# A Practical Forward-Secure Public-Key Encryption Scheme with Untrusted Update

Xiujie Zhang[1], Chunxiang Xu[2]
*(Corresponding author: Xiujie Zhang)*
School of Computer Engineering, WeiFang University[1]
No.5147, Dongfeng Street, Weifang 261061, China
(Email: 2008xiujie@163.com)
School of Computer Science and Engineering, University of Electronic Science and Technology of China[2]
No.2006, Xiyuan Avenue, West Hi-tech Zone, Chengdu 611731, China
*(Received Feb. 9, 2014; revised and accepted Nov. 15 & Dec. 26, 2014)*

## Abstract

Forward-secure public-key schemes with untrusted update are a variant of forward-secure public-key schemes such that the private key can be updated in the encrypted version. In this paper, we firstly describe forward-secure public-key scheme with untrusted update and security definition. We put forward a generic construction and prove its security in standard model. In our construction, the forward security is realized by applying a binary tree encryption and the update security is achieved by using a symmetric encryption. Meanwhile, we proved our scheme in the standard model. Our generic construction is very practical compare with previous works.

*Keywords: Composite order bilinear groups, forward security, key exposure, provable security, public key encryption, untrusted update*

## 1 Introduction

Traditionally, a forward secure cryptography assumes that one to mitigate the damage caused by exposure of secret keys. In a forward-secure public-key system, private keys are updated at regular time periods; an exposure of the private key $SK_i$ corresponding to a given time period $i$ does not enable an adversary to compromise the security of the scheme for any time period prior to $i$. For the case of signature, forward security guarantees that past signatures are protected even if the secret key of the current time period exposed. There are a number of known forward-secure signature schemes [1, 2, 10, 11, 12, 17], and forward-secure symmetric-key setting has also been studied [3]. For the case of encryption, forward security ensures that even after an adversary having the private key $SK_i$(for some $i$), the adversary can obviously compromise future usage of the scheme but messages encrypted for past periods remains secret. In Eurocrypt 2003, Canetti et al. [8] introduced forward security to Public-Key Encryption(fsPKE) scheme. Subsequently, the work of [4, 13, 20, 21] construct forward-secure (Hierarchical) IBE scheme based on the study of a number of Identity-Based Encryption(IBE) schemes [5, 19]. Recently, Nieto et al. [18] put forth a forward-secure Hierarchical Predicate Encryption(HPE) which ensures forward security for plaintexts and for attributes that are hidden in HPE ciphertexts.

However, Boyen et al. [7] showed that forward security is not applied to many software environments such as GNU-PG or S/MIME. A Forward-Secure Signature scheme with Untrusted Update(FSS-UU) was first proposed by Boyen et al. [7] in which the signing key is additionally shielded by a second factor and key update can be performed on an encrypted version of the signing key. And they left the open problem of adding untrusted update to other existing fsPKE scheme. Sequently, the work of [15] solved the open problem of [7] and presented a very efficient generic construction from any FSS scheme by expanding MMM construction [17]. Specializing for the case of encryption, which is the focus of this work, forward security with untrusted update guarantees that even if an adversary learns the encrypted private key $EncSK_i$(for some $i$), he can not decrypt correctly any ciphertext. The work of [16] extended the untrusted update model to fsPKE scheme. Libert el al. gave the resulting fsPKE with untrusted update(uufsPKE) scheme with update security and forward security in the chosen-plaintext setting. In their scheme, their method needs both a fsPKE and a traditional PKE which is not practical and leads to double-cost of size of ciphertext,public key and secret key. Therefore, how to construct more efficient generic uufsPKE scheme is worth researching.

This paper shows that forward-secure public-key encryption scheme with untrusted update is easier to derive from Symmetric Encryption and semantically secure Binary Tree Encryption. We firstly give the formal def-

inition of forward-secure public-key encryption scheme with untrusted update and security definition. We propose a generic transformation of forward-secure public-key encryption scheme with untrusted update which supports both forward security and update security. In our generic transformation, the forward security can be achieved by applying binary tree encryption scheme and the update security can be obtained using symmetric encryption scheme. Our scheme does not rely random oracle. Meanwhile, we present a concrete construction based on Binary Tree Encryption which we propose and prove its security in standard model. To the author's knowledge, this is the first provable efficient uufsPKE scheme which realizes constant ciphertext size and decreases the size of private key compared with Libert [16].

The rest of this paper is organized as follows. In Section 2, we provide some necessary preliminaries about binary tree encryption and symmetric encryption. In Section 3, we define uufsPKE scheme and its security notion formally. Our generic construction and its security proof will be proposed in Section 4. Finally, we conclude in Section 5.

# 2 Preliminaries

For a positive integer $T$, $[T]$ denotes $\{1, \cdots, T\}$. Let $N$ be a positive integer and $Z_N = \{1, \ldots, N-1\}$. We let $x$ be chosen uniformly from $Z_N$ denote by $x \xleftarrow{\$} Z_N$. By $negl(\lambda)$, we denote a negligible function of parameter $\lambda$. PPT stands for probabilistic polynomial time. Algorithm $A$ with input $x$ and random tape $r$ is denoted by $A(x;r)$.In this section, we give the definitions of primitives which our scheme is based on.

## 2.1 Composite Order Bilinear Groups

We will use composite order bilinear groups introduced in [6]. In the group generator algorithm $\mathcal{G}$ takes as input a security parameter $1^\lambda$ and outputs the description of a bilinear group $G$ of composite order $N = p_1p_2p_3$ where $p_1, p_2, p_3$ are three prime numbers of magnitude $\Theta(2^\lambda)$. We let $G_1, G_2, G_3$ denote these subgroups. We assume that the generator algorithm outputs the values of $p_1, p_2, p_3$ and $g_i$ is a generator of subgroup $G_i$.

For $i, j = \{1, 2, 3\}$, we also let $G_{ij}$ denote the subgroup of order $p_ip_j$. If $h_i \in G_i$ and $h_j \in G_j$ for $i \neq j$, we have that $e(h_i, h_j) = 1_{G_T}$.

## 2.2 Hardness Assumptions

We define the following three assumptions in composite order bilinear groups.

**Assumption 1.** *The challenger runs $\mathcal{G}(1^\lambda)$ and gives to the adversary $\mathcal{A}$ the tuple $D^1 = (N, G, G_T, e, g_1, g_3)$. Then the challenger flips a random coin $\nu \in_R \{0,1\}$ and picks random $(z, \xi) \xleftarrow{\$} Z_{p_1} \times Z_{p_2}$. He computes $T_0^1 = g_1^z$ and $T_1^1 = g_1^z g_2^\xi$, sends $T_\nu^1$ to $\mathcal{A}$. In the end, $\mathcal{A}$ outputs a bit $\nu'$, and succeeds if $\nu' = \nu$.*

