One Round Group Key Exchange with Forward Security in the Standard Model

M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto

Information Security Institute, Faculty of IT, Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia.
Email: mc.gorantla@isi.qut.edu.au, {c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. Constructing a one round group key exchange (GKE) protocol that provides forward secrecy is an open problem in the literature. In this paper, we investigate whether or not the security of one round GKE protocols can be enhanced with any form of forward secrecy without increasing the number of rounds. We apply the *key evolving* approach used for forward secure encryption/signature schemes and then model the notion of forward security for the first time for key exchange protocols. This notion is slightly weaker than forward secrecy, considered traditionally for key exchange protocols. We then revise an existing one round GKE protocol to propose a GKE protocol with forward security. In the security proof of the revised protocol we completely avoid reliance on the random oracle assumption that was needed for the proof of the base protocol. Our security proof can be directly applied to the base protocol, making it the most efficient one round GKE protocol secure in the standard model. Our one round GKE protocol is generically constructed from the primitive of forward secure encryption. We also propose a concrete forward secure encryption scheme with constant size ciphertext that can be used to efficiently instantiate our protocol.
Keywords. group key exchange, forward security, key evolving, standard model

1 Introduction

A group key exchange (GKE) protocol allows a set of parties to agree upon a common secret session key over a public network. The communication efficiency of such a protocol is determined by the number of *rounds* it takes to complete the protocol and the size of the messages exchanged. A round includes all the messages that can be sent simultaneously by the parties during the protocol execution. This implies that in any given round, the parties do not have to wait for the messages from the other parties before sending out their messages in that particular round. Hence, in a one round GKE protocol no party should have to wait before initiating the protocol and sending its outgoing messages (if any). In these days of ubiquitous connectivity to the Internet, the parties are often at distant locations with each party connected via different devices over different local networks. If these parties want to establish a common session key by executing a GKE protocol, a single party connected to a congested local network is enough to cause considerable delay in computing the session key as the number of communication rounds increases. Hence, it is highly desirable to have a protocol with as minimum number of rounds as possible.

Diffie and Hellman [23] first proposed a simple one round two-party key exchange (2PKE) protocol and later many one round 2PKE protocols have been proposed with improved security properties. Joux [31] generalized the Diffie-Hellman protocol to the three-party setting using bilinear pairings over elliptic curves, with only one round of communication. However, all such generalizations to a group of size more than three have multiple rounds [29, 19, 44]. To the best of our knowledge, the only GKE protocols with one round of communication are the protocols of Boyd and González Nieto [14] and Gorantla et al. [28].

Forward Secrecy. A key exchange protocol with forward secrecy ensures that even if the long-term key of a party is exposed, all the past session keys established using that long-term key will remain uncompromised. Forward secrecy is one of the most important security attributes for key exchange protocols since it limits the damage of long-term key exposure. Forward secrecy can be classified into two types: full (perfect) forward secrecy (FFS) and weak forward secrecy (WFS), based on the adversarial power. A protocol with FFS allows the adversary to be active during the protocol execution and still makes sure that the session key will remain uncompromised when the long-term key is revealed. On the other hand, in a protocol with WFS the adversary is required to remain passive during the protocol execution. There exist many one-round 2PKE protocols [23, 40, 38, 36, 13] which achieve WFS. The recent one round protocol of Gennaro et al. [26] also achieves FFS using identity based techniques. The common technique to achieve WFS/FFS in all these protocols is to ensure that the ephemeral Diffie-Hellman key between the two parties is part of the shared secret from which the session key is derived.

A similar approach is followed for the case of tripartite setting by ensuring that the ephemeral bilinear Diffie-Hellman key is embedded within the shared secret [31, 2]. However, it seems no such ephemeral key can be derived for the group case in a single round unless n-linear pairings are realized [12]. On the other hand, it is known that the both the existing one round GKE protocols [14, 28], which are not Diffie-Hellman type protocols, do not provide even WFS. Since forward secrecy is a highly desirable property for key exchange protocols, it is essential that one-round GKE protocols achieve some level of security when the long-term keys are compromised. A natural question to ask now is whether it is possible to strengthen the security of one-round GKE protocols or not. More specifically, can we provide any form of security to the session key established in one round GKE protocols when the long-term key is revealed? We answer this question in the affirmative by adopting the notion of *forward security* for the first time for key exchange protocols.

Forward Security. Back [4] floated the idea of *non-interactive forward secrecy* (now known as forward security) for public key cryptosystems and Anderson [3] later extended it to public key encryption and signature schemes. The notion of forward security defined for public key cryptosystems [3, 7, 20] is similar to the notion of forward secrecy considered for key exchange protocols. The goal of forward security for public key schemes is to ensure the appropriate notion of security, even if the long-term private key used in the corresponding scheme is compromised. For example, in the case of encryption schemes, forward security should assure that all the messages encrypted earlier will remain confidential in the event of the long-term private key leakage.

Forward security is achieved by employing a *key evolving* scheme for the long-term private keys of the parties. In this scheme, a party initially starts with a standard public-private key pair. The lifetime of the scheme is divided into N time periods with each period being labelled $0, \ldots, N-1$. As the party enters a time period τ, it applies a *one-way* transformation to the private key $SK_{\tau-1}$ to obtain a private key SK_τ for the current time period. $SK_{\tau-1}$ is then erased. The public key remains the same for all the N time periods. Bellare and Miner [7] and Canetti et al. [20] proposed key evolving schemes that are more efficient than the initial proposal of Anderson [3] such that the private key size does not grow linearly with the number of periods. There exist many other public key cryptographic primitives for which the notion of forward security has been considered [37, 43, 1]. Itkis [30] gave an excellent survey on forward security for public key cryptography.

Forward Security for Key Exchange. The notion of forward security has been not yet considered for key exchange protocols. The main reason for this seems to be the fact that there exist both 2PKE and GKE protocols in abundance, which provide forward secrecy. As mentioned above there exist even one-round 2PKE protocols that achieves both WFS and FFS. However, when it comes to the case of one-round GKE protocols, we still do not how to construct concrete protocols with forward secrecy. Hence, all that one round GKE protocols can achieve at this stage seems to be forward security. A naive approach to attain forward security for GKE protocols may involve having different certified public keys with short life time and then the parties may execute the known one-round GKE protocols. However, note that this approach involves significant overhead in terms of the number of interactions with the certifying authority and the related certificate management issues. On the other hand, the key evolving approach avoids this problem by allowing a user to update its own private key, while keeping the public key the same for longer periods. The key evolving approach may also be applied to one-pass key establishment protocols [41] to make them achieve *receiver forward security.*

Security Notions for GKE. Adversaries against a GKE protocol can be divided into two types: outsiders, who are not part of the group and insiders, who are members of the group. A desired notion of outsider security for GKE protocols, introduced by Bresson et al. [17, 15, 16], is authenticated key exchange (AKE) security. Informally, AKE-security demands that an outsider adversary should not learn the session key. The property of forward secrecy can be captured by an appropriate definition of AKE-security. The notions of mutual authentication and contributiveness in the presence of insiders have been defined as desired notions of insider security for GKE [33, 9, 18]. GKE protocols with these additional security guarantees often require more than one round [34, 33, 39, 27]. Since our emphasis here is on communication efficiency, we concentrate only on one round AKE-secure GKE protocols with forward security.

1.1 Contributions

In this paper, we introduce a new notion of AKE-security for GKE protocols with forward security. We assume that each party in a forward secure GKE protocol has a long-term public-private key pair. A key evolving scheme is then applied to each long-term private key. Forward security for GKE protocols guarantees that even if the private key SK_τ is exposed, all the session keys established prior to the time period τ will remain uncompromised. However, note that this does not imply forward secrecy since the session keys established so far with the private key SK_τ will be compromised if SK_τ is leaked. Hence, in forward secure GKE protocols, it is not possible to ensure the secrecy of the session keys established during period τ or any subsequent periods. However, if the GKE protocol is executed only once in a time period τ, the level of security offered by forward security would be identical to the one offered by the traditional forward secrecy considered for GKE protocols.

We can construct one-round GKE protocols with forward security by revising the one round protocols of Boyd and González Nieto [14] (BG) and Gorantla et al. [28]. However, note that the BG protocol is *role asymmetric* compared to the protocol of Gorantla et al. i.e., in the BG protocol only one party performs public key encryption and signature operations whereas in Gorantla et al.'s protocol all the parties execute a public key encryption algorithm. Consequently, the BG protocol is computationally more efficient than the protocol of Gorantla et al. for all except one party. Hence, we modify the BG protocol by replacing the normal public key encryption scheme in their protocol with a forward secure encryption scheme.

We then show that the revised one round GKE protocol satisfies our new AKE-security. Although our modifications to the BG protocol are straightforward, our crucial contribution is that we prove the security of the revised protocol in the standard model. Note that the BG protocol till now is known to be secure in the random oracle model. The security proof of the BG protocol assumes that the underlying public key encryption is chosen plaintext secure. However, we prove the security of the revised protocol assuming that the underlying forward secure encryption scheme is chosen ciphertext secure. Surprisingly, this minor change in assumptions enable us to carry out the proof in the standard model. Particularly, it allows us to simulate the *session key reveal* and *test* queries (for which the random oracle assumption was needed in the BG protocol proof) without resorting to the random oracle assumption. Our proof technique can be directly applied to show that the BG protocol is secure in the standard model. Consequently, the BG protocol becomes the most efficient one round GKE protocol secure in the standard model.

Our protocol is generic in the sense that any forward secure encryption scheme with chosen ciphertext security can be used to instantiate it. However, the ciphertext lengths of the known forward secure encryption schemes of Canetti et al. [20] have logarithmic complexity in the number of total time periods. Instantiating our protocol with any of these two encryptions schemes will result in a GKE protocol whose messages will also have the same complexity. Hence, we also construct a new forward secure encryption scheme with constant size ciphertext from the *hierarchical ID-based encryption* scheme of Boneh et al. [11] and show that it achieves chosen ciphertext security in the standard model. Note that although Boneh et al. observed that a forward secure encryption scheme could be obtained from their hierarchical ID-based encryption scheme, constructing the forward secure encryption scheme and proving it chosen ciphertext secure has still not yet been done. We recommend instantiating our GKE protocol with the new forward secure encryption scheme since it will result in a protocol that has $O(1)$ message size complexity.

