leftarrow \mathcal{B}(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}) : \mathrm{V}(x, \pi, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}) = 1] = 1,$$

where ρ *denotes* $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$*.*

3.2 Construction

Let $\mathbb{G} = \langle g \rangle$ is a multiplicative cyclic group of order q which is a k-bit prime number and $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$ is a target collision resistant hash function. We define a non-interactive DV-PoK scheme $\Pi = (\mathrm{Setup}, \mathrm{Gen}, \mathrm{P}, \mathrm{V})$ for the language $L = \{x : x = g^w, w \in \mathbb{Z}_q\} \in \mathcal{NP}$ with corresponding witness relation R_L, the set of all witnesses for public value x in L is defined to be $W_L(x) = \{w : x = g^w\}$. Note that $R_L(x, w)$ holds if and only if $x \in L$ and $w \in W_L(x)$. The concrete scheme is described as follows.

– **[Setup]** $\mathrm{Setup}(1^k)$:
  1. Chooses secure group parameters $(\mathbb{G}, g, q)$.
  2. Chooses a target collision resistant hash function H.
  3. Defines $\mathrm{pp} = (\mathbb{G}, g, q, \mathrm{H})$ and outputs it.
– **[Generation]** $\mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V})$:
  1. Chooses $\mathrm{sk}_{\hat{\mathrm{P}}} \in_R \mathbb{Z}_q, \mathrm{sk}_{\hat{\mathrm{V}}} \in_R \mathbb{Z}_q$ and set $\mathrm{pk}_{\hat{\mathrm{P}}} = g^{\mathrm{sk}_{\hat{\mathrm{P}}}}, \mathrm{pk}_{\hat{\mathrm{V}}} = g^{\mathrm{sk}_{\hat{\mathrm{V}}}}$.
  2. Outputs $\rho = (\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$.
– **[Prover]** $\mathrm{P}(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$:
  1. Chooses $r \in_R \mathbb{Z}_q$ and compute $R = g^r$.
  2. Computes $\sigma = \mathrm{pk}_{\hat{\mathrm{V}}}^{(\mathrm{sk}_{\hat{\mathrm{P}}}+w)}$, and $h = \mathrm{H}(\sigma, R)$.
  3. Computes $z = r + w \cdot h$ and output $\pi = (R, z)$.
– **[Verifier]** $\mathrm{V}(x, \pi, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}})$:
  1. Computes $\sigma' = (\mathrm{pk}_{\hat{\mathrm{P}}} \cdot x)^{\mathrm{sk}_{\hat{\mathrm{V}}}}$, and $h' = \mathrm{H}(\sigma', R)$.
  2. Computes $Z' = R \cdot x^{h'}$.
  3. If $Z' = g^z$ output 1, else output 0.

We prove the following theorem to show that Π constructed above is a non-interactive designated verifier proof of knowledge. Completeness always succeeds when the honest prover does know the witness of x. By using rewinding technique, if the proof is valid we can construct an extractor to reveal the witness of x and the validity property is directly derived. To prove adaptive zero-knowledge, we construct a simulator blinded to the witness to produce a valid proof that is indistinguishable from a real view. In addition, we reduce the unforgeability property to the GDH assumption, that is, any adversary breaking the unforgeability property can be used as a sub-routine to violate the GDH assumption. Lastly, the symmetry property is enjoyed since any honest designated verifier which gets the witness can generate a valid proof.

Theorem 1 *The scheme* $\Pi = (\mathrm{Setup}, \mathrm{Gen}, \mathrm{P}, \mathrm{V})$ *described above is a non-interactive DV-PoK in the random oracle model under GDH assumption.*

Proof. We give a formal proof to show that Π enjoys completeness, validity, adaptive zero-knowledge, unforgeability and symmetry as follows.

Completeness: Completeness follows by inspection. If the prover truly knows the witness w of x, then

$$g^z = R \cdot x^{\mathrm{H}(\mathrm{pk}_{\hat{\mathrm{V}}}^{(\mathrm{sk}_{\hat{\mathrm{P}}}+w)}, R)} = R \cdot x^{\mathrm{H}((\mathrm{pk}_{\hat{\mathrm{P}}} \cdot x)^{\mathrm{sk}_{\hat{\mathrm{V}}}}, R)} = R \cdot x^{h'},$$

and thus the verifier accepts z as a valid proof.

Special Soundness: We construct an extractor to reveal the witness. The extractor takes full control of the hash function modeled as a random oracle so that when the prover makes queries to the oracle, the extractor can decide what value to return. It runs the prover twice on the same randomness by rewinding such that the prover gives the same value $R = g^r$ both times, and then replies with two different values $c \neq c'$ to obtain two responses:

$$\begin{aligned} z &= r + w \cdot c \\ z' &= r + w \cdot c' \end{aligned}$$

Suppose that both of these executions would have convinced a honest verifier to accept. Given these two equations, the witness w can be derived:

$$w = \frac{z - z'}{c - c'}.$$

Adaptive Zero-Knowledge: For every public parameter $\mathrm{pp} \leftarrow \mathrm{Setup}(1^{|x|})$, we construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that the ensembles

$$\{\rho \leftarrow \mathrm{Gen}(\mathrm{pp}, \hat{P}, \hat{V}), \mathrm{P}(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}})\}$$

and

$$\{(\tau, \rho) \leftarrow \mathcal{S}_1(\mathrm{pp}, \hat{P}, \hat{V}), \mathcal{S}_2(x, \tau, \mathrm{pp}, \rho)\}$$

are computationally indistinguishable, where τ denotes the auxiliary information used by $\mathcal{S}_2$, ρ denotes $(\mathrm{sk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$. The simulator is operated in the following way:

1. Chooses $c \in_R \mathbb{Z}_q$ and $z \in_R \mathbb{Z}_q$.
2. Computes $R = g^z / x^c$.
3. Sets $\mathrm{H}((\mathrm{pk}_{\hat{\mathrm{P}}} \cdot x)^{\mathrm{sk}_{\hat{\mathrm{V}}}}, R) = c$.