**Assumption 2.** *The challenger runs $\mathcal{G}(1^\lambda)$ and picks random exponents $(z, \upsilon, \mu, \tau) \xleftarrow{\$} Z_{p_1} \times Z_{p_2} \times Z_{p_2} \times Z_{p_3}$. He gives to $\mathcal{A}$ the tuple $D^2 = (N, G, G_T, e, g_1, g_3, g_1^z g_2^\upsilon, g_2^\mu g_3^\tau)$. Then he flips a random coin $\nu \leftarrow \{0,1\}$ and picks random $(\gamma, \xi, \kappa) \xleftarrow{\$} Z_{p_1} \times Z_{p_2} \times Z_{p_3}$. He computes $T_0^2 = g_1^\gamma g_3^\kappa$ and $T_1^2 = g_1^\gamma g_2^\xi g_3^\kappa$, sends $T_\nu^2$ to $\mathcal{A}$. In the end, $\mathcal{A}$ outputs a bit $\nu'$, and succeeds if $\nu' = \nu$.*

**Assumption 3.** *The challenger runs $\mathcal{G}(1^\lambda)$ and picks random exponents $(\alpha, s, \xi, \mu, \varpi) \xleftarrow{\$} Z_{p_1} \times Z_{p_1} \times Z_{p_2} \times Z_{p_2} \times Z_{p_2}$. He gives to $\mathcal{A}$ the tuple $D^3 = (N, G, G_T, e, g_1, g_3, g_1^\alpha g_2^\xi, g_1^s g_2^\mu, g_2^\varpi)$. Then he flips a random coin $\nu \leftarrow \{0,1\}$ and picks a random $w \xleftarrow{\$} Z_{p_1}$. He computes $T_0^3 = e(g_1, g_1)^{\alpha s}$ and $T_1^3 = e(g_1, g_1)^{\alpha w}$, sends $T_\nu^3$ to $\mathcal{A}$. In the end, $\mathcal{A}$ outputs a bit $\nu'$, and succeeds if $\nu' = \nu$.*

**Assumption 4.** *The advantage of any PPT adversary $\mathcal{A}$ in Assumption $i$ where $i \in \{1,2,3\}$ is $Adv^i_{\mathcal{G},\mathcal{A}} = \frac{1}{2}(Pr[\mathcal{A}(D^i, T_0^i) = 0] - Pr[\mathcal{A}(D^i, T_1^i) = 0])$. We say that $\mathcal{G}$ satisfies Assumption $i$ for all PPT algorithms $\mathcal{A}$, $Adv^i_{\mathcal{G},\mathcal{A}} \leq negl(n)$.*

## 2.3 Binary Tree Encryption Scheme

Binary tree encryption (BTE) was introduced by Canetti et al. [8]. In a BTE scheme, each node $w$ has two children (labeled $w0$ and $w1$) while in a HIBE scheme, each node has arbitrarily-many children labeled with arbitrary strings. We review the relevant definitions of BTE scheme due to [8].

**Definition 1.** *A binary tree encryption scheme BTE is a 4-tuple of PPT algorithms (Gen,Der,Enc,Dec) such that:*

*$Gen(\lambda, \ell) \rightarrow (sk_\varepsilon, pk)$. The randomized algorithm inputs a security parameter $k$ and the maximum tree depth $\ell$. It outputs some system parameters $pk$ along with a master (root) secret key $sk_\varepsilon$. (We assume that $\lambda$ and $\ell$ are implicit in $pk$ and all secret keys.)*

*$Der(w, sk_w) \rightarrow (sk_{w0}, sk_{w1})$. The key derivation algorithm takes as input the name of a node $w \in \{0,1\}^{\leq \ell}$ and its associated secret key $sk_w$. It outputs secret keys $sk_{w0}, sk_{w1}$ for the two children of $w$.*

*$Enc(w, pk, M) \rightarrow CT$. It takes as input $pk$, the name of a node $w \in \{0,1\}^{\leq \ell}$, and a message $M$, and returns a ciphertext $CT$.*

*$Dec(w, sk_w, CT) \rightarrow M$. The deterministic algorithm takes as input $w \in \{0,1\}^{\leq \ell}$, its associated secret key $sk_w$, and a ciphertext $CT$. It returns a message $M$ or the distinguished symbol $\perp$.*

*We require that for all $(pk, sk_w)$ output by Gen, any $w \in \{0,1\}^{\leq \ell}$ and any correctly-generated secret key $sk_w$ for this node, any message $M$, and all CT output by $Enc_{pk}(w, M)$ we have $Dec_{sk_w}(w, Enc_{pk}(w, M)) = M$.*

**Definition 2.** *A binary tree encryption scheme BTE is secure against selective-node, chosen-plaintext attacks (SN-CPA) if for all polynomially-bounded functions $\ell(\cdot)$ the advantage of any PPT adversary A in the following game cpa-bte is negligible in the security parameter $k$.*

**Setup Phase.** *$\mathcal{A}(k, \ell(k))$ outputs a node label $w^* \in \{0,1\}^{\leq \ell(k)}$. Algorithm $Gen(1^\lambda, 1^\ell)$ outputs $(pk, sk_\varepsilon)$.*

**Phase 1.** *In addition, algorithm $Der()$ is run to generate the secret keys of all the nodes on the path from the root to $w^*$ (we denote this path by $P$). The adversary is given $pk$ and the secret keys $sk_w$ for all nodes $w$ of the following form:*

- *$w = w'b$,where $w'b$ is a prefix of $w^*$ and $b \in \{0,1\}$(w is a sibling of some node in P);*
- *$w = w^*0$ and $w = w^*1$, if $|w^*| < \ell$(w is a child of $w^*$). Note that it allows the adversary to compute $sk_{w'}$ for any node $w' \in \{0,1\}^{\leq \ell}$ that is not a prefix of $w^*$.*

**Challenge Phase.** *The adversary generates a request challenge $(M_0, M_1)$. A random bit $b$ is selected and the adversary is given $C^* = Enc(pk, w^*, M_b)$.*

**Guess Phase.** *Eventually, $\mathcal{A}$ outputs a guess $b' \xleftarrow{\$} \{0,1\}$.*

### 2.4 Symmetric Encryption

A symmetric encryption (SE) scheme is a tuple of PPT algorithms $\Pi = (Gen, Enc, Dec)$ such that:

1) The key-generation algorithm **Gen** is a randomized algorithm. Takes as input the security parameter $1^n$ and outputs a key $k$; we write this as $k \leftarrow Gen(1^n)$. Without loss of generality, we assume that any key $k$ output by $Gen(1^n)$ satisfies $n \leq |k|$.

2) The encryption algorithm **Enc** may be randomized. Takes as input a key $k$ and a plaintext message $m \in \{0,1\}^*$, and outputs a ciphertext $c$, write by $c \leftarrow Enc_k(m)$.

3) The decryption algorithm **Dec** is deterministic. Takes as input a key $k$, a ciphertext $c$ and outputs a message $m$. That is, $m := Dec_k(c)$.

It is required that for every n, every key k output by $Gen(1^n)$, and every $m \in \{0,1\}^*$, it holds that $Deck(Enck(m)) = m$.

We present the formal security definition against chosen-ciphertext attacks where the adversary has access to a decryption oracle and the encryption oracle. Consider the following game c/a for any private-key encryption scheme $\Pi = (Gen, Enc, Dec)$, adversary $\mathcal{A}$ and the security parameter $n$.

1) A random key $k$ is generated by running $Gen(1^n)$.

2) The adversary $\mathcal{A}$ is given input $1^n$ and oracle to $Enc_k(\cdot)$ and $Dec_k(\cdot)$. It outputs a pair of messages $m_0, m_1$ of the same length.