Specific contributions of our paper are:

- Security model for forward secure GKE protocols
- A one-round GKE protocol with forward security
- Proof of security for the proposed protocol in the standard model
- A forward secure encryption scheme with constant size ciphertext

OUTLINE. In the remainder of this section, we briefly review an important GKE protocol proposed recently. In Section 2, we present a forward secure encryption scheme with constant size ciphertext. Section 3 introduces the model for forward secure GKE protocols. In Section 4, we present a generic one round GKE protocol. Finally, we conclude the paper in Section 5. Preliminaries and proofs of both the protocol and the forward secure encryption scheme are given in the appendix.

1.2 Related Work

Wu et al. [45] recently proposed a one round *asymmetric* GKE protocol. This protocol is different from the conventional GKE protocols in that the parties agree upon a common public key instead of a shared symmetric key. Each user computes a private key corresponding to the common public key. Similar to the Diffie-Hellman protocol [23], their protocol is designed from *scratch* i.e. without assuming prior infrastructure. However, it is well known that the security of such protocols depends on the assumption of an authenticated network. The protocol of Wu et al. can be made secure against active adversaries by applying Katz and Yung's [34] adaptation of Bellare et al.'s approach [6] to the group case, which introduces an extra round and $O(n^2)$ messages for a group of size n. On the other hand, the BG protocol has only one communication round and can be shown secure against active adversaries. But, it assumes the existence of public key infrastructure. In this paper, we concentrate on conventional GKE protocols as the main idea behind a key exchange protocol (two-party or group) is to establish a symmetric key and thereby leverage the efficiency of symmetric cryptographic primitives for further communication.

2 A Forward Secure Encryption Scheme

The initial key evolving schemes of Back [4] and Anderson [3] result in private keys with linear size complexity in the total number of time periods. Bellare and Miner [7] proposed a key evolving scheme using a binary tree technique, which has private keys of logarithmic size complexity. In their scheme, the time periods are associated to the leaves of a binary tree in an *in-order* traversal.

Canetti et al. [20] further improved the efficiency of this binary tree based key evolving scheme. The time periods are now associated to all the nodes of the binary tree in a *pre-order* traversal. This improves the efficiency of the key generation and key updating phases of the key evolving scheme.

In this section, we present a new forward secure encryption scheme from the hierarchical ID-based encryption scheme of Boneh et al. [11] using the key evolving approach of Canetti et al. Unlike Canetti et al., we directly construct the forward secure encryption scheme without proposing an intermediate *binary tree encryption* scheme. In the following, we use the terms key evolving public key encryption and forward secure encryption interchangeably.

2.1 Key Evolving Public Key Encryption

The definition of a key evolving public key encryption (ke-PKE) scheme due to Canetti et al. [20] is presented below. A ke-PKE scheme has four PPT algorithms:

KeyGen takes as input the security parameter k and the total number of time periods N. The output is a public key PK and an initial private key SK_0.
KeyUpd takes as input PK, an index $\tau \in [0, N-1)$ of the current time period and a corresponding private key SK_τ. The output is $SK_{\tau+1}$, the private key for the following time period.
Encrypt takes as input PK, an index $\tau \in [0, N-1]$ of a time period, and a plaintext M. The output is a ciphertext C.
Decrypt takes as input PK, an index $\tau \in [0, N-1]$ of the current time period, a corresponding private key SK_τ and a ciphertext C. The output is either a plaintext M or $\perp$.

For any ke-PKE to be valid it is required that $M = \mathsf{Decrypt}(PK, \tau, SK_\tau, \mathsf{Encrypt}(PK, \tau, M))$, for any message M, any key pair output by KeyGen, any time period $\tau \in [0, N-1]$ and the corresponding SK_τ.

Canetti et al. defined the notions of indistinguishability against chosen-plaintext attacks (fs-IND-CPA) and indistinguishability against chosen-ciphertext attacks (fs-IND-CCA) for forward secure encryption schemes. These notions are reviewed in Appendix B

2.2 A Forward Secure Encryption Scheme with Constant Size Ciphertext

Let N be the number of time periods, labelled 0 through $N-1$. We use a binary tree of depth $l = \lceil \log(N+1) \rceil - 1$ and associate each node in the binary tree with a time period τ, for $\tau \in [0, N-1]$, as follows: The root is labelled with the empty string ϵ. If a node has a label the binary string w then its left child is labelled $w0$ and its right child $w1$ i.e., appending w with either 0 or 1. Let w^τ denote the node of the binary tree corresponding to a time period τ. The time periods are associated to the binary tree nodes in a pre-order traversal as described below:

- $w^0 = \epsilon$ (i.e. the root the tree)
- If w^τ is an internal node then $w^{\tau+1} = w^\tau 0$.
- If w^τ is a leaf node and $\tau < N-1$ then $w^{\tau+1} = w'1$, where w' is the longest string such that $w'0$ is a prefix of w^τ.

As stated earlier, the public key of the scheme remains the same throughout its lifetime. The private key SK_w corresponding to a node w in the binary tree is derived from the private key of its parent node. The private key SK_τ of the party in the time period τ (τ is associated with w^τ) contains the private key SK_{w^τ} of the node w^τ and also the private keys of all the right siblings of the nodes on the path from the root to the node w^τ. Note that SK_τ is the private key of the party in time period τ, whereas SK_{w^τ} is part of SK_τ and serves as the decryption key in the time period τ. The pre-order traversal of the binary tree requires a *stack* (e.g. an array or a list, from which elements are removed in the reverse order to the order of their insertion). In any time period, the private key of the party is a set of all the keys stored in the stack, with the current decryption key at the top.

A binary tree for $N = 7$ is shown in Figure 1. The elements of the stack at time period 2 are also shown in the figure. Note that the current decryption key SK_{00} associated with the node 00 is at the top of the stack. The key update algorithm is explained in more detail below. We now describe the proposed forward secure encryption scheme.

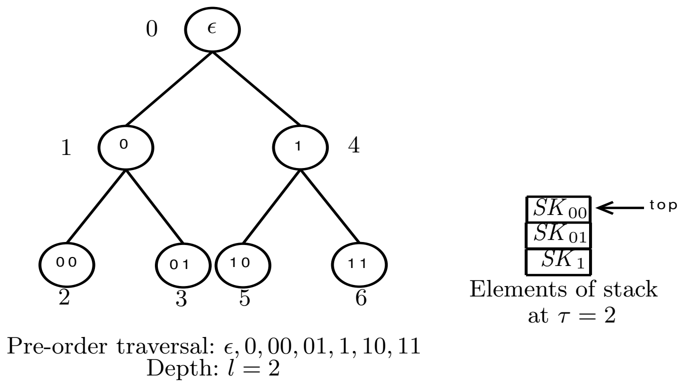

Fig. 1. A binary tree construction for $N = 7$

KeyGen(k, N) Let $\mathbb{G}_0$ be a bilinear group of prime order p and let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a bilinear map as described in Appendix A.1. Note that a node associated to a time period τ is labelled with a binary string w^τ. We use a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$.

1. Select a random generator g of $\mathbb{G}_0$, a random $\alpha \in \mathbb{Z}_p$ and compute $g_1 = g^\alpha$.
2. Select $g_2, g_3, h_0, h_1, \ldots, h_l \overset{R}{\leftarrow} \mathbb{G}_0$.
3. The public key PK is $(\mathbb{G}_0, \mathbb{G}_1, e, g, g_1, g_2, g_3, h_0, h_1, \ldots, h_l, H)$, where $l = \lceil \log(N+1) \rceil - 1$.
4. Choose $r \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute $I_0 = H(\epsilon)$.
5. The initial private key is computed as $SK_\epsilon = (g_2^\alpha \cdot ((h_0)^{I_0} \cdot g_3)^r, g^r, h_1^r, \ldots, h_l^r)$.

KeyUpd(PK, τ, SK_τ) As described above, the nodes are labelled in such a way that all the nodes in the path from root to the node w^τ have labels which are prefixes of w^τ (including ϵ for the root). The length of the label w^τ would be v, where v is the depth of the node labelled w^τ. Let the $(v+1)$ prefixes of w^τ be $w_0^\tau, \ldots, w_v^\tau$ where $w_0^\tau = \epsilon$ and $w_v^\tau = w^\tau$ and let $I_0, \ldots, I_v$ be the corresponding hash values.
Recall that the private key in time period τ contains the private key associated to the node w^τ and also the private keys associated to the right siblings of the nodes in the path from root to w^τ. We first pop the current decryption key SK_{w^τ}, which is at the top, off the stack. The private key for time period $\tau + 1$ is derived as follows:

1. If w^τ is a leaf node, the next key on the stack (top of the stack) is set as $SK_{w^{\tau+1}}$.
2. If w^τ is an internal node, then the key SK_{w^τ} is used to compute the private keys of its two children as follows:

(a) Note that the private key SK_{w^τ} is of the form $(g_2^\alpha \cdot (h_0^{I_0} \cdots h_v^{I_v} \cdot g_3)^r, g^r, h_{v+1}^r, \ldots, h_l^r)$ for $v \le l$. Let this be $(a_0, a_1, b_{v+1}, \ldots, b_l)$.
(b) Select $t_0, t_1 \xleftarrow{R} \mathbb{Z}_p$
(c) Compute $I'_v = H(w^\tau 0)$ and $I''_v = H(w^\tau 1)$.
(d) Compute $SK_{w^\tau 0} = (a_0 \cdot b_{v+1}^{I'_v} \cdot ((h_0)^{I_0} \cdots h_{v+1}^{I'_v} \cdot g_3)^{t_0}, a_1 \cdot g^{t_0}, b_{v+2} \cdot h_{v+2}^{t_0}, \ldots, b_l \cdot h_l^{t_0})$
(e) Compute $SK_{w^\tau 1} = (a_0 \cdot b_{v+1}^{I''_v} \cdot ((h_0)^{I_0} \cdots h_{v+1}^{I''_v} \cdot g_3)^{t_1}, a_1 \cdot g^{t_1}, b_{v+2} \cdot h_{v+2}^{t_1}, \ldots, b_l \cdot h_l^{t_1})$
(f) Push $SK_{w^\tau 1}$ and then $SK_{w^\tau 0}$ into the stack with the decryption key $SK_{w^\tau 0}$ at the top of the stack.