It is observed that the output (R, z) of the simulation is a valid proof which succeeds in convincing the honest verifier and is indistinguishable from the output of the real view.

Unforgeability: We show that a PPT adversary $\mathcal{A}$ without $\mathrm{sk}_{\hat{\mathrm{P}}}$ and $\mathrm{sk}_{\hat{\mathrm{V}}}$ is unable to forge a valid proof with $R_L(x, w)$ holds. In order to bound the probability of a forging attack, we construct another adversary $\mathcal{S}$ solving the GDH problem. $\mathcal{S}$ takes full control of the hash function modeled as a random oracle and simulates a run of the scheme against $\mathcal{A}$. $\mathcal{S}$ takes a GDH challenge $(A = g^a, B = g^b)$ as input and sets $\mathrm{pk}_{\hat{\mathrm{P}}} = A$, $\mathrm{pk}_{\hat{\mathrm{V}}} = B$.

Firstly, we model the verifier oracle $\mathrm{V}_{\mathrm{SIM}}(\cdot)$ and prover oracle $\mathrm{P}_{\mathrm{SIM}}(\cdot)$ without secret keys of verifier and prover as follows.

- $\mathrm{V}_{\mathrm{SIM}}(x, \pi = (R, z), \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$:
  1. Search the output values recorded in the list of oracle H, and find a value h', whose corresponding input is a pair (σ', R') with $\mathsf{DDH}(\mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}} \cdot x, \sigma') = 1$ and $R' = R$.
  2. Check that if $g^z = R \cdot x^{h'}$ is true, output the proof is valid. Otherwise, output the proof is invalid.
- $\mathrm{P}_{\mathrm{SIM}}(x, w, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$:
  1. Randomly choose a value $h \in \mathbb{Z}_q$ (assume that the output of H is defined in the group $\mathbb{Z}_q$) and a randomness $r \in \mathbb{Z}_q$, now we can simulate a proof $z = r + w * h$. we make a record on the H list with information (r, x, $\mathrm{pk}_{\hat{\mathrm{P}}}$, $\mathrm{pk}_{\hat{\mathrm{V}}}$, h).

The only way that the adversary detects that the simulated proof produced by the prover oracle is invalid is to make a query to oracle H with input a pair (σ', R'), where $\mathsf{DDH}(\mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}} \cdot x, \sigma')$ equals 1 and $R' = g^r$. In this situation, we respond the query with the corresponding value h in the list of H recorded previously in the process of prover oracle.

As long as the forging attack succeeds, there must be a query $(\sigma = \mathsf{CDH}(A, B) \cdot \mathrm{pk}_{\hat{\mathrm{V}}}^w, R)$ in the queries list. By applying the extractor constructed in the proof of special soundness, $\mathcal{S}$ can obtain the witness w and compute $\mathrm{pk}_{\hat{\mathrm{V}}}^w$ and obtain $\mathsf{CDH}(A, B) = \sigma / \mathrm{pk}_{\hat{\mathrm{V}}}^w$. Due to the fact that the total number q of random oracle queries is polynomial, the simulator $\mathcal{S}$ can successfully guess the query (σ, R) corresponding to the GDH problem with probability $1/q$ and then

solve the GDH problem.

Symmetry: This property is obviously satisfied in our construction, since every PPT algorithm $\mathcal{B}$ with input $(x, w, \mathrm{pp}, \mathrm{sk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{P}}}$ could compute a valid proof to convince the honest designated verifier with the secret key $\mathrm{sk}_{\hat{\mathrm{V}}}$:

1. Chooses $r \in_R \mathbb{Z}_q$ and compute $R = g^r$.
2. Computes $\sigma = (\mathrm{pk}_{\hat{\mathrm{P}}} \cdot x)^{\mathrm{sk}_{\hat{\mathrm{V}}}}$ and $h = \mathrm{H}(\sigma, R)$.
3. Computes $z = r + w \cdot h$ and output $\pi = (R, z)$.

4 DAKE protocol

4.1 Protocol Construction

In this subsection we discuss the design rationales of our AKE protocol named DAKE to show the central techniques for achieving both perfect forward secrecy and full deniability at the same time, and describe the protocol execution in detail. In an execution of the protocol, the honest initiator (responder) produces an ephemeral public key $X = g^x$ ($Y = g^y$) along with a valid proof of X (Y) by using its secret key to convince the honest designated verifier that it does know the witness x (y). We prevent DAKE from active attacks by using DV-PoK as a building block which requires that everyone except the honest prover and the designated verifier is infeasible to generate a valid proof. In addition, DV-PoK allows any honest designated verifier to generate a valid proof by using its secret key and the witness of the ephemeral public key. Therefore, each role in DAKE could act like an honest designated receiver (verifier) to produce a communication which is indistinguishable from that generated by the honest sender (prover). In a word, DAKE provides perfect forward secrecy as well as the full deniability.

The protocol includes protocol setup, key generation and protocol execution. Figure 1 shows the informal description of our protocol, and the detail is described as follows.