3) A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow Enc_k(m_b)$ is computed and given to $\mathcal{A}$. We call $c$ the challenge ciphertext.

4) The adversary $\mathcal{A}$ continues to have oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, but is not allowed to query the latter on the challenge ciphertext itself. Eventually, $\mathcal{A}$ outputs a bit $b' \leftarrow \{0,1\}$.

5) The adversary wins the game if output of the game is 1, that is, $b' = b$.

**Definition 3.** *The symmetric encryption scheme SE has indistinguishable encryptions under a chosen-ciphertext attack(IND-CCA) if for every PPT adversaries A has the advantage $Adv^{c/a}_{SE,A}(\cdot)$ is negligible.*

## 3 Our Model

Our model extends the forward security model of the work [8] to untrusted update setting. We focus here on public-key encryption schemes secure against breakin attack in the untrusted update environments.

### 3.1 Forward-Secure Public-Key Encryption Scheme with Untrusted Update

We give the formal definition of Forward-Secure Public-Key encryption scheme with untrusted update (uufsPKE).

**Definition 4.** *A uufsPKE scheme consists of four algorithms, each of which is described in the following.*

*$KeyGen_{uu}(\lambda, N)$. The key setup algorithm is a probabilistic algorithm that takes as input a security parameter $\lambda$ and time period $N$, outputs decryption key $DecK$, initial encrypted secret key $EncSK_0$ and public parameters $PK$.*

*$Update_{uu}(EncSK_{i-1}, i)$. The untrusted update algorithm that takes as input the encrypted secret key $EncSK_{i-1}$ for time period $i-1$, generates a new encrypted secret key $EncSK_i$. Then deletes the old key $EncSK_{i-1}$. Note that this algorithm does not require the decryption key.*

*$Encrypt_{uu}(PK, i, M)$. The encryption algorithm is a probabilistic algorithm that takes as input public parameters, the current time period $i$ and a message $M$, outputs the ciphertext $CT$ for time period $i$.*

*$Decrypt_{uu}(PK, i, EncSK_i, DecK, CT)$. The decryption algorithm is a deterministic algorithm that takes as input public parameters $PK$, the current time period*

*i, the current encrypted secret key $EncSK_i$, the decryption key DecK and a ciphertext CT, outputs the message M.*

**Decyption Consistency Requirements.** *For any message M, the public key PK, the decryption key DecK, the secret key $SK_i$ for time period i and the output of $Encrypt_{uu}(PK, i, M)$, $Decrypt(PK, SK_w, Encrypt_{uu}(PK, i, M)) = M$ always holds.*

## 3.2 Security Definitions for uufsPKE

Now we give the formal security definition for Forward-secure Public-Key encryption scheme with untrusted update in terms of two games.

### 3.2.1 Forward Security

Formally, for a uufsPKE scheme, its semantic security against an adaptive chosen ciphertext attack under an adaptive break-in attack can be defined via the following game $fsc$ between an attacker $\mathcal{A}$ and a challenger $\mathcal{B}$.

**Setup Phase.** The challenger $\mathcal{B}$ runs algorithm $\mathbf{KeyGen}_{uu}(\lambda, N)$ and gives $\mathcal{A}$ the resulting public parameters $PK$, keeping the secret key $(EncSK_0, DecK)$ to itself. Here, a handle counter $i$ is set to 0.

**Phase 1.** $\mathcal{A}$ adaptively issues the following three queries:

- $update(i)$ queries. $\mathcal{B}$ runs algorithm $\mathbf{Update}_{uu}$ and updates the handle counter to $i \leftarrow i + 1$.
- $breakin(i')$ queries. At any time $i'$, $\mathcal{B}$ firstly checks if $i' \leq N - 1$. If this is true, it responds with the corresponding private-key share $EncSK_{i'}$ for current time period $i'$.
- $decryption(j, CT)$ queries. At any time $j$, $\mathcal{A}$ picks a ciphertext $CT$ and sends to $\mathcal{B}$. The challenger makes a call to $\mathbf{Decrypt}_{uu}(PK, j, EncSK_j, DecK, CT)$ using the corresponding private-key and forwards the result to $\mathcal{A}$.

**Challenge Phase.** $\mathcal{A}$ submits two message $M_0, M_1$ of equal size. $\mathcal{B}$ flips a uniform coin $v \xleftarrow{\$} \{0, 1\}$ and encrypts $M_v$ under $i^*$ with a call to $\mathbf{Encrypt}_{uu}(i^*, M_v, PK)$, where $i^*$ is the index of the current time period. Then $\mathcal{B}$ sends the resulting ciphertext $CT^*$ to $\mathcal{A}$.

**Phase 2.** The adversary $\mathcal{A}$ continues to issue additional queries as in Phase 1 other than $decryption(i^*, CT)$ with $CT \neq CT^*$ or $decryption(i, CT^*)$ with $i \neq i^*$.

**Guess Phase.** Finally, $\mathcal{A}$ outputs a guess $v' \xleftarrow{\$} \{0, 1\}$.

We refer to the above game as an IND-fs-CCA2 game. We let $Adv^{fsc}_{\mathbf{\Pi},\mathcal{A}}$ denote the advantage of an attacker $\mathcal{A}$ in this game fsc.

**Definition 5.** *An uufsPKE scheme $\Pi$ is IND-fs-CCA2 secure if for every PPT adversary $\mathcal{A}$, we have $Adv^{fsc}_{\mathbf{\Pi},\mathcal{A}} \leq negl(\lambda)$ in the above IND-fs-CCA2 game.*

### 3.2.2 Update Security

Formally, we define update security for a uufsPKE scheme via the following game $uuc$ between $A$ and $B$.

**Setup Phase.** The challenger $\mathcal{B}$ runs algorithm $\mathbf{KeyGen}_{uu}(\lambda, N)$ and gives $(PK, EncSK_0)$ to $\mathcal{A}$, keeping the secret key $DecK$ for itself. Also, a handle counter $i$ is set to 0.

**Phase 1.** $\mathcal{A}$ adaptively issues the following two queries:

- $update(i)$ queries. $\mathcal{B}$ updates the handle counter to $i \leftarrow i + 1$.
- $decryption(j, CT)$ queries. At any time $j$, $\mathcal{A}$ picks a ciphertext $CT$ and sends to $\mathcal{B}$. The challenger runs algorithm $\mathbf{Decrypt}_{uu}$ using $(EncSK_0, DecK)$ and sends the result to $\mathcal{A}$.

**Challenge Phase.** Once $\mathcal{A}$ decides that Phase1 is over, it submits two message $M_0, M_1$ of equal size. $\mathcal{B}$ flips a uniform coin $c \xleftarrow{\$} \{0, 1\}$ and encrypts $M_c$ under $i^*$ with a call to $\mathbf{Encrypt}_{uu}(i^*, M_c, PK)$, where $i^*$ is the index of the current time period. Then $\mathcal{B}$ sends the resulting ciphertext $CT^*$ to $\mathcal{A}$.

**Phase 2.** $\mathcal{A}$ issues additional queries as in Phase 1 other than $decryption(i^*, CT)$ with $CT \neq CT^*$) or $decryption(i, CT^*)$ with $i \neq i^*$.