3. In either case above, the key SK_{w^τ} is erased. The stack of keys is returned as the private key for the time period $\tau + 1$.

Encrypt(PK, τ, M) To encrypt a message $M \in \mathbb{G}_1$ under the public key PK and to be decrypted in time period τ we do the following:

1. Select random $s \in \mathbb{Z}_p$.
2. Compute ciphertext $CT = \left(e(g_1, g_2)^s \cdot M, g^s, (h_0^{I_0} \cdots h_v^{I_v} \cdot g_3)^s\right)$

Decrypt(SK_τ, CT) Let the ciphertext be (X, Y, Z) and the private key be $(a_0, a_1, b_{v+1}, \ldots, b_l)$ output

$$X \cdot e(a_1, Z)/e(Y, a_0) = M$$

Scheme parameters. The ciphertext size in the above scheme remains constant for all the time periods i.e., it always contains two elements of $\mathbb{G}_0$ and one element of $\mathbb{G}_1$. A single decryption key is of size $O(l)$ and the length of the stack is of size $O(l)$. Hence, the private key in any time period is of size $O(l^2)$. But, as noticed by Boneh et al. [11], the private key size can be reduced to $O(l)$ by employing $O(l^2)$ public storage. The users may also agree that the parameters $h_0, \cdots, h_l$ be made common throughout the system, making the size of individual public keys constant. In this case, all the users will have to use the same number of time periods.

Theorem 1. *The proposed fs-PKE scheme is secure under the fs-IND-CPA notion assuming the hardness of decisional $(l+1)$-wBDHI* problem. The advantage of an fs-IND-CPA adversary $\mathcal{A}$ is upper bounded by $N \cdot \lambda$, where λ is the advantage of decisional $(l+1)$-wBDHI* problem solver $\mathcal{B}$ and N is the number of time periods.*

The proof of above theorem is given in Appendix D.

The scheme can be made fs-IND-CCA secure in the standard model using the generic technique of Canetti et al. [21]. Appendix E gives some more details.

3 Forward Secure Group Key Exchange

We first give a definition of forward secure GKE protocol, which involves three PPT algorithm:

KeyGen takes as input the security parameter k and the total number of time periods N. The output is a public key PK and an initial private key SK_0.

KeyUpd takes as input PK, an index $\tau \in [0, N-1)$ of the current time period, a corresponding private key SK_τ. The output is the private key $SK_{\tau+1}$ for the following time period.

KeyEx is an interactive algorithm that takes as input the public key PK, the private key for the current period SK_τ, the identities and public keys of the peers and the incoming messages. The final output is either a session key κ or the $\perp$ symbol.

We now describe the security model for a forward secure GKE protocol. The adversarial capabilities are similar to those in the earlier models for GKE protocols [17, 34].

3.1 Adversarial Model

Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n parties and let N be the total number of time periods that the protocol can support. The protocol may be run among any subset of $\tilde{\mathcal{U}} \subseteq \mathcal{U}$ containing $\tilde{n} \leq n$ parties. A GKE protocol π executed among $\tilde{n}$ users is modelled as a collection of $\tilde{n}$ programs running at the $\tilde{n}$ parties in $\tilde{\mathcal{U}}$. Each instance of π within a party is defined as a session and each party may have multiple such sessions running concurrently. Each party $U_i \in \mathcal{U}$ initially generates a pair of long-term public and private keys, (PK^i, SK^i_0) using the KeyGen algorithm during an initialization phase prior to the protocol run. When entering a time period τ, it updates its long-term private key to SK^i_τ using the KeyUpd algorithm.

Let $\pi_i^{j,\tau}$ be the j-th run of the protocol π at party $U_i \in \tilde{\mathcal{U}}$ during time period τ for $\tau \in [0, N-1]$. We assume that the session ID is derived during the run of the protocol and that the session ID is unique to a protocol run. The session ID of an instance $\pi_i^{j,\tau}$ is denoted by $\mathsf{sid}_i^{j,\tau}$. We assume that each party knows who the other participants are, for each protocol instance. The partner ID $\mathsf{pid}_i^{j,\tau}$ of an instance $\pi_i^{j,\tau}$ is a set of identities of the parties with whom $\pi_i^{j,\tau}$ wishes to establish a common group key. Note that $\mathsf{pid}_i^{j,\tau}$ includes the identity of U_i itself.

An instance $\pi_i^{j,\tau}$ enters an *accepted* state when it computes a session key $sk_i^{j,\tau}$. Note that an instance may terminate without ever entering into an accepted state. The information of whether an instance has terminated with acceptance or without acceptance is assumed to be public. Two instances $\pi_i^{j,\tau}$ and $\pi_{i'}^{j',\tau}$ at two different parties U_i and $U_{i'}$ respectively are considered *partnered* iff (1) both the instances have accepted, (2) $\mathsf{sid}_i^{j,\tau} = \mathsf{sid}_{i'}^{j',\tau}$ and (3) $\mathsf{pid}_i^{j,\tau} = \mathsf{pid}_{i'}^{j',\tau}$.

The communication network is assumed to be fully controlled by an adversary $\mathcal{A}^\pi$, which schedules and mediates the sessions among the parties. $\mathcal{A}^\pi$ is allowed to insert, delete or modify the protocol messages. If $\mathcal{A}^\pi$ honestly forwards the protocol messages among the parties, then all the instances are partnered and they output identical session keys. Such a protocol is called a correct GKE protocol. In addition to controlling the message transmission, $\mathcal{A}^\pi$ is allowed to ask the following queries.

- $\mathsf{Execute}(\mathsf{pid},\tau)$ prompts a complete execution of the protocol among the parties in pid during the time period τ. A unique session ID sid is established during the run of the protocol. $\mathcal{A}^\pi$ is given all the protocol messages, modelling passive attacks.
- $\mathsf{Send}(\pi_i^{j,\tau},m)$ sends a message m to the instance $\pi_i^{j,\tau}$. If the message is (pid, τ), the instance $\pi_i^{j,\tau}$ is initiated with partner ID pid. The response of $\pi_i^{j,\tau}$ to any Send query is returned to $\mathcal{A}^\pi$.
- $\mathsf{RevealKey}(\pi_i^{j,\tau})$ If $\pi_i^{j,\tau}$ has accepted, $\mathcal{A}^\pi$ is given the session key $sk_i^{j,\tau}$ established at $\pi_i^{j,\tau}$.
- $\mathsf{Corrupt}(U_i, \tau)$ The evolved long-term secret key SK^i_τ of U_i for the time period τ is returned to $\mathcal{A}^\pi$. Note that the key may have to be computed by repeated application of the KeyUpd algorithm.
- $\mathsf{Test}(\pi_i^{j,\tau})$ A random bit b is secretly chosen. If $b = 1$, $\mathcal{A}^\pi$ is given $K_1 = sk_i^{j,\tau}$ established at $\pi_i^{j,\tau}$. Otherwise, a random value K_0 chosen from the session key probability distribution is given. Note that a Test query is allowed only on an accepted instance.

3.2 AKE Security

We assume that the protocol participants execute the protocol honestly i.e., the adversary is an outsider. We now define the notion of *freshness*, which is central to the definition of AKE Security.

Freshness. An instance $\pi_i^{j,\tau}$ is fresh if the following conditions hold:

1. neither the instance $\pi_i^{j,\tau}$ nor any of its partners have been asked a $\mathsf{RevealKey}$ query after their acceptance;
2. there has been no $\mathsf{Corrupt}(U_{i'}, d)$ issued for any $U_{i'} \in \mathsf{pid}_i^{j,\tau}$ (including $U_{i'} = U_i$) and $d \leq \tau$.

Definition 1 (AKE-Security). An adversary $\mathcal{A}^\pi$ against the AKE-security notion is allowed to make $\mathsf{Execute}$, Send, $\mathsf{RevealKey}$ and $\mathsf{Corrupt}$ queries in Stage 1. $\mathcal{A}^\pi$ makes a Test query to an instance $\pi_i^{j,\tau}$ at the end of Stage 1 and is given a challenge key K_b as described earlier. It can continue asking queries in Stage 2. Finally, $\mathcal{A}^\pi$ outputs a bit b' and wins the AKE security game if (1) $b' = b$ **and** (2) the instance $\pi_i^{j,\tau}$ that was asked Test query remained fresh till the end of $\mathcal{A}^\pi$'s execution. Let $\mathsf{Succ}_{\mathcal{A}^\pi}$ be the success probability of $\mathcal{A}^\pi$ in winning the AKE security game. The advantage of $\mathcal{A}^\pi$ in winning this game is $\mathsf{Adv}_{\mathcal{A}^\pi} = |2 \cdot \Pr[\mathsf{Succ}_{\mathcal{A}^\pi}] - 1|$. A protocol is called AKE-secure if $\mathsf{Adv}_{\mathcal{A}^\pi}$ is negligible in the security parameter k for any polynomial time $\mathcal{A}^\pi$.

3.3 Discussion

A forward secure GKE protocol should ensure that the session key established in a session during a given time period τ is independent of the other session keys established during τ and as well as session keys established during time periods before or after τ. This is modelled by allowing the adversary to reveal any session key except the one in the Test session. The protocol should also ensure that revealing a long-term key in time period τ should not compromise the session keys established in time periods prior to τ. This is modelled by allowing $\mathcal{A}^\pi$ to corrupt long-term keys in time periods $d > \tau$.