Protocol setup. Let $k \in \mathbb{N}$ be the security parameter, $\mathbb{G} = \langle g \rangle$ is a multiplicative cyclic group of prime order q, $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$, $\hat{\mathrm{H}} : \{0,1\}^* \to \{0,1\}^\lambda$ are hash function and $\Pi = (\mathrm{Setup}, \mathrm{Gen}, \mathrm{P}, \mathrm{V})$ is a DV-PoK scheme. Define public parameter $\mathrm{pp} = (\mathbb{G}, g, q, \mathrm{H}, \hat{\mathrm{H}})$.

Key generation KGen(pp). For any party $\hat{U}$, the algorithm randomly chooses $u \in \mathbb{Z}_q$, sets $\mathrm{sk}_{\hat{\mathrm{U}}} = u, \mathrm{pk}_{\hat{\mathrm{U}}} = g^u$ and outputs (u, g^u).

Protocol execution. The protocol runs between two parties $\hat{A}$ (intitiator) and $\hat{B}$ (responder), which own the long-term key pairs $(\mathrm{sk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{A}}})$ and $(\mathrm{sk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{B}}})$ respectively. An execution of the protocol proceeds as follows.

1. Upon the activation, $\hat{A}$ selects $x \in \mathbb{Z}_q$ at random, computes $X = g^x$, generates $\pi_{\hat{\mathrm{A}}} = \mathrm{P}(X, x, \mathrm{sk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{B}}})$ and sends $(\hat{A},\ X,\ \pi_{\hat{\mathrm{A}}})$ to $\hat{B}$.
2. Upon receiving $(\hat{A},\ X,\ \pi_{\hat{\mathrm{A}}})$, $\hat{B}$ verifies if $\mathrm{V}(X, \pi_{\hat{\mathrm{A}}}, \mathrm{sk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{A}}}) = 1$, if true, $\hat{B}$ performs the following steps: (a) selects $y \in \mathbb{Z}_q$ at random, computes $Y = g^y$, generates $\pi_{\hat{\mathrm{B}}} = \mathrm{P}(Y, y, \mathrm{sk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{A}}})$, and sends $(\hat{B},\ Y,\ \pi_{\hat{\mathrm{B}}})$ to $\hat{A}$; (b) Computes $K_{\hat{\mathrm{B}}} = (\mathrm{pk}_{\hat{\mathrm{A}}} \cdot X)^{(\mathrm{sk}_{\hat{\mathrm{B}}} + y)}$ and sets $SK_{\hat{\mathrm{B}}} = \hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K_{\hat{\mathrm{B}}})$; otherwise, aborts.
3. Upon receiving $(\hat{B},\ Y,\ \pi_{\hat{\mathrm{B}}})$, $\hat{A}$ verifies if $\mathrm{V}(Y, \pi_{\hat{\mathrm{B}}}, \mathrm{sk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{B}}}) = 1$, if true, computes $K_{\hat{\mathrm{A}}} = (\mathrm{pk}_{\hat{\mathrm{B}}} \cdot Y)^{(\mathrm{sk}_{\hat{\mathrm{A}}} + x)}$ and sets $SK_{\hat{\mathrm{A}}} = \hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K_{\hat{\mathrm{A}}})$; otherwise, aborts.

It is not difficult to see that both the parties compute the same secret value $K = g^{(x+\mathrm{sk}_{\hat{\mathrm{A}}})(y+\mathrm{sk}_{\hat{\mathrm{B}}})}$ in a valid execution. Therefore, they have the same session key $SK = \hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K)$. All the intermediate computation values are erased immediately after the session completes, the session state of a party is specified as containing only the peer's identity, the outgoing and incoming Diffie-Hellman values X, Y, and proofs $\pi_{\hat{\mathrm{A}}}$, $\pi_{\hat{\mathrm{B}}}$ and the session key SK.

4.2 CK Security with Perfect Forward Secrecy

In this subsection, we investigate the CK security with perfect forward secrecy of DAKE. We present a formal security proof with two typical simulation phases to show that DAKE provide perfect forward secrecy under the CDH assumption in the random oracle model. In the first phase, we show that given a successful active adversary $\mathcal{A}$ against DAKE we can construct a forger to break the unforgeability property of the DV-PoK scheme involved in DAKE in

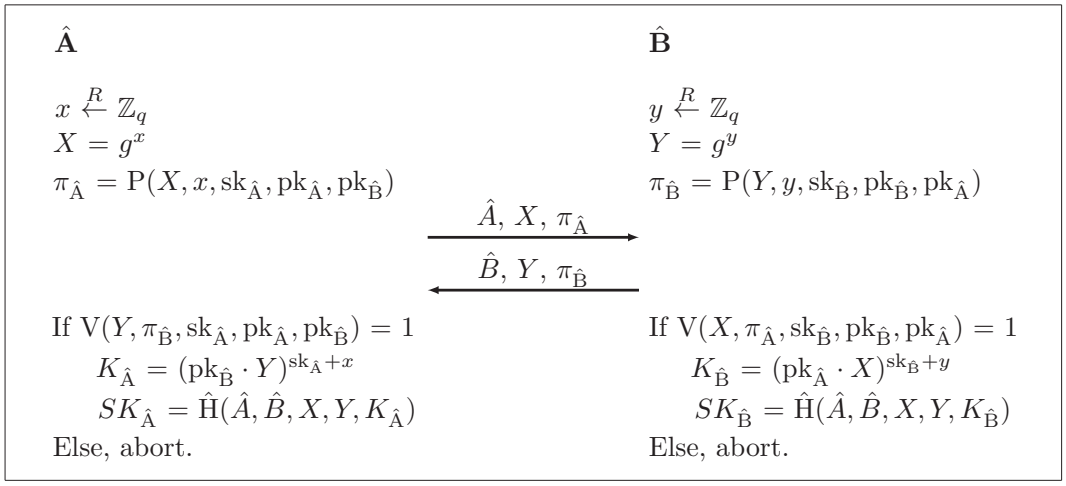

Fig. 1: The Description of **DAKE** Protocol.

contradiction to the assumed infeasibility of forging a valid proof without the secret keys of the honest prover and the honest verifier, and thus DAKE is resilient against active attacks.