**Guess Phase.** Finally, $\mathcal{A}$ outputs a guess $c' \xleftarrow{\$} \{0, 1\}$.

We refer to the above adversary $\mathcal{A}$ as a IND-uu-CCA2 adversary. We let $Adv^{uuc}_{\mathbf{\Pi},\mathcal{A}}$ denote the advantage of an attacker $\mathcal{A}$ in this game $uuc$.

**Definition 6.** *An uufsPKE scheme $\Pi$ is IND-uu-CCA2 secure if for any PPT adversary $\mathcal{A}$, we have $Adv^{uuc}_{\mathbf{\Pi},\mathcal{A}} \leq negl(\lambda)$ in the above IND-uu-CCA2 game.*

# 4 Generic Construction from BTE and SE

In our construction, it allows for automated updates of encrypted keys and the user holding the second factor does not have to intervene in operations where the update algorithm is programmed to update the blinded version of the secret key at the beginning of each period. The second factor is only needed for decrypting messages as in many typical implementations of public key encryption. Beyond the forward security requirement, such a scheme prevents an adversary just in possession of the encrypted secret key from forging ciphertext for past, current, and future periods.

### 4.1 The General Transformation

In this section, we present the generic construction of uufsPKE from any binary tree encryption. We apply a symmetric encryption scheme to implement update security. And we can use any chosen ciphertext secure symmetric encryption. Formally, a BTE consists of PPT algorithms $\mathcal{E}_1 = (Gen, Der, Enc, Dec)$ and a SE scheme $\mathcal{E}_2 = (Gen', Enc', Dec')$. Our scheme can be described by four algorithms, denoted by $\mathbf{\Pi} = (\mathbf{KeyGen_{uu}}, \mathbf{Update_{uu}}, \mathbf{Encrypt_{uu}}, \mathbf{Decrypt_{uu}})$.

$KeyGen_{uu}(k, N)$**.** It runs algorithm $Gen(k, \ell) \rightarrow (sk_\varepsilon, pk)$ that takes as input a security parameter $k \in \mathbb{N}$ and $\ell$, the smallest integer satisfying $N \leq 2^\ell$. Firstly, sets $DecK \leftarrow Gen'(1^k)$ Then, symmetric encryption algorithm generates $esk_\varepsilon = Enc'_{DecK}(sk_\varepsilon)$ and uses $esk_\varepsilon$ generating the initial encrypted secret key of our generic construction by calling algorithm $Der(.,.)$. Denoted by $EncSK_0 = (esk_{0^\ell}, \{esk_1, esk_{01}, esk_{001}, ..., esk_{0^{\ell-1}1}\})$. Finally, it returns public key $(pk, N)$ and the secret key $(EncSK_0, DecK)$.

$Update_{uu}(EncSK_i, i+1)$**.** The encrypted secret key $EncSK_i$ be organized as a stack of node keys where the secret key $esk_{\langle i \rangle}$ on top. Here, $\langle i \rangle = i_1 i_2 ... i_\ell$ is the binary expression for the $i^{th}$ time period. Firstly, pops the top off the stack. If $i_\ell = 0$, search the only path from node $\langle i \rangle$ to root, denoted by $P_i$. And generate another set $R_i = (\{esk_{i_1 ... i_l}\}_{l \in \{1,...,\ell\} s.t. i_l = 0})$. Push the node secret key onto the stack from $R_i$ according the relation between the leaf $\langle i \rangle$ and the sibling of the node in $R_i$. The closer two nodes, then first-in stack. Otherwise, let $h \in \{1, ..., \ell\}$ denote the largest index such that $i_h = 0$. It have $w = i_1 i_2 ... i_{h-1} 1 \in \{0,1\}^h$ and recursively use $Der(w, esk_w)$ to generate node keys $esk_{w1}, esk_{w01}, ..., esk_{w0^{\ell-h-1}}, esk_{w0^{\ell-l}}$. Push all these node secret keys onto the stack by the reverse order. The new top of the stack is $esk_{w0^{\ell-l}}$ that is $\langle i+1 \rangle = w0^{\ell-l}$. In both cases, it erase the leaf node key $esk_{\langle i \rangle}$ and return the new stack.

$Encrypt_{uu}(i, pk, M)$**.** In period $i$, to encrypt a message $M \in G_T$, the sender parses $\langle i \rangle$ as $i_1 i_2 ... i_\ell$. Then, it computes $CT \leftarrow Enc(pk, \langle i \rangle, M)$.

$Decrypt_{uu}(i, DecK, EncSK_i, pk, CT)$**.** To decrypt ciphertext $CT$, it regenerates $sk_{\langle i \rangle} \leftarrow Dec'_{DecK}(esk_{\langle i \rangle})$. Then, it computes $M \leftarrow Dec(sk_{\langle i \rangle}, \langle\ i\ \rangle, CT)$.

### 4.2 Simulation Theorems

In this section, we give the proof on how to reduce the security of SN-CPA for BTE scheme and SE scheme against chosen ciphertext attack to forward security and update security for a uufsPKE scheme. We only consider the strongest security, that is, chosen ciphertext security. We will prove our general transformation's forward security and update security against an adaptive chosen ciphertext attack in the following two theorems.

**Theorem 1.** *Suppose $\mathcal{E}_1$ is a SN-CPA(resp.SN-CCA) secure BTE and $\mathcal{E}_2$ is a symmetric encryption scheme with chosen ciphertext security. Then the uufsPKE scheme $\mathbf{\Pi}$ described above is IND-fs-CPA(resp.IND-fs-CCA2) securtiy.*

*Proof.* Suppose there is an adversary $\mathcal{A}$ has non-negligible advantage in attacking forward security of the above scheme. That is, $Adv^{fsc}_{\mathcal{A},\mathbf{\Pi}} > \varepsilon$, $\varepsilon$ is a negligible parameter. We build an algorithm $\mathcal{B}$ that breaks BTE scheme $\mathcal{E}_1$ with advantage $Adv^{fsc}_{\mathcal{A},\mathbf{\Pi}}/N(k)$ where $N(k)$ is polynomial in the security parameter $k$. Algorithm $\mathcal{B}$ uses $\mathcal{A}$ to interact with a BTE challenger as follows:

**Initialization.** Firstly, $\mathcal{A}$ outputs the challenge time period $i^*$ and the corresponding to the node label $\langle i^* \rangle$ of the binary tree. Secondly, $\mathcal{B}$ runs $Gen'(k) \rightarrow DecK$. It outputs $(i^*, DecK)$.

**Setup.** The BTE challenger runs $\mathbf{Gen}(k, \ell) \rightarrow (sk_\varepsilon, PK)$ and gives to $\mathcal{B}$. And then $\mathcal{B}$ forwards $(PK, DecK)$ to $\mathcal{A}$.