Note that if the adversary corrupts the long-term private key in time period τ, then all the session keys established during τ will be compromised. Hence, a GKE protocol secure under our model offers a somewhat weaker security guarantee than the protocols secure under the WFS notion. However, if we use a long-term key SK_τ for only one execution of the GKE protocol i.e., with the key evolving for each protocol run, then the security offered by a protocol secure in the above definition is identical to the conventional forward secrecy considered for GKE protocols with fixed long-term private keys.

In the above model, we have assumed that the number of time periods is the same for all the parties and that all the parties begin/end a time period simultaneously. We can easily extend the model to the case where the number and length of the time periods may differ from one party to the other. This allows a party in time period τ to establish a session key with parties in time period z. In practice, this makes sense only if τ and z have some overlapped window. In this case, we will have to assume that each party knows the time intervals of all other parties in the protocol.

4 One Round GKE Protocol with Forward Security

In this section, we first give a sketch of a one round GKE protocol with forward security that can be constructed from existing GKE protocols [19, 35, 9]. However, in this construction the size of the system parameters increase linearly with the number of time periods. We then use the BG protocol [14] and replace a normal public key encryption scheme in their protocol with a forward secure encryption scheme. We show that this construction achieves our notion of AKE-security. Note that this is a generic protocol and works with any instantiation of a forward secure encryption scheme. To compare our construction with existing GKE protocols, we consider the forward secure encryption scheme in Section 2.2 for efficiency reasons.

4.1 A Protocol with Linear Complexity

A simple one round GKE protocol with forward security can be constructed by combining the earlier key evolving approach [4, 3] with existing GKE protocols [19, 35, 9]. We first briefly review the key evolving approach and the protocol of Burmester and Desmedt [19] and then describe how these two can be combined to derive a one-round GKE protocol with forward security.

Let N be the number of time periods. Any party using the key evolving scheme initially generates N pairs of keys $(sk_0, pk_0), \ldots, (sk_{N-1}, pk_{N-1})$. The public key of the system is $PK = \{pk_0, \ldots, pk_{N-1}\}$. The private key at time period τ is $SK_\tau = SK_{\tau-1} \setminus \{sk_{\tau-1}\}$ for $\tau \in [1, N-1]$, where $SK_0 = \{sk_0, \ldots, sk_{N-1}\}$. The key $sk_{\tau-1}$ is erased after computing SK_τ.

Note that in the Burmester and Desmedt [19] protocol, the parties are organized in a logical ring with assumption that each party knows who its neighbours are. The protocol takes two rounds to establish a session key among the parties. In the first round each party chooses an ephemeral public-private key pair and broadcasts the ephemeral public key to other parties. In the second round each party computes a pair-wise ephemeral Diffie-Hellman key with its two neighbours and broadcasts their ratio to all other parties. The session key is then computed by each party using its own ephemeral private key, the ephemeral Diffie-Hellman keys and the messages sent in second round.

To construct a one-round forward secure GKE protocol from Burmester and Desmedt protocol, we simply eliminate the first round. Note that each party is assumed to know the fixed public key of all the other parties. We use the public key component pk_τ of a time period τ as the ephemeral public key during that time period. Each party establishes a pair-wise non-interactive key for the time period τ with its neighbours. The protocol continues normally in the second round using the non-interactive keys as the ephemeral Diffie-Hellman keys. This protocol may also be seen as a generic construction of a one-round forward secure GKE protocol from a forward secure non-interactive two-party key exchange. A similar approach can also be applied to the recent protocols [35, 9], which are based on Burmester and Desmedt's protocol. However, a major disadvantage of this approach is that the size of the public and private keys increases linearly with the number of time periods.

4.2 A One-round Forward Secure GKE Protocol

Boyd and González Nieto (BG) [14] proposed a one-round GKE protocol by employing public key encryption. Choo et al. [22] later showed that their protocol was not secure against unknown key share attacks. They also suggested an improvement to this protocol but did not provide a proof of security. We use the improved BG protocol and then replace the public key encryption primitive with forward secure encryption. When we instantiate this generic protocol with the proposed forward secure encryption scheme, it results in a concrete one-round forward secure GKE protocol with private keys of $O(\log(N))$ size and constant size protocol messages.

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_{\tilde{n}}\}$ be the set of users who want establish a session key among themselves. We assume that the users agree upon a distinguished user for each execution of the protocol. Note that this user does not have to be fixed. Without loss of generality let U_1 be the distinguished user. The protocol uses a forward secure encryption scheme $\mathcal{FSE} = (\mathcal{K}_g, \mathcal{K}_u, \mathcal{E}, \mathcal{D})$, where $\mathcal{K}_g$, $\mathcal{K}_u$, $\mathcal{E}$ and $\mathcal{D}$ are the KeyGen, KeyUpd, $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ algorithms respectively. It also uses a signature scheme $\Sigma = (\mathcal{K}_s, \mathcal{S}, \mathcal{V})$, where $\mathcal{K}_s$, $\mathcal{S}$ and $\mathcal{V}$ are the key generation, signature and verification algorithms respectively. Each user initially generates a key pair for each of the schemes. For the forward secure encryption the parties update their long-term private key using the $\mathcal{K}_u$ algorithm, while we assume Σ to be a standard public key signature scheme.

In the protocol, the distinguished user U_1 chooses a nonce $N_1 \stackrel{R}{\leftarrow} \{0,1\}^k$ and encrypts it along with its identity for all the other parties using their respective public keys and the current time period τ. U_1 signs all these ciphertexts together with the set of identities of all the users $\mathcal{U}$. The set $\mathcal{U}$, the signature generated and the ciphertexts are then broadcast. All the other parties $U_i \in \mathcal{U}$ for $1 < i \leq \tilde{n}$ broadcast their nonces $N_i \stackrel{R}{\leftarrow} \{0,1\}^k$ along with their identities. Each party computes the session ID as the concatenation of all the outgoing and incoming protocol messages. Each U_i first verifies the signature of U_1 on the incoming message and then decrypts the corresponding ciphertext using its long-term private key for the time period τ to obtain N_1. A pseudo random function f is used to derive the session key with the nonce N_1 as the random seed and the session ID as input. As there is no restriction on who should send a protocol message first, the protocol can be completed in one round. Figure 2 presents the protocol message transmission and session key computation.

Round 1

$$U_1 \rightarrow * : \mathcal{U} = \{U_1, U_2, \ldots, U_{\tilde{n}}\}, \mathcal{S}_{sk_{s_1}}(\mathcal{U}, \mathcal{E}(PK^2, \tau, (N_1, U_1)), \ldots, \mathcal{E}(PK^{\tilde{n}}, \tau, (N_1, U_1)))$$

$$U_1 \rightarrow * : \mathcal{E}(PK^i, \tau, (N_1, U_1)) \text{ for } 1 < i \leq \tilde{n}$$

$$U_i \rightarrow * : U_i, N_i \text{ for } 1 < i \leq \tilde{n}$$

Key Computation

$$sid = \mathcal{U} \| \mathcal{S}_{sk_{s_1}}(\mathcal{U}, \mathcal{E}(PK^2, \tau, (N_1, U_1)), \ldots, \mathcal{E}(PK^{\tilde{n}}, \tau, (N_1, U_1))) \| \mathcal{E}(PK^i, \tau, (N_1, U_1)) \| U_i \| N_i$$

The session key is $\kappa = f_{N_1}(sid)$

Fig. 2. A one round GKE protocol with forward security

Theorem 2. *The proposed protocol is secure under the AKE security notion in Definition 1 in the standard model assuming the underlying fs-PKE is fs-IND-CCA secure and the signature scheme is existentially unforgeable. The advantage of a polynomial adversary* $\mathcal{A}^{\pi}$ *against the AKE-security of the protocol is upper bounded by*

$$\tilde{n} \cdot Adv_{\mathcal{A}^{CMA}} + \frac{\tilde{n} \cdot q_s^2}{2^k} + N \cdot q_s(\tilde{n} \cdot (\tilde{n} - 1) \cdot Adv_{\mathcal{A}^{CCA}} + Adv_{\mathcal{A}^{PRF}})$$

where k *is the security parameter,* $\tilde{n}$ *is the number of protocol participants,* N *is the number of time periods,* q_s *is the number of sessions* $\mathcal{A}^{\pi}$ *is allowed to activate,* $Adv_{\mathcal{A}^{CMA}}$ *is the advantage of a polynomial adversary against the existential unforgeability of the signature scheme,* $Adv_{\mathcal{A}^{CCA}}$ *is the advantage of a polynomial adversary against the fs-IND-CCA security of the encryption scheme and* $Adv_{\mathcal{A}^{PRF}}$ *is the advantage of a polynomial adversary against the pseudo random function.*

The proof of the above theorem is in Appendix C.

5 Conclusion

Table 1 gives a comparison among existing GKE protocols. All the terms in the table are self-explanatory except the ones with "*" and "#".

The BG protocol has only one round but it does not provide forward secrecy. Moreover, till now the protocol is known to be secure only in the random oracle model. Our modification to their protocol enables it to achieve forward security. More importantly, the proof is given in the standard model. The entry "Std.*" indicates that from the proof of Theorem 2, it now implies that the BG protocol is secure in the standard model, assuming that the underlying encryption scheme is chosen ciphertext secure. The entry "Yes*" in the table indicates that our protocol has forward security. But, as discussed earlier, it can also achieve a level of security that is identical to forward secrecy.