By excluding active attacks in the protocol executions, we limit $\mathcal{A}$ to perform passive attacks which means that $\mathcal{A}$ is not actively involved with the choice of the X, Y values at a session (specifically it does not choose or learn the exponents of X and Y respectively). Under the circumstances, in the second phase, we prove that if $\mathcal{A}$ is a passive adversary then the resultant session key does enjoy perfect forward secrecy which allows $\mathcal{A}$ to learn the secret keys of both peers to the session by making **Corrupt** queries after the session key expired. We show that if $\mathcal{A}$ could distinguish the session key from a random key, we can get the existence of an efficient CDH solver in contradiction to the CDH assumption.

Theorem 2 *Suppose that* Π *is a DV-PoK scheme and* H *and* $\hat{\mathrm{H}}$ *are modeled as random oracles, then DAKE is a secure AKE protocol with perfect forward secrecy in the CK model under the GDH assumption.*

Proof. We start by observing that since the session key of the test session is computed as $SK = \hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K)$, the adversary $\mathcal{A}$ has only two ways to distinguish SK from a random key:

- Forging attack. At some point $\mathcal{A}$ queries $\hat{\mathrm{H}}$ on the same element group $(\hat{A}, \hat{B}, X, Y, K)$.
- Key-replication attack. $\mathcal{A}$ succeeds in forcing the establishment of another session that has the same session key as the test session.

The key-replication attack is impossible since the session key of the test session is computed under the random oracle $\hat{\mathrm{H}}$ which acts like a collision-resistant hash function. In turn, distinct sessions must have distinct secret values. To win the experiment, $\mathcal{A}$ must perform a forging attack if random oracles produce no collisions. Therefore, the following proof focus on the forging attack. Assume that the adversary $\mathcal{A}$ will test a session between party $\hat{A}$ and party $\hat{B}$. We separate the forging attack into two sub-case. In the first case, we consider the active adversary, which will actually pose an active attack on behalf of party $\hat{A}$ or party $\hat{B}$, i.e. actively sends messages. In the second case, after excluding the active adversary, we consider the passive adversary, which will not pose an active attack on behalf of party $\hat{A}$ and party $\hat{B}$.

Phase 1. The adversary $\mathcal{A}$ could break the security of DAKE via insertion of a message of its choice. Without loss of generality, we assume that $\mathcal{A}$ impersonates the party $\hat{B}$ (a similar argument hold for $\hat{A}$). In order to bound the probability of an active attack, we construct an adversary $\mathcal{S}$ against the unforgeability property of Π. $\mathcal{S}$ takes a challenge input $(\mathrm{pp}, \mathrm{pk}_{\hat{\mathrm{P}}}, \mathrm{pk}_{\hat{\mathrm{V}}})$ and simulates the behavior of the honest party $\hat{A}$ as follows.

- **Protocol Setup.** Setups the public parameter by choosing secure group parameters and two hash functions $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_q$ and $\hat{\mathrm{H}} : \{0,1\}^* \to \{0,1\}^\lambda$.
- **Key Generation.**
  1. Randomly chooses two parties $\hat{A}$ (initiator) and $\hat{B}$ (responder) and sets $\mathrm{pk}_{\hat{\mathrm{A}}} = \mathrm{pk}_{\hat{\mathrm{V}}}, \mathrm{pk}_{\hat{\mathrm{B}}} = \mathrm{pk}_{\hat{\mathrm{P}}}$.
  2. For any other party $\hat{U}$ different from $\hat{A}$ and $\hat{B}$, sets $(\mathrm{sk}_{\hat{\mathrm{U}}}, \mathrm{pk}_{\hat{\mathrm{U}}}) \leftarrow \mathrm{KGen}(\mathrm{pp})$.
- **Protocol Execution.** The sessions activated at party $\hat{U}$ other than $\hat{A}$ and $\hat{B}$ can be simulated obeying the protocol. Thus we only consider the sessions activated at party $\hat{A}$ (a similar simulation holds for the session owned by party $\hat{B}$).