**Phase 1.** $\mathcal{A}$ adaptively issues the following queries.

- $update(i)$ queries: $\mathcal{B}$ runs algorithm **Der(.,.)** and updates the handle counter to $i \leftarrow i+1$ at the same time.
- $breakin(i')$ queries: At any time $i'$, $\mathcal{B}$ firstly checks if $i' \leq N-1$. If this is true and $i' \leq i^*$, then $\mathcal{B}$ outputs a random bit and halts. Otherwise, if $i' > i^*$, $\mathcal{B}$ runs $Enc'_{DecK}(sk_\varepsilon)$ to obtain $esk_\varepsilon$ and forwards to the BTE challenger. Using $esk_\varepsilon$. The BTE challenger recursively apply algorithm **Der(.,.)** to obtain node keys and finally $EncSK_{\langle i' \rangle}$. Then it responds with the corresponding private-key $EncSK_{i'}$ for the current time period $i'$.
- $decryption(j, CT)$ queries: At any time $j$, $\mathcal{A}$ picks a ciphertext $CT$ and sends to $\mathcal{B}$. If $i^* \leq j$, $\mathcal{B}$ decrypt the ciphertext by himself. Otherwise,$\mathcal{B}$ makes a call to $\mathbf{Dec'}_{DecK}(esk_{\langle j \rangle})$ and forwards to the BTE challenger for $\mathbf{Dec}(PK, j, sk_{\langle j \rangle}, CT)$ using the corresponding private-key. Then returns the result to $\mathcal{A}$.

**Challenge Phase.** $\mathcal{A}$ outputs two equal length messages $M_0, M_1$. If $i \leq i^*$, $\mathcal{B}$ forwards $M_0, M_1$ to the BTE challenger. It runs $\mathbf{Enc}(PK, \langle i^* \rangle, M_b)$ to obtain $CT^*$ for a random $b \in \{0,1\}$ and gives $\mathcal{A}$ the challenge ciphertext $CT^*$. Otherwise, $\mathcal{B}$ outputs a random bit and halts.

**Phase 2.** $\mathcal{A}$ continues to issue queries as **Phase 1** where decryption queries for $(j, CT)$ with $j \neq i^*$.

**Guess Phase.** Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for $b$. $\mathcal{B}$ forwards $b'$ to the BTE challenger and wins the game if $b = b'$.

This completes the description of algorithm $\mathcal{B}$. Let $Adv^{cpa-bte}_{\mathcal{B},\mathbf{\Pi}}$ be $\mathcal{B}$'s advantage in winning the BTE game cpa-bte. Let $Adv^{fsc}_{\mathcal{A},\mathbf{\Pi}}$ be $\mathcal{A}$'s advantage in winning the fsc game of uufsPKE scheme. It is straightforward to see that when $i^* = i$ the copy of $\mathcal{A}$ running within $\mathcal{B}$ has exactly the same view as in a real fsc game. Since $\mathcal{B}$ guesses $i^* = i$ with probability $1/N$, we have that $\mathcal{A}$ correctly predicts the bit $b$ with advantage $Adv^{fsc}_{\mathcal{A},\mathbf{\Pi}}/N(k)$. □

**Theorem 2.** *Suppose $\mathcal{E}_1$ is a SN-CPA(resp. SN-CCA) secure BTE and $\mathcal{E}_2$ is a symmetric encryption scheme against chosen ciphertext attack. Then the uufsPKE scheme $\mathbf{\Pi}$ above is IND-uu-CPA(resp.IND-uu-CCA2) security.*

*Proof.* Suppose $\mathcal{A}$ wins uuc game with non-negligible advantage in the above scheme. That is, $Adv^{uuc}_{\mathcal{A},\mathbf{\Pi}} > \varepsilon$ where $\varepsilon$ is a negligible parameter. Then we show how to construct an algorithm $\mathcal{B}$ that breaks SE scheme $\mathcal{E}_2$. Algorithm $\mathcal{B}$ starts by breaking the SE scheme $\mathcal{E}_2$. Using $\mathcal{A}$, $\mathcal{B}$ interacts with a SE challenger as follows:

**Setup.** To launch the game, $\mathcal{B}$ runs $\mathbf{Gen}(k, \ell) \rightarrow (sk_\varepsilon, PK)$. It sends $sk_\varepsilon$ to the SE challenger and obtains the initial encrypted node key $esk_\varepsilon$. Using $esk_\varepsilon$, $\mathcal{B}$ recursively runs algorithm **Der(.,.)** and generates all the encrypted keys $\{EncSK_0, EncSK_1, ..., EncSK_N\}$. And then, $\mathcal{B}$ sends the set of all the encrypted keys to $\mathcal{A}$.

**Phase 1.** $\mathcal{A}$ adaptively issues the following two queries:

- *update*$(i)$ queries. $\mathcal{B}$ updates the handle counter to $i \leftarrow i + 1$.
- *decryption*$(j, CT)$ queries. At any time $j$, $\mathcal{A}$ picks a ciphertext $CT$ and sends to $\mathcal{B}$. $\mathcal{B}$ request the secret key of time period $j$ for the SE challenger. It runs algorithm **Dec'**$(EncSK_j)$ and outputs the secret key $sk_{\langle j \rangle}$. $\mathcal{B}$ computes $\mathbf{Dec}(sk_{\langle j \rangle}, CT) = M$ and sends the result to $\mathcal{A}$.

**Challenge Phase.** $\mathcal{A}$ chooses two equal length messages $M_0, M_1$ and sends to $\mathcal{B}$. It runs $\mathbf{Enc}(PK, \langle i^* \rangle, M_b)$ to obtain $CT^*$ for a random $b \in \{0, 1\}$ and gives $\mathcal{A}$ the challenge ciphertext $CT^*$.
**Phase 2**. $\mathcal{A}$ continues to issue queries as **Phase 1** where decryption queries for $(j, CT)$ with $j \neq i^*$.

**Guess Phase.** Finally, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for $b$. $\mathcal{B}$ forwards $b'$ to the SE challenger and wins the game if $b = b'$.

□

# 5 A Concrete Forward-Secure Public-Key Encryption Scheme with Untrusted Update in Standard Model

In this section, we give a concrete construction of BTE scheme and SE scheme, respectively. Combined with both schemes, we present a uufsPKE scheme. We prove its security in standard model. Finally, we compare our scheme with the previous.

## 5.1 Boilding Blocks: BTE and SE

Firstly, we propose a new binary tree encryption scheme $\mathcal{E}_1$ with SN-CPA based on dual system encryption and prove its security. For simplicity, we do not consider chosen-ciphertext security of binary tree encryption which can be added by simply using CHK transformation [9]. In the description below, we imagine binary tree of height $\ell$ where the root (at depth 0) has label $\varepsilon$.When a node at depth $\leq \ell$ has label $w$, its children are labeled with $w0$ and $w1$.Besides,$\langle i \rangle$ stands for the $\ell$-bit representation of integer $i$. The leaves of the reed correspond to successive time periods, stage $i$ being associated with the leaf labeled by $\langle i \rangle$.The BTE scheme denote by $\mathcal{E}_\mathbf{1} = (\mathbf{Gen}, \mathbf{Der}, \mathbf{Enc}, \mathbf{Dec})$ that works as follows.

*Gen*$(k, \ell)$ is an algorithm that, given a security parameter $k$ and the maximum tree depth $\ell$.

1) Choose bilinear map groups $(G, G_T)$ of order $N = p_1 p_2 p_3$ where $p_i$ is the order of the subgroup $G_i$ in $G$, with $p_i > 2^k$ for each $i \in \{1, 2, 3\}$.
2) Let $w = w_1 ... w_l$ be an $l$-bits string representing a node of binary tree and $w_i \in \{0, 1\}$ for all $i \in \{1, ..., l\}$. Define a function $H : \{0, 1\}^{\leq \ell} \rightarrow G_1$ as $H(w) = h_0 \prod_{j=1}^{l} h_j^{w_j}$ .
3) Outputs the public key $pk = (g_1, g_3, V = e(g_1, g_1)^\alpha, h_0, h_1, ..., h_\ell, H)$ and a root secret key $sk_\varepsilon = \alpha$ for independent and uniformly random $\alpha \in Z_N, (h_0, h_1, ..., h_\ell) \in G_1^{\ell+1}, g_1 \in G_1$ and $g_3$ is a generator of $G_3$.