Our protocol has private key of size $O(\log(N))$, where N is the number of time periods, whereas all other protocols have constant size private keys. To compare more concretely, let us assume that

	Rounds	Forward Secrecy?	Private key Size	Model	Message Size	Insider Security
BG Protocol [14]	1	No	$O(1)$	Std.*	$O(1)$	No
Katz and Yung [34]	3	Yes	$O(1)$	Std.	$O(1)$	No
Bohli et al. [9]	2	Yes	$O(1)$	ROM	$O(1)$	Yes
Bresson and Manulis [18]	3	Yes	$O(1)$	Std.	$O(1)$	Yes
Furukawa et al. [24]	2	Yes	$O(1)$	Std.	$O(1)$	Yes$^{\#}$
Gorantla et al. [28]	1	No	$O(1)$	Std.	$O(1)$	No
Our Protocol	1	Yes*	$O(\log(N))$	Std.	$O(1)$	No

Table 1. Security and efficiency comparison among existing GKE protocols

we need a GKE protocol that provides 128-bit security level. As most of the existing GKE protocols use signature-based authenticators, we compare the size of the signature key with the size of the private keys used in our protocol. Note that we need a private key each for the underlying forward secure encryption scheme and the signature scheme. If RSA is employed for signatures, for 128-bit level security, we typically need 3072-bit RSA key. Let us assume that the certificate corresponding to this signing key is revoked every 5 years. In our scheme, we can divide the total period of 5 years into (roughly) 5*365 = 1825 individual time periods with the private key evolving at the start of each day i.e., $N = 1825$. In the forward secure encryption scheme in Section 2.2, the size of the private key contains at most $\lceil \log(N+1)+2 \rceil$ elements of the group $\mathbb{G}_0$. For 128-bit level security we need the elements of $\mathbb{G}_0$ to be of size at least 256 bits [25]. Hence, the size of the private key of the forward secure encryption scheme is $\lceil \log(N+1)+2 \rceil * 256 = \lceil \log(1826)+2 \rceil * 256 = 3328$. The combined size of the private keys of forward secure encryption scheme and the signature scheme is 6400 , which compares well with the private key size of a signature-based GKE protocol. Hence, our protocol remains practical for a reasonable number of time periods.

Our protocol is not secure under the insider security notions of mutual authentication and contributiveness. However, it can be made insider secure at additional computational and communication costs. The protocol may be modified such that instead of a single party encrypting its nonce, all the parties encrypt their nonces to other parties and the Katz and Shin [33] compiler is then be applied on top it. We speculate that the resulting two-round protocol will have mutual authentication and contributiveness in the presence of insiders. The entry "Yes$^{\#}$" in the table indicates that the protocol of Furukawa et al. was not shown secure under the notion of contributiveness. Moreover, this protocol was proven insider secure in the universal composability framework.

As shown in the table, our protocol achieves an enhanced security property in the form of forward security, compared to previously known one round GKE protocols without introducing any additional rounds. Apart from being a one round protocol, note that our protocol has constant size protocol messages like other GKE protocols. This is possible only if our protocol is instantiated with the forward secure encryption scheme presented in this paper, which has constant size ciphertext.

References

1. Abdalla, M., Miner, S.K., Namprempre, C.: Forward-Secure Threshold Signature Schemes. In Naccache, D., ed.: Topics in Cryptology–CT-RSA 2001. Volume 2020 of LNCS., Springer (2001) 441–456
2. Al-Riyami, S.S., Paterson, K.G.: Tripartite Authenticated Key Agreement Protocols from Pairings. In: Cryptography and Coding, 9th IMA International Conference. Volume 2898 of LNCS., Springer (2003) 332–359
3. Anderson, R.: Two Remarks on Public Key Cryptology. Technical Report 549, Computer Laboratory, University of Cambridge (2002) Available at `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-549.pdf`.
4. Back, A.: Non-interactive forward secrecy (1996) posting to cypherpunks mailing list (6 Sep 1996),`http://www.cypherspace.org/adam/nifs/`. Last Accessed on 17 Jan 2010.

5. Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In Preneel, B., ed.: Advances in Cryptology–EUROCRYPT'00. Volume 1807 of LNCS., Springer (2000) 259–274
6. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract). In: STOC. (1998) 419–428
7. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Advances in Cryptology–CRYPTO '99. Volume 1666 of LNCS., Springer (1999) 431–448
8. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proc. of the 27th Annual ACM Symposium on Theory of Computing–STOC'95, ACM (1995) 57–66
9. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure group key establishment revisited. Int. J. Inf. Sec. **6**(4) (2007) 243–254
10. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Advances in Cryptology–EUROCRYPT 2004. Volume 3027 of LNCS., Springer (2004) 223–238
11. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Advances in Cryptology–EUROCRYPT 2005. Volume 3494 of LNCS., Springer (2005) 440–456
12. Boneh, D., Silverberg, A.: Applications of Multilinear Forms to Cryptography. In: Conferences in memory of Ruth Michler. Volume 324 of Contemporary Mathematics., American Mathematical Society (2003) 71–90
13. Boyd, C., Cliff, Y., Gonzalez Nieto, J., Paterson, K.G.: Efficient One-Round Key Exchange in the Standard Model. In: Information Security and Privacy–ACISP'08. Volume 5107 of LNCS., Springer (2008) 69–83
14. Boyd, C., González Nieto, J.M.: Round-Optimal Contributory Conference Key Agreement. In: Public Key Cryptography–PKC'03. Volume 2567 of LNCS., Springer (2003) 161–174
15. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Advances in Cryptology–ASIACRYPT'01. Volume 2248 of LNCS., Springer (2001) 290–309
16. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Advances in Cryptology–EUROCRYPT'02. Volume 2332 of LNCS., Springer (2002) 321–336
17. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably authenticated group Diffie-Hellman key exchange. In: CCS'01: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM (2001) 255–264
18. Bresson, E., Manulis, M.: Securing Group Key Exchange against Strong Corruptions. In: Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'08), ACM Press (2008) 249–260
19. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System (Extended Abstract). In: Advances in Cryptology–EUROCRYPT'94. (1994) 275–286
20. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In Biham, E., ed.: Advances in Cryptology–EUROCRYPT 2003. Volume 2656 of LNCS., Springer (2003) 255–271
21. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Advances in Cryptology–EUROCRYPT 2004. Volume 3027 of LNCS., Springer (2004) 207–222
22. Choo, K.K.R., Boyd, C., Hitchcock, Y.: Errors in computational complexity proofs for protocols. In: Advances in Cryptology–ASIACRYPT 2005. Volume 3788 of LNCS., Springer (2005) 624–643
23. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory **IT-22**(6) (1976) 644–654
24. Furukawa, J., Armknecht, F., Kurosawa, K.: A Universally Composable Group Key Exchange Protocol with Minimum Communication Effort. In: Security and Cryptography for Networks–SCN'08. Volume 5229 of LNCS., Springer (2008) 392–408
25. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16) (2008) 3113–3121
26. Gennaro, R., Krawczyk, H., Rabin, T.: Okamoto-Tanaka Revisited: Fully Authenticated Diffie-Hellman with Minimal Overhead. Cryptology ePrint Archive, Report 2010/068 (2010) `http://eprint.iacr.org/`.
27. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In: Public Key Cryptography–PKC'09. Volume 5443 of LNCS., Springer (2009) 105–123
28. Gorantla, M.C., Boyd, C., González Nieto, J.M., Manulis, M.: Generic One Round Group Key Exchange in the Standard Model. In: 12th International Conference on Information Security and Cryptology–ICISC 2009, Springer (2009) Available at `http://eprint.iacr.org/2009/514`.
29. Ingemarsson, I., Tang, D.T., Wong, C.K.: A conference key distribution system. IEEE Transactions on Information Theory **28**(5) (1982) 714–719
30. Itkis, G.: Forward security (adaptive cryptography: time evolution). Handbook of Information Security (2006) Available at `http://www.cs.bu.edu/~itkis/pap/forward-secure-survey.pdf`.
31. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Algorithmic Number Theory, 4th International Symposium. Volume 1838 of LNCS., Springer (2000) 385–394
32. Katz, J.: Binary Tree Encryption: Constructions and Applications. In: Information Security and Cryptology–ICISC 2003. Volume 2971 of LNCS., Springer (2003) 1–11

33. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security–CCS'05, ACM (2005) 180–189
34. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Advances in Cryptology–CRYPTO'03. Volume 2729 of LNCS., Springer (2003) 110–125
35. Kim, H.J., Lee, S.M., Lee, D.H.: Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In: Advances in Cryptology–ASIACRYPT'04. Volume 3329 of LNCS., Springer (2004) 245–259
36. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Advances in Cryptology–CRYPTO'05. Volume 3621 of LNCS., Springer (2005) 546–566
37. Krawczyk, H.: Simple forward-secure signatures from any signature scheme. In: ACM Conference on Computer and Communications Security. (2000) 108–115
38. Law, L., Menezes, A., Qu, M., Solinas, J.A., Vanstone, S.A.: An Efficient Protocol for Authenticated Key Agreement. Des. Codes Cryptography **28**(2) (2003) 119–134
39. Manulis, M.: Provably Secure Group Key Exchange. Volume 5 of IT Security. Europäischer Universitätsverlag, Berlin, Bochum, Dülmen, London, Paris (August 2007)
40. Matsumoto, T., Takashima, Y., Imai, H.: On seeking smart public-key distribution systems. Trans. IECE of Japan **E69** (1986) 99–106
41. Okamoto, T., Tso, R., Okamoto, E.: One-Way and Two-Party Authenticated ID-Based Key Agreement Protocols Using Pairing. In: Modeling Decisions for Artificial Intelligence, 2nd International Conference–MDAI'05. Volume 3558 of LNCS., Springer (2005) 122–133
42. Sahai, A.: Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In: FOCS. (1999) 543–553
43. Song, D.X.: Practical forward secure group signature schemes. In: ACM Conference on Computer and Communications Security. (2001) 225–234
44. Steiner, M., Tsudik, G., Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication. In: ACM Conference on Computer and Communications Security. (1996) 31–37
45. Wu, Q., Mu, Y., Susilo, W., Qin, B., Domingo-Ferrer, J.: Asymmetric Group Key Agreement. In: Advances in Cryptology–EUROCRYPT 2009. Volume 5479 of LNCS., Springer (2009) 153–170

A Preliminaries

We briefly review bilinear pairings and some computational problems on which the security of our ke-PKE scheme is based.