1. **Send**($\hat{A}$, *sid*, ($\hat{U}$, *star*)): randomly chooses $x \in \mathbb{Z}_q$, computes $X = g^x$ and produce a proof (R, z) by making use of the prover oracle $\mathrm{P}_{\mathrm{SIM}}(x, X, \mathrm{pk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{U}}})$. Returns ($\hat{A}$, X, (R, z)) to party $\hat{U}$.
2. **Send**($\hat{A}$, *sid*, ($\hat{U}$, Y, $\pi_{\hat{\mathrm{U}}}$)): verifies $\pi_{\hat{\mathrm{U}}}$ by making use of the verifier oracle $\mathrm{V}_{\mathrm{SIM}}(Y, \pi_{\hat{\mathrm{U}}} = (R, z), \mathrm{pk}_{\hat{\mathrm{U}}}, \mathrm{pk}_{\hat{\mathrm{A}}})$. If the proof is invalid, abort. Else denotes the event $\hat{U} = \hat{B}$ and ($\hat{U}$, Y, $\pi_{\hat{\mathrm{U}}}$) is generated by $\mathcal{A}$ by **Forge** (the complementary event of **Forge** is denoted by $\overline{\textbf{Forge}}$) and considers the following two cases:
   (a) **Forge**. aborts and outputs $(Y, \pi_{\hat{\mathrm{U}}})$ as the forging proof.
   (b) $\overline{\textbf{Forge}}$. randomly chooses $x' \in \mathbb{Z}_q$, computes $X' = g^{x'}$ and produces a proof $\pi_{\hat{\mathrm{A}}} = (R', z')$ by making use of the prover oracle $\mathrm{P}_{\mathrm{SIM}}(X', x', \mathrm{pk}_{\hat{\mathrm{A}}}, \mathrm{pk}_{\hat{\mathrm{U}}})$. Returns ($\hat{A}$, X', (R', z')) to party $\hat{U}$. Then, randomly chooses a value $SK' \in \{0,1\}^\lambda$ as the session key and makes a record on the $\hat{\mathrm{H}}$ list with information Y, X', $\mathrm{pk}_{\hat{\mathrm{U}}}$, $\mathrm{pk}_{\hat{\mathrm{A}}}$, SK'. Responds the **Session State Reveal** with ($\hat{U}$, X', Y, $\pi_{\hat{\mathrm{A}}}$, $\pi_{\hat{\mathrm{U}}}$, SK'), and responds the **Session key Reveal** with SK'.

– **Query Oracle** H.
1. $\mathrm{H}(\sigma', R')$: checks that if $\mathsf{DDH}(\mathrm{pk}_V, X' \cdot \mathrm{pk}_P, \sigma)$ equals 1 and $R' = g^{r'}$, and the values (r', X', pk_P, pk_V, h') were recorded in the list of H previously by prover oracle. If true, outputs the corresponding value h'. Otherwise, simulates the random oracle H in the usual way.

– **Query Oracle** $\hat{\mathrm{H}}$.
1. $\mathrm{H}(\hat{A}, \hat{U}, X', Y', K')$: checks that if $\mathsf{DDH}(Y' \cdot \mathrm{pk}_{\hat{\mathrm{U}}}, X' \cdot \mathrm{pk}_{\hat{\mathrm{A}}}, K')$ equals 1, and the values (Y', X', $\mathrm{pk}_{\hat{\mathrm{U}}}$, $\mathrm{pk}_{\hat{\mathrm{A}}}$, SK') were recorded in the list of $\hat{\mathrm{H}}$ previously. If true, outputs the corresponding session key SK'. Otherwise, simulates the random oracle $\hat{\mathrm{H}}$ in the usual way.

This simulation is computationally indistinguishable from the real view. Especially, we take a complicated case in the simulation to analyze. In the event $\overline{\textbf{Forge}}$, the adversary could have registered any public key of its choice (thus knows the corresponding secret key) for party $\hat{U}$, however, the simulator does not know the corresponding secret key. In this case, the adversary does not possess $\hat{U}$'s or $\hat{A}$'s secret key. Fortunately, the simulation can be done successfully because both the prover oracle $\mathrm{P}_{\mathrm{SIM}}(\cdot)$ and the verifier oracle $\mathrm{V}_{\mathrm{SIM}}(\cdot)$ can be queried without both prover's and verifier's secret keys. Finally, if $\mathcal{A}$ succeeds in performing an active attack, then $\mathcal{S}$ directly outputs the proof generated by $\mathcal{A}$ and successfully breaks the unforgeability property of Π. By assumption that forging a valid proof without the secret keys of the prover and the verifier is infeasible, DAKE is resilient against active attacks.

Phase 2. After excluding active attacks in Phase 1, we show that $\mathcal{A}$ is unable to to distinguish the session key from a random key by performing passive attacks, even it is allowed to make **Corrupt** queries after the session key expired.

Given an adversary $\mathcal{A}$ that breaks the perfect forward secrecy property, we construct a CDH solver $\mathcal{S}$ that takes a CDH challenge (X_0, Y_0) as input. $\mathcal{S}$ simulates a run of DAKE as follows.

– **Protocol Setup.** Chooses secure group parameters as real execution of DAKE.
– **Key Generation.** For every party $\hat{U}$, sets $(\mathrm{sk}_{\hat{\mathrm{U}}}, \mathrm{pk}_{\hat{\mathrm{U}}}) \leftarrow \mathrm{KGen}(\mathrm{pp})$.
– **Protocol Execution.** Randomly chooses two parties $\hat{A}$ and $\hat{B}$, and guesses the test session *tid* and its matching session *tid*′.
1. **Send**($\hat{A}$, *tid*, *star*): computes $\pi_{\hat{\mathrm{A}}}$ by using $\mathrm{sk}_{\hat{\mathrm{B}}}$ and outputs ($\hat{A}$, X_0, $\pi_{\hat{\mathrm{A}}}$). If the session associated with *tid* is indeed the test session, then the session state and the session key of this session need not be simulated.
2. **Send**($\hat{B}$, *tid*′, ($\hat{A}$, X_0, $\pi_{\hat{\mathrm{A}}}$)): computes $\pi_{\hat{\mathrm{B}}}$ by using $\mathrm{sk}_{\hat{\mathrm{A}}}$ and outputs ($\hat{B}$, Y_0, $\pi_{\hat{\mathrm{B}}}$). If the session associated with *tid*′ is indeed the matching session of the test session, then the session state and the session key of this session need not be simulated.