*Der*$(pk, w, sk_w)$ is an algorithm that deduce the secret keys $sk_{w0}, sk_{w1}$ where $w0, w1$ are two children of $w$. Execute the following steps.

1) Let $w = w_1 ... w_l$. Parse $sk_w = (k_0, k_1, s_l, ..., s_\ell) = (g_1^\alpha \cdot H(w_1 ... w_l)^r \cdot g_3^u, g_1^r \cdot g_3^{v_0}, h_l^r \cdot g_3^{v_1}, ..., h_\ell^r \cdot g_3^{v_\ell})$, $u, v_0, v_1, ..., v_\ell \in Z_{p_3}, r \in Z_N$.
2) For $j \in \{0, 1\}$, output $sk_{wj} = (k_0 \cdot s_l^j \cdot H(w_1 ... w_l j)^{r'_j} \cdot g_3^{u'}, k_1 \cdot g_1^{r'_j} \cdot g_3^{v'_0}, s_{l+1} \cdot h_{l+1}^{r'_j} \cdot g_3^{v'_1}, ..., s_\ell \cdot h_\ell^{r'_j} \cdot g_3^{v'_\ell}) = (g_1^\alpha \cdot H(w_1 ... w_l j)^{r_j} \cdot g_3^{\widetilde{u}}, g_1^{r_j} \cdot g_3^{\widetilde{v_0}}, h_l^{r_j} \cdot g_3^{\widetilde{v_1}}, ..., h_\ell^{r_j} \cdot g_3^{\widetilde{v_\ell}})$ where $u', v'_0, v'_1, ..., v'_\ell \in$

$Z_{p_3}, r'_j \in Z_N$ and $r_j = r'_j + r, \widetilde{u} = u + u', \widetilde{v_0} = v_0 + v'_0, ..., \widetilde{v_\ell} = v_\ell + v'_\ell$.

$Enc(\langle i \rangle, pk, M)$ does the following:

1) Let $\langle i \rangle = i_1...i_\ell \in \{0,1\}^\ell$, select random $s \in Z_N$.
2) Compute and output the ciphertext $CT = (i, M \cdot V^s, g_1^s, H(i_1...i_\ell)^s)$.

$Dec(w, sk_w, CT)$ does the following:

1) et $\langle i \rangle = i_1...i_\ell \in \{0,1\}^\ell$, parse $sk_{\langle i \rangle}$ as $(k_0, k_1)$ and parse $CT$ as $(i, C_0, C_1, C_2)$.
2) Output the message $M = \dfrac{C_0 \cdot e(k_1, C_2)}{e(k_0, C_1)}$.

**Correctness.** Assuming the ciphertext is well-formed, we have

$$\begin{aligned} \frac{C_0 \cdot e(k_1, C_2)}{e(k_0, C_1)} &= \frac{M \cdot V^s \cdot e(g_1^r \cdot g_3^{v_0}, H(i_1...i_\ell)^s)}{e(g_1^\alpha \cdot H(w_1...w_l)^r \cdot g_3^u, g_1^s)} \\ &= M. \end{aligned}$$

Secondly, we give a construction of SE scheme $\boldsymbol{\mathcal{E}_2} = (\mathbf{Gen'}, \mathbf{Enc'}, \mathbf{Dec'})$ applying to build uufsPKE scheme. The encryption algorithm $Enc'_{sk_{ss}}(m) = (c_1, c_2, ..., c_\kappa, m \cdot \prod_{j \in [\kappa]} c_j^{\mu_j})$ for independent and uniformly random $c_j \in G$. The decryption algorithm $Dec'_{sk_{ss}}(c_1, c_2, ..., c_\kappa, c_0)$ firstly parses the ciphertext $c$ as $(c_1, c_2, ..., c_\kappa, c_0)$. And compute the message

$$m = \frac{c_0}{\prod_{j \in [\kappa]} c_j^{\mu_j}}.$$

## 5.2 A Concrete uufsPKE Scheme

Our concrete uufsPKE scheme consists of the following algorithms where periods are indexed from 0 to $T$-1 with $T = 2^\ell$.

$KeyGen_{uu}(k, N)$.

1) Run $Gen(k, \ell) \rightarrow (sk_\varepsilon, pk)$ where $pk = (g_1, g_3, V = e(g_1, g_1)^\alpha, h_0, h_1, ..., h_\ell, H)$ and $sk_\varepsilon = \alpha$.
2) Generate $G(r) \rightarrow DecK$, denoted by $DecK = (d_1, d_2, ..., d_t)$ where $d_i \in Z_N$.
3) Compute the initial encrypted root key $esk_\varepsilon = Enc'_{DecK}(sk_\varepsilon) = (c_1, c_2, ..., c_t, sk_\varepsilon \prod_{i \in [t]} c_i^{d_i})$.
4) The initial encrypted secret key $EncSK_0 = (esk_{0^\ell}, \{esk_1, esk_{01}, esk_{001}, ..., esk_{0^{\ell-1}1}\})$ using $esk_\varepsilon$ recursively apply algorithm $Der$.

$Update_{uu}(EncSK_i, i+1)$.

1) Parse $\langle i \rangle = i_1...i_\ell \in \{0,1\}^\ell$ and $EncSK_i = (esk_{\langle i \rangle}, \{esk_{i_1...i_l}\}_{l \in \{1,...,\ell\} s.t. i_l = 0})$. And delete $esk_{\langle i \rangle}$.
2) If $i_\ell = 0$, $EncSK_{i+1} = (esk_{i_1 \cdot i_{\ell-1}}, \{esk_{i_1 \cdot i_{l-1}}\}_{l \in \{1,...,\ell-1\} s.t. i_l = 0})$, that is, $EncSK_{i+1}$ includes the remaining node keys. Otherwise, let $l' \in \{0,1\}^\ell$ be the largest index such that $i_{l'} = 0$. Let $w' = i_1...i_{l'-1}1 \in \{0,1\}^{l'}$. Recursively run $Der$ for the node key $esk_{w'}$ to generate node keys $esk_{w'1}$, $esk_{w'01}$, $\cdots$, $esk_{w'0^{\ell-l'-1}1}$ and $esk_{w'0^{\ell-l'-1}1} = esk_{\langle i+1 \rangle}$. Delete $esk_{w'}$ and return $EncSK_{i+1} = (esk_{\langle i+1 \rangle}, \{esk_{i_1...i_{l-1}1}\}_{l \in \{1,...,l'-1\} s.t. i_l = 0}, \{esk_{w'1}, esk_{w'01}, \cdots, esk_{w'0^{\ell-l'-1}1}\})$.

$Encrypt_{uu}(i, pk, M)$. For $i \in [1, N]$, to encrypt the message $M$, does the following.