A.1 Bilinear Pairing

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p. Let g be the generator of $\mathbb{G}_0$. The pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ is called an admissible bilinear map if it has the following properties:

Bilinearity: $\forall u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$
Non-degeneracy: $e(g, g) \neq 1$
Computable: There exists an efficient algorithm to compute $e(g, g)$

A.2 Weak BDHI Assumption

Boneh et al. [11] introduced a weaker variation of the l-th bilinear Diffie-Hellman inversion problem (l-BDHI) [10], called weak bilinear Diffie-Hellman inversion problem and denoted by l-wBDHI. It is as follows:

Let g and h be two random generators of $\mathbb{G}_0$. Let α be a random number in $\mathbb{Z}_p{}^*$. The l-wBDHI problem is defined as:

$$l\text{-wBDHI: given } g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^l)} \text{ compute } e(g, h)^{1/\alpha}$$

Boneh et al. also defined another problem l-wBDHI* which is equivalent to l-wBDHI in linear time reduction.

$$l\text{-wBDHI}^*\text{: given } g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^l)} \text{ compute } e(g, h)^{(\alpha^{l+1})}$$

Decisional l-wBDHI* problem The decisional variant of l-wBDHI* problem is defined as follows: Consider the following distributions

$$\mathcal{P}_{wBDHI^*} = \{(g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^l)}, e(g,h)^{(\alpha^{l+1})}) \text{ for } g, h \in \mathbb{G}_0^*, \alpha \in \mathbb{Z}_p^*\}$$

$$\mathcal{R}_{wBDHI^*} = \{(g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^l)}, T) \text{ for } g, h \in \mathbb{G}_0^*, \alpha \in \mathbb{Z}_p^*\, T \in \mathbb{G}_1^*\}$$

An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ω in solving Decision l-wBDHI* if

$$\Pr\left[\mathcal{B}(1^k, \rho) = 1 | \rho \xleftarrow{R} \mathcal{P}_{wBDHI^*}\right] - \Pr\left[\mathcal{B}(1^k, \rho) = 1 | \rho \xleftarrow{R} \mathcal{R}_{wBDHI^*}\right] \geq \omega$$

We say that l-wBDHI* holds in $\mathbb{G}_0$ if ω is negligible in k.

B Definitions of Security for ke-PKE

Canetti et al. [20] defined the notions of indistinguishability against chosen-plaintext attacks (fs-IND-CPA) and indistinguishability against chosen-ciphertext attacks (fs-IND-CCA) for forward secure encryption schemes. We now review these notions.

Definition 2. A ke-PKE is forward-secure against chosen-plaintext attacks (fs-IND-CPA) if the advantage of any PPT adversary in the following game is negligible in the security parameter k for all time periods N polynomial in k.

Setup The challenger runs the KeyGen algorithm and obtains a key pair (PK, SK_0). PK is given to the adversary.

Attack The adversary is allowed to issue one breakin query and one challenge query as described below.

breakin The adversary asks this query with an index $\tau \in [0, N)$ of a time period as input. The challenger (repeatedly) runs the KeyUpd algorithm and returns the output SK_τ to the adversary.

challenge This query is asked with input (z, M_0, M_1) where $z \in [0, N)$ and M_0 and M_1 are two arbitrary messages chosen by the adversary. The challenger selects a random bit b and returns $C^* = \mathsf{Encrypt}(PK, z, M_b)$ to the adversary.

The breakin and challenge queries may be asked in either order with the obvious restriction that $z < \tau$.

Guess The adversary outputs a guess bit b' and it succeeds if $b' = b$. The advantage of the adversary is given as $Adv_{\mathcal{A}^{CPA}} = |2 \cdot \Pr[b' = b] - 1|$.

An analogous definition for fs-IND-CCA notion is given below:

Definition 3. A ke-PKE is forward-secure against chosen-ciphertext attacks (fs-IND-CCA) if the advantage of any PPT adversary in the following game is negligible in the security parameter k for all time periods N polynomial in k.

Setup The challenger runs the KeyGen algorithm and obtains a key pair (PK, SK_0). PK is given to the adversary.

Attack The adversary is allowed to issue one breakin, one challenge and multiple decryption (dec) queries as described below.

– The breakin and challenge queries are answered as in Definition 2.

– On a query $\mathsf{dec}(d, C)$ for $d \in [0, N)$, the appropriate key SK_d is first derived via (repeatedly) running the KeyUpd algorithm. The adversary is then given the output of $\mathsf{Decrypt}(PK, d, SK_d, C)$. If the adversary has already received a response C^* from query $\mathsf{challenge}(z, M_0, M_1)$, then the query $\mathsf{dec}(z, C^*)$ is disallowed. But queries of type $\mathsf{dec}(d, C^*)$ with $d \neq z$ and $\mathsf{dec}(z, C)$ with $C \neq C^*$ are allowed. As in the above definition the $\mathsf{breakin}$ and $\mathsf{challenge}$ queries may be asked in either order with the obvious restriction $z < \tau$.

Guess The adversary outputs a guess bit b' and it succeeds if $b' = b$. The advantage of the adversary is given as $Adv_{\mathcal{A}^{CCA}} = |2 \cdot \Pr[b' = b] - 1|$.

C Security Proof of the Protocol

Note that in our protocol the distinguished party encrypts the same message (N_1, U_1) for the rest of the parties. Hence, the adversary can obtain encryptions of the same plaintext under multiple independent public keys. Bellare and Rogaway [8] called such an adversary as a *multiple eavesdropper* and remarked that the advantage of a multiple adversary is negligible against any secure encryption scheme [8, Lemma 8]. Later, Bellare et al. [5] formalised the security of public key encryption schemes against multiple eavesdroppers. They showed that security in the single user setting (where the adversary gets the encryption of a message under a single public key) implies security in the multi-user setting (where a multiple eavesdropper can obtain the encryption of a message under multiple public keys) as long as the former is secure under IND-CPA or IND-CCA security notions. In our proof, we consider a special case, where the adversary can obtain the encryption of only one message for each public key. The security reduction of Bellare et al. for this special case is stated as below.

Lemma 1 ([5]). *Suppose that Suppose that an adversary has advantage at most $Adv_{\mathcal{A}^{CCA}}$ for an encryption scheme. Then a multiple eavesdropper against the encryption scheme has advantage not more than $r \cdot Adv_{\mathcal{A}^{CCA}}$, where r is the number of public keys under which the message has been encrypted.*

The above lemma can be extended to forward secure encryption schemes that are fs-IND-CPA and fs-IND-CCA secure in a straightforward way. For the case of fs-IND-CPA/fs-IND-CCA secure schemes, the multiple eavesdropper can obtain the encryptions of a message in a time period τ under multiple public keys. For the purpose of our proof, consider the special case, wherein the adversary obtains the encryptions of a single message under multiple public keys to be decrypted in a single time period. Thus the advantage of a multiple eavesdropper against fs-IND-CPA/fs-IND-CCA in such case will be upper bounded by the same advantage stated in Lemma 1. However, note that if the multiple eavesdropper obtains encryptions of multiple messages with each message to be decrypted in different time periods under different public keys, then its advantage will be upper bounded by $q_e \cdot N \cdot r \cdot Adv_{\mathcal{A}^{CCA}}$, where q_e is the number of messages, N is the total number of time periods and r is the number of public keys.

We now give the proof for Theorem 2.

Proof. The proof is given below in a sequence of games. Let σ_i be the event that $\mathcal{A}^\pi$ wins in Game$_i$.

Game 0. This is the original AKE security game. All the queries of $\mathcal{A}^\pi$ are answered as defined in Section 3.2. In particular, when answering the $\mathsf{Corrupt}$ query $\mathcal{A}^\pi$ is given the long-term private key SK_τ of time period τ in which the session is being executed and also the long-term key used for the signature scheme. By definition, we have

$$Adv_{\mathcal{A}^\pi} = |2 \cdot \Pr[\sigma_0] - 1| \tag{1}$$

Game 1. This game is the same as the previous one, except that the simulation fails if $\mathcal{A}^\pi$ outputs a valid forgery of a signature of the distinguished party in any session. Let Forge be such an event. We have

$$|\Pr[\sigma_1] - \Pr[\sigma_0]| \leq \Pr[\mathsf{Forge}] \tag{2}$$

Note that for Forge to have occurred the adversary cannot have corrupted the distinguished party in any time period. If Forge occurs, we can use $\mathcal{A}^\pi$ to forge a signature for a given public key under a chosen message attack as follows: The given public key is assigned to one of the $\tilde{n}$ parties. All other parties are initialized according to the protocol. All queries to the parties can be easily answered by following the protocol specification since all secret keys are known, except the signing key corresponding to the given public key. In the latter case the signing oracle that is available as part of the chosen message attack can be used to simulate answers to $\mathcal{A}^\pi$. The probability of $\mathcal{A}^\pi$ choosing this user as the distinguished user is at least $\frac{1}{\tilde{n}}$. Hence, $Succ_{\mathcal{A}^{CMA}} \geq \frac{\Pr[\mathsf{Forge}]}{\tilde{n}}$. We have

$$\Pr[\mathsf{Forge}] \leq n \cdot Succ_{\mathcal{A}^{CMA}} \tag{3}$$

Game 2. This game is the same as the previous one except that the simulation fails if an instance at a party U_i chooses the same nonce that was chosen by another instance at U_i. Since there can be at most q_s sessions and each nonce is of k-bit length, the probability of this event happening at one party is $\frac{q_s^2}{2^k}$. Since there are $\tilde{n}$ parties in the protocol, we have

$$|\Pr[\sigma_2] - \Pr[\sigma_1]| \leq \frac{\tilde{n} \cdot q_s^2}{2^k} \tag{4}$$

Note that this game excludes replay attacks and accidental collisions that may happen when the parties choose the same nonces that were used in the earlier sessions. $\mathcal{A}^\pi$ may still modify the protocol messages in any session, which may result in the parties in that session not being partners (if any party completes the session).