According to the adaptive zero-knowledge property of Π, $\mathcal{S}$ can generate a valid proof of X_0 (Y_0) by using the secret key $\mathrm{sk}_{\hat{\mathrm{B}}}$ ($\mathrm{sk}_{\hat{\mathrm{A}}}$) of the designated verifier $\hat{B}$ ($\hat{A}$) without the witness of X_0 (Y_0), and thus the simulation above is perfect. In particular, when **Corrupt**($\hat{A}$) (**Corrupt**($\hat{B}$)) occurs after the session key expired, $\mathcal{S}$ directly provides $\mathrm{sk}_{\hat{\mathrm{A}}}$ ($\mathrm{sk}_{\hat{\mathrm{B}}}$) generated by KGen(pp) to $\mathcal{A}$.

If $\mathcal{A}$ does choose *tid*, whose matching session is *tid*′, as the test session and distinguishes the session key from a random key with non-negligible probability, then $\mathcal{A}$ must have made a query ($\hat{A}$, $\hat{B}$, X_0, Y_0, $K = \mathsf{CDH}(X_0, Y_0) \cdot X_0^{\mathrm{sk}_{\hat{\mathrm{B}}}} \cdot Y_0^{\mathrm{sk}_{\hat{\mathrm{A}}}} \cdot \mathrm{pk}_{\hat{\mathrm{A}}}^{\mathrm{sk}_{\hat{\mathrm{B}}}}$) to the random oracle $\hat{\mathrm{H}}$ from which the session key of *tid* (*tid*′) is derived. Now, with the knowledge of $\mathrm{sk}_{\hat{\mathrm{A}}}$ and $\mathrm{sk}_{\hat{\mathrm{B}}}$, $\mathcal{S}$ can compute $\mathsf{CDH}(X_0, Y_0) = K/(X_0^{\mathrm{sk}_{\hat{\mathrm{B}}}} \cdot Y_0^{\mathrm{sk}_{\hat{\mathrm{A}}}} \cdot \mathrm{pk}_{\hat{\mathrm{A}}}^{\mathrm{sk}_{\hat{\mathrm{B}}}})$ and solve the CDH problem.

4.3 Full Deniability

We prove that DAKE provides full deniability which is a combination of honest receiver deniability and dishonest receiver deniability according to Definition 3. Given an adversary $\mathcal{M}$ whose goal is to convince a third party that a given sender authenticated a given message, we attempt to construct a simulator $\mathcal{S}$ to produce the communications

during the AKE sessions as well as their session keys which are indistinguishable from the real execution.

Theorem 3 *Suppose that* Π *is a DV-PoK scheme, then DAKE is a fully deniable key exchange protocol in the random oracle model under the CDH assumption.*

Proof. According to Definition 3, the proof consists of two cases, passive deniability and active deniability, corresponding to the fact that $\mathcal{M}$ corrupts the specific party or not.

- **Passive deniability:** In this case, $\mathcal{S}$ has the same limited power as $\mathcal{M}$ which does not hold the secret keys of the honest parties. In order to bound the probability of an attack against passive deniability, we let $\mathcal{S}$ act like a CDH solver taking a CDH challenge (U, V) as input. $\mathcal{S}$ randomly chooses two parties $\hat{A}$ (initiator), $\hat{B}$ (responder) and embeds (U, V) into their public keys, that is, $\mathrm{pk}_{\hat{\mathrm{A}}} = U$ and $\mathrm{pk}_{\hat{\mathrm{B}}} = V$. Under the unforeability of the DV-PoK scheme, $\mathcal{M}$ can not actively send a valid message or participate in a completed protocol run. We only consider that $\mathcal{M}$ acts as an eavesdropper. Thus, $\mathcal{S}$ aims to simulate the view of an eavesdropper, which is a complete execution of DAKE in which $\hat{A}$ and $\hat{B}$ participate. In particular, we simulate protocol messages and session keys of the sessions activated by $\hat{A}$ as follows and the same simulations hold for the sessions activated by $\hat{B}$.
  1. Protocol message: $\mathcal{S}$ chooses $x \in \mathbb{Z}_q, R \in \mathbb{G}, z \in \mathbb{Z}_q$ at random, computes $X = g^x$ and outputs $(\hat{A}, X, (R, z))$. This simulation is computational indistinguishable from a real execution of DAKE under the CDH assumption. If $\mathcal{M}$ distinguishes the random message from a valid message, then $\mathcal{M}$ must have made a query $(\mathsf{CDH}(U, V) \cdot V^x, R)$ to the random oracle from which the valid proof of X is derived. Now, with the knowledge of x, $\mathcal{S}$ can compute $\mathsf{CDH}(U, V)$ and solve the CDH problem.
  2. Session key: $\mathcal{S}$ chooses $SK \in \{0,1\}^\lambda$ at random and outputs SK. This simulation is computational indistinguishable from a real execution where the session key computed as $SK{=}\hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K)$ otherwise violates the CK security with perfect forward secrecy of DAKE.
- **Active deniability:** In this case, $\mathcal{S}$ is allowed to take the same inputs as $\mathcal{M}$ including its random coins and the secret key of the specific party corrupted by $\mathcal{M}$. The simulation is under the assumption that $\mathcal{M}$ corrupts an honest responder $\hat{B}$ who interacts with an honest initiator $\hat{A}$, and obtains its secret key $\mathrm{sk}_{\hat{\mathrm{B}}}$. $\mathcal{S}$ takes an input $(\mathrm{sk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{B}}}, \mathrm{pk}_{\hat{\mathrm{A}}}, \mathrm{aux})$ and produces a simulated view as follows. Note that we are simulating an honest initiator party $\hat{A}$ (an honest responder party $\hat{B}$ can also be simulated by a similar way).
  1. Protocol message: $\mathcal{S}$ chooses $x \in \mathbb{Z}_q, r \in \mathbb{Z}_q$ at random, computes $X = g^x, R = g^r, z = r + x \cdot \mathrm{H}(\mathrm{pk}_{\hat{\mathrm{A}}}^{\mathrm{sk}_{\hat{\mathrm{B}}}} \cdot X^{\mathrm{sk}_{\hat{\mathrm{B}}}}, R)$ and outputs $(\hat{A}, X, \pi_{\hat{\mathrm{A}}} = (R, z))$. This simulation is perfect since $\mathcal{S}$ could generate a valid proof of X by using $\mathrm{sk}_{\hat{\mathrm{B}}}$ according to the symmetry property of Π.
  2. Session key: Upon receiving $(\hat{B}, Y = g^y, \pi_{\hat{\mathrm{B}}})$ generated by $\mathcal{M}$, $\mathcal{S}$ firstly verifies the proof of Y. Recall that $\mathcal{S}$ holds random coins of $\mathcal{M}$ and thus it could check the validity of $\pi_{\hat{\mathrm{B}}}$ with the knowledge of y and $\mathrm{sk}_{\hat{\mathrm{B}}}$. If $\pi_{\hat{\mathrm{B}}}$ is valid, computes $K = (X \cdot \mathrm{pk}_{\hat{\mathrm{A}}})^{(\mathrm{sk}_{\hat{\mathrm{B}}} + y)}$ and outputs the session key $SK = \hat{\mathrm{H}}(\hat{A}, \hat{B}, X, Y, K)$, otherwise, aborts the session and sets SK an error value. It is obvious that this simulation is perfect.