1) Parse $\langle i \rangle$ as $i_1...i_\ell \in \{0,1\}^\ell$.
2) Run $Enc(\langle i \rangle, pk, M)$ and return the ciphertext $CT = (i, M \cdot e(g_1, g_1)^{\alpha s}, g_1^s, H(i_1...i_\ell)^s)$.

$Decrypt_{uu}(i, EncSK_i, pk, CT)$. Given a ciphertext $CT = (i, C_0, C_1, C_2)$ and a encrypted secret key $EncSK_i = (esk_{\langle i \rangle}, \{esk_{i_1...i_l}\}_{l \in \{1,...,\ell\} s.t. i_l = 0})$ for the current period $i$.

1) Parse $esk_{\langle i \rangle}$ as $(esk^0_{\langle i \rangle}, esk^1_{\langle i \rangle})$.
2) Compute $sk_{\langle i \rangle} = (\dfrac{esk^0_{\langle i \rangle}}{\prod_{i \in [t]} c_i^{d_i}}, \dfrac{esk^1_{\langle i \rangle}}{\prod_{i \in [t]} c_i^{d_i}})$ by applying decryption algorithm $Dec'$.
3) Run the algorithm $Dec(sk_{\langle i \rangle}, \langle i \rangle, CT)$ to obtain the message $M$.

## 5.3 Security Proof

In the following, we devote to prove SN-CPA for the above BTE scheme $\mathcal{E}_1$ under three static assumptions in the standard model. Our security proof will use semi-functional ciphertext and semi-functional node keys which defined as follows. All the ciphertexts and node keys defined by the following are normal, where by normal we mean that they have no $G_2$ parts. On the other hand, a semi-functional key or ciphertext has $G_2$ parts.

- semi-functional ciphertexts is generated from a normal ciphertext $CT = (C'_0, C'_1, C'_2)$ and some $g_2 \in G_{p_2}$, by choosing random $x, z_c \xleftarrow{\$} Z_N$ and setting $CT^{semi} = (C'_0, C'_1 \cdot g_2^{x \cdot z_c}, C'_2 \cdot g_2^x)$.
- semi-functional node key are generated from a normal node key $sk'_w = (k'_0, k'_1, s'_l, ..., s'_\ell)$ by choosing random $y, z_w \xleftarrow{\$} Z_N$, and setting $sk_w^{semi1} = (k'_0 \cdot g_2^{y \cdot z_w}, k'_1 \cdot g_2^y, s'_l \cdot g_2^y, ..., s'_\ell \cdot g_2^y)$.

We now prove the semantic security of this BTE against selective-node chosen plaintext attacks(SN-CPA). In order to prove security we need a hybrid argument using a sequence of games. For each $i$, we denote by $S_i$ the probability that the challenger returns 1 at the end of $Game_i$. We also define $Adv_i = |Pr[S_i - 1/2]|$ for each $i$.

- *Game*: It is the real security game as described in Section 2.3.
- *Game′*: It is exactly the same as *Game* except that in the challenge phase, the ciphertext responds with the semi-functional ciphertext instead of normal ciphertext. Besides, at the begging of the game, the challenger chooses an index $i^* \xleftarrow{R} \{0,1,...,T-1\}$. At the challenge phase, the challenger halts and outputs a random bit if the challenge ciphertext is encrypted for a period $i$ such that $i \neq i^*$. Because the choice of $i^*$ is independent of $\mathcal{A}$'s view, we have $Pr[S_1] \leq Pr[S_0]/T$. Note that we denote $Path^*$ the path from the root to the leaf associated with $i^*$.
- $Game_k$ In this game, for $k \in [1, T-1]$, the ciphertext given to the attacker is semi-functional. For the first $k$ keys, if the corresponding node $w^k \in Path^*$, then returns semi-functional node key. Otherwise, the rest of the keys are normal.
- $Game_T$ is the same as $Game_{T-1}$ but the semi-functional challenge ciphertext is replaced by a a semi-functional encryption of a random message instead of $M_b$.

**Theorem 3.** *If Assumption1,2 and 3 hold, then our BTE scheme $\mathcal{E}_1$ is a SN-CPA.*

*Proof.* The proof proceeds using a sequence of games including steps similar to [14]. In order to prove this theorem we need the following lemmas:

**Lemma 1.** *Suppose there exists an algorithm $\mathcal{A}$ such that $Adv^{Game}_{\mathcal{A},\mathcal{E}_1} - Adv^{Game'}_{\mathcal{A},\mathcal{E}_1} \leq \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\epsilon$ in breaking Assumption 1.*

*Proof.* Suppose a PPT challenger $\mathcal{B}$ that breaks Assumption 1 with the help of a PPT adversary $\mathcal{A}$. $\mathcal{B}$ simulates *Game* or *Game′*. Initially, $\mathcal{B}$ receives input from the assumption's challenger, i.e. $D^1 = (N, G, G_T, e, g_1, g_3)$ and a challenge term $T$ which is equal to $g_1^z$ and $g_1^z g_2^\xi$. Algorithm $\mathcal{B}$ works as follows:

**Setup Phase.** $\mathcal{B}$ picks $\alpha, x_0, x_1, ..., x_\ell \xleftarrow{\$} Z_N$ and computes $V = e(g_1, g_1)^\alpha, h_0 = g_1^{x_0}, h_1 = g_1^{x_1}, ...,$ and $h_\ell = g_1^{x_\ell}$. It gives the public parameters $pk = (g_1, g_3, V = e(g_1, g_1)^\alpha, h_0, h_1, ..., h_\ell, H)$ to $\mathcal{A}$ where $(N, e, g_1, g_3)$ are given by the challenger $\mathcal{B}$. Initially, $\mathcal{A}$ outputs the target node $w^* = w_1^* w_2^* ... w_l^*$ with $l \leq \ell$. Next, $\mathcal{B}$ constructs node keys as follows. For a node label $w = w_1 w_2 ... w_l (l < \ell)$, $\mathcal{B}$ chooses $u, v_0, v_1, ..., v_\ell \in Z_{p_3}, r \xleftarrow{\$} Z_N$ and sets $sk_w = (g_1^\alpha \cdot \prod_{j=1}^{l} (g_1^{x_j \cdot w_j})^r \cdot g_3^u, g_1^r \cdot g_3^{v_0}, h_l^r \cdot g_3^{v_1}, ..., h_\ell^r \cdot g_3^{v_\ell})$.

**Phase 1.** $\mathcal{B}$ provides $\mathcal{A}$ with all secret keys for sibling of the nodes on the path from the root to $w^*$, as well as for the children of $w^*$.

**Challenge Phase** : Once $\mathcal{A}$ outputs two equal-length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. $\mathcal{B}$ flips a random coin $c \in \{0,1\}$, and generates the challenge ciphertext to be $CT^* = (M_c \cdot e(T, g_1)^\alpha, T, \prod_{j=1}^{l} T^{x_j w_j})$, which is sent to $\mathcal{A}$.

**Analysis.** If $T = g_1^z$, then this is a normal ciphertext which has no $G_2$ component. If $T = g_1^z g_2^\xi$, this is a semi-functional ciphertext with $z_c = \Sigma_{j=1}^{l} x_j w_j$. If $\mathcal{A}$ succeeds in distinguishing these two games then our challenger $\mathcal{B}$ can use $\mathcal{A}$ to break Assumption 1. Thus if Assumption 1 is holds, these two games are indistinguishable.