Game 3. This game is identical to the previous game except that it chooses an index $z \in [0, N-1]$ and proceeds as follows. The game aborts and outputs a random bit if the Test query does not occur during this time period. Let E_3 be the event that this guess is correct.

$$\Pr[\sigma_3] = \Pr[\sigma_3|E_3]\Pr[E_3] + \Pr[\sigma_3|\neg E_3]\Pr[\neg E_3] = \Pr[\sigma_2]\frac{1}{N} + \frac{1}{2}\left(1 - \frac{1}{N}\right) \tag{5}$$

Game 4. This game is identical to the previous game except that it chooses $t \in [1, q_s]$ and proceeds as follows. The game aborts and outputs a random bit if the Test query does not occur in the t-th session. Let E_4 be the event that this guess is correct.

$$\Pr[\sigma_4] = \Pr[\sigma_4|E_4]\Pr[E_4] + \Pr[\sigma_4|\neg E_4]\Pr[\neg E_4] = \Pr[\sigma_3]\frac{1}{q_s} + \frac{1}{2}\left(1 - \frac{1}{q_s}\right) \tag{6}$$

Game 5. This is the same as the previous game except that the pseudo-random function used to derive the session key replaced by a truly random function. We have

$$|\Pr[\sigma_5] - \Pr[\sigma_4]| \leq Adv_{\mathcal{A}^{PRF}} \tag{7}$$

Game 6. This game is the same as the previous game except that the queries asked by $\mathcal{A}^\pi$ are now simulated by an adversary $\mathcal{A}^{CCA}$ against the fs-IND-CCA security of the forward secure encryption scheme.

$\mathcal{A}^{CCA}$ starts by randomly selecting a user from the set of $\tilde{n}$ users. Without loss of generality let this user be U_1. It guesses that this party will play the distinguished user in the Test session. $\mathcal{A}^{CCA}$ generates a key pair for U_1. It obtains the public keys of all the users $\tilde{\mathcal{U}} \setminus \{U_1\}$ from its challenger. Note that $\mathcal{A}^{CCA}$ is allowed to ask breakin, challenge and dec queries in the fs-IND-CCA game. $\mathcal{A}^{CCA}$ randomly chooses two nonces $N_0, N_1 \stackrel{R}{\leftarrow} \{0,1\}^k$ and sets $M_0 = (N_0^* \| U_1)$ and $M_1 = (N_1^* \| U_1)$. It now issues challenge(z, M_0, M_1) to its challenger with respect to the public keys of all users in $\tilde{\mathcal{U}} \setminus \{U_1\}$. $\mathcal{A}^{CCA}$ in return obtains $\alpha_2 = \mathcal{E}_{PK^2}(M_b), \cdots, \alpha_{\tilde{n}} = \mathcal{E}_{PK^{\tilde{n}}}(M_b)$, where $M_b = (N_b^* \| U_1)$ for a $b \in \{0,1\}$ chosen randomly by the challenger. It also generates key pairs for the signature scheme by running the $\mathcal{K}_s$ algorithm. The goal of $\mathcal{A}^{CCA}$, as described in Definition 3 is to guess the bit b. We describe only the interesting cases below, all the other cases can be answered trivially.

Informally, the proof works as below: To gain any advantage from $\mathcal{A}^{\pi}$, $\mathcal{A}^{CCA}$ has to inject the multiples ciphertexts $\alpha_2, \cdots, \alpha_{\tilde{n}}$ into the test session. Note this can be done only when an Execute or a Send query is issued to the test session at U_1. Hence, the adversary simulates these queries on behalf of U_1. If $\mathcal{A}^{\pi}$ gets any advantage at all in winning this game, as explained at the end of simulation $\mathcal{A}^{CCA}$ can turn that into its own advantage. The answers to $\mathcal{A}^{\pi}$'s queries are simulated as follows:

Execute(pid,z): If this is t-th activation, to simulate the messages on behalf of U_1, $\mathcal{A}^{CCA}$ injects $\alpha_2, \cdots, \alpha_n$ as the ciphertexts encrypting its nonce and constructs the protocol message by signing the ciphertexts along with the pid using the signature key corresponding to U_1. Note that the signing key pair of U_1 has been generated by $\mathcal{A}^{CCA}$ itself. It simply follows the protocol specification for all other users. The protocol messages are returned to $\mathcal{A}^{\pi}$.

Send$(\pi_i^{t,z}, m)$: If $U_i = U_1$ and $m = (\mathsf{pid}, z)$, $\mathcal{A}^{CCA}$ injects $\alpha_2, \cdots, \alpha_n$ as described above. Otherwise, $\mathcal{A}^{CCA}$ simply follows the protocol.

In all sessions other than the test session, if the incoming message m in Send$(\pi_i^{j,\tau}, m)$ query contains a ciphertext and if $U_i \neq U_1$, $\mathcal{A}^{CCA}$ queries the decryption oracle dec corresponding to U_i with the input (τ, C), where C is the ciphertext encrypted with the public key of U_i. If dec returns a valid plaintext, $\mathcal{A}^{CCA}$ accepts the session, otherwise the session is terminated. In all cases above, $\mathcal{A}^{CCA}$ records the incoming and/or outgoing messages. In particular, $\mathcal{A}^{CCA}$ stores the nonce that it has either selected or extracted from the incoming ciphertext. The decision of whether the session has been accepted or not is also recorded.

RevealKey$(\pi_i^{j,\tau})$: If $\pi_i^{j,\tau}$ has not been accepted as per the recorded entries, $\mathcal{A}^{CCA}$ returns nothing. Otherwise, $\mathcal{A}^{CCA}$ continues the simulation as follows:

1. If U_i is not the distinguished user in the session,
   (a) If $U_i \neq U_1$, let C be the ciphertext encrypted using the public key of U_i and stored in the recorded entry corresponding to $\pi_i^{j,\tau}$. As $\pi_i^{j,\tau}$ has already been accepted, there must be a entry for the nonce, which was returned as an answer to the dec(τ, C) corresponding to U_i. $\mathcal{A}^{CCA}$ uses this nonce and all other nonces sent in plain to compute the session key and returns it to $\mathcal{A}^{\pi}$.
   (b) If $U_i = U_1$, $\mathcal{A}^{CCA}$ can easily answer this query as it knows the private key of U_i.
2. If U_i is the distinguished user in the session, this query can be trivially answered as $\mathcal{A}^{CCA}$ will have known all the nonces.

Corrupt(U_i, τ): If $U_i \notin \mathcal{U}$, $\mathcal{A}^{CCA}$ could trivially answer this query. If $U_i \neq U_1$ and $\tau \leq z$, $\mathcal{A}^{CCA}$ outputs fail and terminates. Otherwise, $\mathcal{A}^{CCA}$ issues a breakin(τ) to the challenger with respect to U_i and obtains the private key for the period τ. The key is returned to $\mathcal{A}^{\pi}$.

Test$(\pi_1^{t,z})$: $\mathcal{A}^{CCA}$ randomly selects a bit θ and uses the nonce N_θ^* (selected from $\{N_0^*, N_1^*\}$) along with the other nonces sent in plain during the test session to compute a session key K_θ. It returns K_θ to $\mathcal{A}^{\pi}$.

As the simulation by $\mathcal{A}^{CCA}$ is perfect, Game 5 and Game 4 are indistinguishable.

$$\Pr[\sigma_6] = \Pr[\sigma_5] \tag{8}$$

Now, we claim that

$$|2 \cdot \Pr[\sigma_6] - 1| \leq \tilde{n} \cdot (\tilde{n} - 1) \cdot Adv_{\mathcal{A}^{CCA}} \tag{9}$$

$\mathcal{A}^{\pi}$ eventually outputs a bit θ'. If $\theta' = 1$ ($\mathcal{A}^{\pi}$'s guess for real key), then $\mathcal{A}^{CCA}$ returns θ to its challenger. Otherwise, $\mathcal{A}^{CCA}$ returns $1 - \theta$. Hence, the advantage of $\mathcal{A}^{CCA}$ winning the fs-IND-CCA security game with respect all the $(\tilde{n} - 1)$ public keys is at least the same as that of $\mathcal{A}^{\pi}$.
This is true only if $\mathcal{A}^{CCA}$'s guess is correct in that the distinguished party in the test session is U_1. This happens with at least a probability of $\frac{1}{\tilde{n}}$. Hence, from Lemma 1 we have, $(\tilde{n}-1) \cdot Adv_{\mathcal{A}^{CCA}} \geq \frac{|2 \cdot \Pr[\sigma_6]-1|}{\tilde{n}}$.

By combining Equations 1 to 9, we have the claimed advantage for $\mathcal{A}^{\pi}$.

□

D Proof of Theorem 1

Proof. Our proof follows the proof of the hierarchical identity based encryption scheme by Boneh et al. [11]. Let the advantage of $\mathcal{A}$ against the fs-PKE scheme be ϵ. We then construct a $(l+1)$-wBDHI* solver $\mathcal{B}$ running $\mathcal{A}$ as a subroutine, where $l = \lceil \log(N+1) \rceil - 1$ is the depth the binary tree.