5 Discussion and concluding remarks

We put forward a new paradigm for constructing one-round deniable AKE protocols with perfect forward secrecy by combining a Diffie-Hellman type KE protocol with a DV-PoK scheme. Our construction significantly guarantee full deniability, rather than weaker deniability (e.g. peer-and-time deniability) provided in the other AKE protocols which use traditional explicit authentication techniques (e.g. signature or MAC) as a building block. DV-PoK is a special kind of non-interactive proof of knowledge with certain natural cryptographic indistinguishability properties, and can be efficiently instantiated for the discrete logarithm language at the cost of two exponentiations of the prover and three exponentiations of the verifier. An execution of DAKE requires for each party two exponentiations (one for the ephemeral public key and one for the session key), one DV-PoK generation and one DV-PoK verification, thus it is seven in total. We compare DAKE with other closely related protocols as follows.

Cremers and Feltz extended the Diffie-Hellman protocol to an one-round AKE protocol with perfect forward secrecy as well as peer-and-time deniability by using signature as a primitive [10]. A run of the protocol requires for each party two exponentiations (one for the ephemeral public key and one for the session key), one signature generation and one signature verification. The protocol is instantiated by using a deterministic signature scheme from Boneh *et al*'s work [23], which costs one exponentiation in the signature generation and one DH-tuple check in the signature verification. In comparison to Cremers and Feltz's protocol, our construction requires additional exponentiations of generating and verifying the proof of the ephemeral key and thus is slightly more expensive. However, our protocol provides full deniability which is a stronger security property than peer-and-time deniability.

To construct KE protocols with PFS, Boyd and Nieto [9] propose a generic one-round protocol which essentially is a compiler transforming any existing secure protocol with weak forward secrecy to achieve PFS. Instead of using signature, they introduced an extra public/secret key pair for each party to generate a shared MAC key between any two parties, such that the protocol can use a MAC scheme to thwart active attacks and thus achieves PFS at the cost of adding roughly one exponentiation per party. However, it is not claimed that the protocol achieves deniability. Comparing with the protocols that get from applying the Boyd and Nieto's compiler to existing protocols, e.g. TS3 protocol [24] and NAXOS [4], our protocol requires more expense because of the extra cost of DV-PoK, but it is worth emphasizing that, our protocol achieves higher security - full deniability..

The most similar protocol to ours is YAK protocol proposed by Hao [25], which is a one-round AKE protocol combining a Diffie-Hellman type KE protocol with a Zero-Knowledge proof scheme. The YAK protocol uses *Schnorr signature as a PoK scheme which allows the sender to prove the knowledge of the exponent without leaking it* [25, Section 3, Page 5], thus it does not achieve full deniability. It is claimed that the YAK protocol possesses many nice security properties, including *full forward secrecy* which is recognized as *weak perfect forward secrecy* [10, Section 8, Page 14]. However, we emphasize that the definitional method, the security model and proof techniques presented in the work of Hao [25] are fundamentally different from ours, hence we can not precisely compare the work of Hao with ours.

To sum up, our protocol offers a remarkable combination of advanced security properties and efficiency. In our future work, we concentrate on enhancing our protocol in a stronger security model, and constructing efficient DV-PoK schemes which can be proved secure without random oracles, subsequently providing a protocol can be formally proved secure in standard model.