For a generator $g \in \mathbb{G}_0$ and $\alpha \in \mathbb{Z}_p^*$, let $y_x = g^{(\alpha^x)} \in \mathbb{G}_0$ for $1 \leq x \leq (l+1)$. $\mathcal{B}$ is given as input a random tuple $(g, h, y_x, \ldots, y_{(l+1)}, T)$ that is sampled from either $\mathcal{P}_{wBDHI^*}$ (in which case $T = e(g,h)^{(\alpha^{l+2})}$) or $\mathcal{R}_{wBDHI^*}$ (in which case $T \stackrel{R}{\leftarrow} \mathbb{G}_1^*$) as described in Section A.2. The goal of $\mathcal{B}$ is to output 1 when the given input is sampled from $\mathcal{P}_{wBDHI^*}$ and 0 otherwise. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and answers its queries as follows:

1. $\mathcal{B}$ starts by selecting a time period τ^*. Let w^* be the node that is associated with τ^*. Let the length of w^* be v, where v is the depth of the node represented by w^*.
2. The $v+1$ prefixes of w^* are $(w_0^*, w_1^*, \ldots, w_v^*)$, where $w_0^* = \epsilon$ and $w_v^* = w^*$. Let $I^* = (I_0^*, I_1^*, \ldots, I_v^*)$ be the corresponding hashes. If $v < l$, then $\mathcal{B}$ adds $l - v$ zeros to I^* on the right to make it a vector of length $l + 1$.
3. $\mathcal{B}$ sets up the public key as follows
   (a) Selects $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets $g_1 = y_1 = g^{\alpha}$, $g_2 = y_{l+1} \cdot g^{\gamma} = g^{\gamma + (\alpha^{l+1})}$.
   (b) Picks $\gamma_0, \ldots, \gamma_l \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets $h_x = g^{\gamma_x}/y_{l-x+1}$ for $0 \leq x \leq l$.
   (c) Picks $\delta \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sets $g_3 = g^{\delta} \cdot \prod_{x=0}^{l} y_{l-x+1}^{I_x}$
   (d) The public key $(\mathbb{G}_0, \mathbb{G}_1, e, g, g_1, g_2, g_3, h_0, h_1, \ldots, h_l, H)$ is given to $\mathcal{A}$.
4. Note that the component $g_2^{\alpha} = g^{\alpha(\alpha^{l+1}+\gamma)} = y_{l+2} \cdot y_1^{\gamma}$ is not known to $\mathcal{B}$ as $\mathcal{B}$ does not have y_{l+2}.
5. When $\mathcal{A}$ asks a $\mathsf{breakin}(\tau)$ query, if $\tau \leq \tau^*$ then $\mathcal{B}$ outputs a random bit and halts. Otherwise $\mathcal{B}$ computes appropriate secret keys as described below and gives them to $\mathcal{A}$.
   (a) Let w^{τ} be the node that is associated with the time period τ. Note that the secret key SK_{τ} is a set containing the secret keys corresponding to the node w^{τ} and also the siblings of the nodes on the path from root to w^{τ} i.e., SK_{τ} is a stack of keys with $SK_{w^{\tau}}$ at the top.

(b) Let the depth of the node w^τ be u. The $u+1$ prefixes of w are $(w_0^\tau, w_1^\tau, \ldots, w_u^\tau)$ and let $(I_0, I_1, \ldots, I_u)$ be the corresponding hashes. We set m such that it is the smallest integer that satisfies the condition $I_m \neq I_m^*$. As $\tau > \tau^*$, w^τ is not a prefix of w^*. Hence, there exists such an m, for $0 \leq m \leq u$.

(c) Note that the secret key SK_{w^m} corresponding to the node w^m, which is at depth m, is of the form:

$$(g_2^\alpha \cdot (h_0^{I_0} \cdots h_m^{I_m} \cdot g_3)^r, g^r, h_{m+1}^r, \ldots, h_l^r) \tag{10}$$

(d) To generate SK_w, $\mathcal{B}$ selects $\tilde{r} \xleftarrow{R} \mathbb{Z}_p$. We assume $r = \frac{\alpha^{m+1}}{I_m - I_m^*} + \tilde{r} \in \mathbb{Z}_p$.

(e) Note that

$$(h_0^{I_0} \cdots h_m^{I_m} \cdot g_3)^r = \left(g^{\delta + \sum_{x=0}^m I_x \gamma_x} \cdot \underbrace{\prod_{x=0}^{m-1} y_{l-x+1}^{I_x^* - I_x}}_{=1} \cdot y_{l-m+1}^{(I_m^* - I_m)} \cdot \prod_{x=m+1}^{l} y_{l-x+1}^{I_x^*} \right)^r \tag{11}$$

$$= \left(g^{\delta + \sum_{x=0}^m I_x \gamma_x} \cdot y_{l-m+1}^{(I_m^* - I_m)} \cdot \prod_{x=m+1}^{l} y_{l-x+1}^{I_x^*} \right)^r \tag{12}$$

$$= \underbrace{\left(g^{\delta + \sum_{x=0}^m I_x \gamma_x} \cdot \prod_{x=m+1}^{l} y_{l-x+1}^{I_x^*} \right)^r}_{=Z} \cdot y_{l-m+1}^{r(I_m^* - I_m)} \tag{13}$$

Note that the second term in Equation (2) equals 1 as $I_x = I_x^*$ for all $x < m$. $\mathcal{B}$ can compute the term Z in Equation (4) using the y_x's (for $1 \leq x \leq l+1$) from its input. Note that

$$y_{l-m+1}^{r(I_m^* - I_m)} = y_{l-m+1}^{\frac{\alpha^{m+1}}{I_m - I_m^*}(I_m^* - I_m)} \cdot y_{l-m+1}^{\tilde{r}(I_m^* - I_m)} = y_{l-m+1}^{\tilde{r}(I_m^* - I_m)} / y_{l+2}$$

The first component of the private key in Equation (1) is

$$g_2^\alpha (h_0^{I_0} \cdots h_m^{I_m} \cdot g_3)^r = (y_{l+1} y_1^\gamma) \cdot Z \cdot (y_{l-m+1}^{\tilde{r}(I_m^* - I_m)} / y_{l+1}) = y_1^\gamma \cdot Z \cdot (y_{l-m+1}^{\tilde{r}(I_m^* - I_m)})$$

which can be computed by $\mathcal{B}$ easily as the term y_{l+1} cancels out. The second component is

$$g^r = g^{\frac{\alpha^{m+1}}{I_m - I_m^*} + \tilde{r}} = y_{m+1}^{\frac{1}{I_m - I_m^*}} \cdot g^{\tilde{r}}$$

which can be computed by $\mathcal{B}$ with its knowledge of y_{m+1} as $m \leq l$. Similarly, $\mathcal{B}$ can also compute the remaining components $h_{m+1}^r, \ldots, h_l^r$ of the private key SK_{w^m}.

(f) $\mathcal{B}$ can now compute SK_w using SK_{w^m}.

(g) Similarly, all the right siblings of the nodes on the path from root to w can be computed.

(h) $\mathcal{B}$ returns these keys as the private key for the time period τ.

6. When $\mathcal{A}$ asks a $\mathsf{challenge}(z, M_0, M_1)$, if $z \neq \tau^*$ then $\mathcal{B}$ outputs a random bit as answer to the $(l+1)$-wBDHI* challenger and halts. Otherwise, $\mathcal{B}$ computes the challenge ciphertext

$$\mathsf{CT} = \left(M_b \cdot T \cdot e(y_1, h^\gamma), h, h^{\delta + \sum_{x=0}^{l} I_x^* \gamma_x} \right)$$

where h and T are from the input given to $\mathcal{B}$ by the decision w-BDHI* challenger.

(a) Let $h = g^c$ for some unknown $c \in \mathbb{Z}_p$. Note that if $T = e(g,h)^{(\alpha^{l+2})}$ then the challenger ciphertext is

$$\begin{aligned}
\mathsf{CT} &= \left(M_b \cdot T \cdot e(y_1, h^\gamma), h, h^{\delta + \sum_{x=0}^{l} I_x^* \gamma_x}\right) && (14)\\
&= \left(M_b \cdot e(g,h)^{(\alpha^{l+2})} \cdot e(y_1, h^\gamma), h, h^{\delta + \sum_{x=0}^{l} I_x^* \gamma_x}\right) && (15)\\
&= \left(M_b \cdot e(g,g)^{c\cdot(\alpha^{l+1})\cdot\alpha} \cdot e(y_1, g^\gamma)^c, g^c, \left(\prod_{x=0}^{l} (g^{\gamma_x}/y_{l-x+1})^{I_x^*} \cdot (g^\delta \prod_{x=0}^{l} y_{l-x+1}^{I_*})^c\right)\right) && (16)\\
&= \left(M_b \cdot e(g^\alpha, g^{(\alpha^{l+1})})^c \cdot e(y_1, g^\gamma)^c, g^c, (h_0^{I_0^*}, \cdots, h_l^{I_l^*})^c\right) && (17)\\
&= \left(M_b \cdot e(g_1, g^{(\alpha^{l+1})} g^\gamma)^c, g^c, (h_0^{I_0^*}, \cdots, h_v^{I_v^*}, \cdots, h_l^{I_l^*})^c\right) && (18)\\
&= \left(M_b \cdot e(g_1, g_2)^c, g^c, (h_0^{I_0^*}, \cdots, h_v^{I_v^*})^c\right) && (19)\\
& && (20)
\end{aligned}$$

which is a valid encryption under the given public key for the time period τ^*, associated with the node w^*.

(b) If T is from $\mathbb{G}_1{}^*$, the ciphertext CT is independent of b in $\mathcal{A}$'s view.

7. When $\mathcal{A}$ outputs its bit b', then $\mathcal{B}$ forwards this bit to the $(l+1)$-wBDHI* challenger as the solution.

Probability Analysis. As described above, $\mathcal{B}$ simulates a perfect execution environment for $\mathcal{A}$. It does not get any advantage only if its guess for the challenge time period τ^* is incorrect. In case the guess is correct it gets the same advantage as $\mathcal{A}$. Hence, the advantage of $\mathcal{B}$ is given as $\lambda \geq \frac{Adv_{\mathcal{A}^{CPA}}}{N}$. Rewriting the equation we have

$$Adv_{\mathcal{A}^{CPA}} \leq N \cdot \lambda$$

□

E Achieving fs-IND-CCA security.

Canetti et al. [20] used the simulation-sound NIZK proofs [42] to obtain chosen ciphertext security for forward secure encryption schemes in the standard model. However this approach results in very inefficient scheme and serves only as a proof of feasibility [20]. The same authors later proposed an efficient and generic technique to achieve chosen ciphertext security for public key encryption scheme using any chosen plaintext secure identity based encryption (IBE) scheme and strong one-time signatures [21]. They later extended this technique to achieve chosen ciphertext security for binary tree encryption (BTE) schemes [32]. The same technique can be directly used to achieve fs-IND-CCA security for forward secure encryption schemes. This is possible since as any BTE scheme can be turned into a forward secure encryption scheme [20]. Using this technique, an fs-IND-CPA secure scheme that uses a binary tree of depth l (supporting $2^{l+1} - 1$ time periods) can be turned in to an fs-IND-CCA secure scheme which uses another binary tree of depth $\frac{l-l_s-1}{2}$, where l_s is the length of the public key of the one-time signature used. In other words, the resulting fs-IND-CCA secure scheme supports $2^{\frac{l-l_s+1}{2}} - 1$ time periods.