References

1. Diffie, W. and Hellman, M.: 'New directions in cryptography', IEEE Trans. Inf. Theor., 1986, **22**, (6), pp 644–654.
2. Law, L., Menezes, A., Qu, M., Solinas, J., and Vanstone, S.: 'An efficient protocol for authenticated key agreement', Des. Codes Cryptogr., 2003, **28**, (2), pp 119–134.
3. Krawczyk, H.: 'HMQV: A high-performance secure Diffie-Hellman protocol', In Shoup, V. (Ed.): 'CRYPTO' (Springer-Verlag, LNCS 3621, 2005), pp 546–566.
4. LaMacchia, B., Lauter, K., and Mityagin, A.: 'Stronger security of authenticated key exchange', In Susilo, W., Liu, J. and Mu, Y. (Eds.): 'Provable Security' (Springer-Verlag, LNCS 4784, 2007), pp 1–16.
5. Ustaoglu, B.: 'Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS', Des. Codes Cryptogr., 2008, **46**, (3), pp 329–342.
6. Dwork, C., Naor, M. and Sahai, A.: 'Concurrent Zero-Knowledge', In Jeffrey Scott Vitter (Eds.): 'Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998.
7. Raimondo, M.D., Gennaro, R., Krawczyk, H.: 'Deniable Authentication and Key Exchange', IACR Cryptology ePrint Archive, 2006, **2006**, pp 280.
8. Diffie, W., Oorschot, P.C.V., Wiener, M.J.: 'Authentication and authenticated key exchanges', Des., Codes and Cryptogr., 1992, **2**, (2), pp 107–125.
9. Boyd, C., Nieto, J.G.: 'On Forward Secrecy in One-Round Key Exchange', In Chen, L.Q. (Ed.): 'Cryptography and Coding' (Springer-Verlag, LNCS 7089, 2011), pp 451–468.
10. Cremers, C.J.F., Feltz, M.:'One-round Strongly Secure Key Exchange with Perfect Forward Secrecy and Deniability', IACR Cryptology ePrint Archive, 2011, **28**, pp 300.
11. Xie, M., Wang, L.B.: 'One-round identity-based key exchange with Perfect Forward Security', Inf. Process. Lett., 2012, **112**, (14-15), pp 587–591.
12. Gennaro, R., Krawczyk, H. and Rabin, T.: 'Okamoto-Tanaka revisited: fully authenticated Diffie-Hellman with minimal overhead', In Zhou, J. and Yung, M. (Eds.): 'Proceedings of the 8th international conference on Applied cryptography and network security' (Springer-Verlag, LNCS 6123, 2010), pp 309–328.
13. Blum, M., Feldman, P., Micali, S.: 'Non-interactive zero-knowledge and its applications', In 'Proceedings of the twentieth annual ACM symposium on Theory of computing' (ACM, STOC, 1988), pp 103–112.
14. Canetti, R., Krawczyk, H.: 'Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels', In Pfitzmann, B. (Ed.): 'EUROCRYPT' (Springer-Verlag, LNCS 2045, 2001), pp 453–474.
15. Boyd, C., Mao, W.B., Paterson, K.G.: 'Key Agreement using Statically Keyed Authenticators', In Jakobsson, M., Yung, M. and Zhou, J.Y. (Eds.): 'Applied Cryptography and Network Security' (Springer-Verlag, LNCS 3089, 2004), pp 248–262.
16. Raimondo, M.D., Gennaro, R., Krawczyk, H.: 'Secure off-the-record messaging', In 'Proceedings of the 2005 ACM workshop on Privacy in the electronic society' (ACM, WPES, 2005), pp 81–89.
17. Krawczyk, H.: 'SKEME: a versatile secure key exchange mechanism for Internet', In Ellis, James T. and Neuman, B. Clifford and Balenson, David M. (Eds.): 'Proceedings of the 1996 Symposium on Network and Distributed System Security' (IEEE Computer Society, NDSS, 1996), pp 114–127.
18. Dwork, C., Naor, M., Sahai, A.: 'Concurrent Zero-Knowledge', In 'Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing' (ACM, STOC, 1999), pp 409–418.
19. Jiang, S.Q., Safavi-Naini, R.: 'An Efficient Deniable Key Exchange Protocol (Extended Abstract)', In Tsudik, G. (Ed.): 'Financial Cryptography and Data Security' (Springer-Verlag, LNCS 5143, 2008), pp 47-52.
20. Bellare, M., Rogaway, P.: 'Entity Authentication and Key Distribution', In Stinson, D.R. (Ed.): 'CRYPTO' (Springer-Verlag, LNCS 773, 1993), pp 232-249.
21. Yao, A.C., Zhao, Y.L.: 'Deniable internet key exchange', In Zhou, J. and Yung, M. (Eds.): 'Proceedings of the 8th international conference on Applied cryptography and network security' (Springer-Verlag, LNCS 6123, 2010), pp 329–348.

22. Jiang, S.Q.: 'Timed Encryption with Application to Deniable Key Exchange', In Agrawal, M., Cooper, S.B. and Li, A. (Eds.): 'Theory and Applications of Models of Computation' (Springer-Verlag, LNCS 7287, 2012), pp 248-259.
23. Boneh, D., Lynn, B., Shacham, H.: 'Short signatures from the Weil pairing', In Boyd, C. (Ed.): 'Advances in CryptologyASIACRYPT 2001' (Springer-Verlag, Journal of Cryptology 17, 2001), pp 514–532.
24. Jeong, I.R., Katz, J., Lee, D.H.: 'One-round protocols for two-party authenticated key exchange', In Jakobsson, M., Yung, M. and Zhou, J. (Eds.): 'Applied Cryptography and Network Security' (Springer-Verlag, LNCS 3089, 2004), pp 220–232.
25. Hao, F.: 'On robust key agreement based on public key authentication', In Sion, R. (Ed.): 'Financial Cryptography and Data Security' (Springer-Verlag, LNCS 6052, 2010), pp 383-390.