□

It is straightforward that from *Game′* to $Game_1$ is just a conceptual change since the adversary $\mathcal{A}$'s view is the same in both games. Thus we have $Adv_1 = Adv'/T$.

**Lemma 2.** *Suppose there exists an algorithm $\mathcal{A}$ such that $Adv^{Game_{k-1}}_{\mathcal{A},\mathcal{E}_1} - Adv^{Game_k}_{\mathcal{A},\mathcal{E}_1} \leq \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\epsilon$ in breaking Assumption 2.*

*Proof.* $\mathcal{B}$ first receives $D^2 = (N, G, G_T, e, g_1, g_3, g_1^z g_2^v, g_2^\mu g_3^\tau)$ and $T$ where $T = g_1^\gamma g_3^\kappa$ or $T = g_1^\gamma g_2^\xi g_3^\kappa$. $\mathcal{B}$ picks a random exponents $a, b, \alpha \in Z_N$ and sets the public parameters as Lemma 1.

When $\mathcal{A}$ requests the $i^{th}$ key for the period $i$ corresponding to node $\langle i \rangle$ in the tree. If $i < k$ and $\langle i \rangle \in Path^*$, $\mathcal{B}$ creates a semi-functional node key of $Type1$. It does this by choosing random exponents $y, z_w, r \in Z_N$ and setting $sk_w^{semi1} = (g_1^\alpha \cdot \prod_{j=1}^{l} (g_1^{x_j \cdot w_j})^r \cdot (g_2^\mu g_3^\tau)^{y \cdot z_w}, g_1^r \cdot (g_2^\mu g_3^\tau)^y, h_l^r \cdot (g_2^\mu g_3^\tau)^y, ..., h_\ell^r \cdot (g_2^\mu g_3^\tau)^y)$. This is a properly distributed semi-functional node key of $Type1$ with $G_2$ component $g_2^y$. For $i > k$, $\mathcal{B}$ generates normal keys by using random exponents $r, y \in Z_N$ and setting $sk_w = (g_1^\alpha \cdot \prod_{j=1}^{l} (g_1^{x_j \cdot w_j})^r \cdot g_3^y, g_1^r \cdot g_3^y, h_l^r \cdot g_3^y, ..., h_\ell^r \cdot g_3^y)$. To create the $k^{th}$ requested key, $\mathcal{B}$ sets $z_k = \sum_{j=1}^{l} x_j \cdot w_j$, chooses a random exponent $y_w \in Z_N$, and sets $sk_w^* = (g_1^\alpha \cdot T^{z_k} \cdot g_3^{y_w}, T, h_l^r \cdot T^{y_w}, ..., h_\ell^r \cdot T^{y_w})$.

**Challenge Phase.** $\mathcal{A}$ sends $\mathcal{B}$ two equal-length plaintexts $M_0, M_1 \in \mathcal{M}$. $\mathcal{B}$ chooses a random coin $v \in \{0,1\}$, and generates the ciphertext $CT = (M_c \cdot e(g_1^z g_2^v, g_1)^\alpha, g_1^z g_2^v, \prod_{j=1}^{l} (g_1^z g_2^v)^{x_j w_j})$, which is sent to $\mathcal{A}$.

**Analysis.** We note that this sets $z_c = \Sigma_{j=1}^{l} x_j w_j$. If $T = g_1^\gamma g_3^\kappa$, then this is a normal ciphertext which has no $G_2$ component. If $T = g_1^\gamma g_2^\xi g_3^\kappa$, this is a semi-functional ciphertext with $z_c = \Sigma_{j=1}^{l} x_j w_j$. If $\mathcal{A}$ succeeds in distinguishing these two games then our challenger $\mathcal{B}$ can use $\mathcal{A}$ to break Assumption 2. Thus if Assumption 2 is holds, these two games are indistinguishable.

□

**Lemma 3.** *Suppose there exists an algorithm $\mathcal{A}$ such that $Adv^{Game_{T-1}}_{\mathcal{A},\mathcal{E}_1} - Adv^{Game_T}_{\mathcal{A},\mathcal{E}_1} \leq \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\epsilon$ in breaking Assumption 3.*

Table 1: Comparison of performance with existing schemes

| S(Scheme) | Encryption/Decryption | Hard Problems | Key update time | Ciphertext length | Public-key/Secret-key size |
|---|---|---|---|---|---|
| S of [16] | $\mathcal{O}(N \cdot (loglogN)^2)$ | Three static assumptions | $\mathcal{O}(logN)$ | $\mathcal{O}(logN)$ | $\mathcal{O}(logN)$ |
| Our Scheme | $\mathcal{O}(1)$ | $BDDH$ | $\mathcal{O}(logN)$ | $\mathcal{O}(1)$ | $\mathcal{O}(logN)$ |

*Proof.* $\mathcal{B}$ first receives $(N, G, G_T, e, g_1, g_3, g_1^{\alpha} g_2^{\xi}, g_1^{s} g_2^{\mu}, g_2^{\varpi})$ and $T$, where $T = e(g_1, g_1)^{\alpha s}$ or $T = e(g_1, g_1)^{\alpha w}$. $\mathcal{B}$ chooses random exponents $x_0, x_1, ..., x_\ell \overset{\$}{\leftarrow} Z_N$ and sets the public parameters as $V = e(g_1, g_1^{\alpha} g_2^{\xi}), h_0 = g_1^{x_0}, h_1 = g_1^{x_1}, ...,$ and $h_\ell = g_1^{x_\ell}$. It sends these to $\mathcal{A}$. When $\mathcal{A}$ requests a key for time period $i$, $\mathcal{B}$ generates a semi-functional. It does this by setting $sk_w = (g_1^{\alpha} g_2^{\xi} \cdot \prod_{j=1}^{l} (g_1^{x_j \cdot w_j})^r \cdot g_3^u, g_1^r \cdot g_3^{v_0}, h_l^r \cdot g_3^{v_1}, ..., h_\ell^r \cdot g_3^{v_\ell})$. After providing the appropriate secret keys, $\mathcal{B}$ responds to the challenge query from $\mathcal{A}$. Specifically, $\mathcal{B}$ chooses a random bit $b$ and returns $CT = (M_b \cdot T, g_1^s g_2^{\mu}, (g_1^s g_2^{\mu})^{\Sigma_{j=1}^{l} x_j w_j})$. If $T = e(g_1, g_1)^{\alpha s}$, then this is a properly distributed semi-functional ciphertext with message $M_b$. On the other hand, if $T \overset{\$}{\leftarrow} G_T$, then this is a semi-functional ciphertext with a random message. Therefore, the value of $b$ is information-theoretically hidden and the probability of success of any algorithm $\mathcal{A}$ in this game is exactly $1/2$, since $b \overset{\$}{\leftarrow} \{0, 1\}$. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 3 with non-negligible advantage. □

This conclude the proof of Theorem 3. □

According to Theorem 1 and Theorem 2, we conclude our concrete scheme has forward security and update security.

### 5.4 Comparison with the Existing Schemes

Now we compare the efficiency of our method with the prevail classic uufsPKE [16] in Table 1. Similarly, we associate time periods $N$ with the leaves only, we have the same efficiency of key generation phase. From Table 1, our scheme maintains the efficiency of key update and the size of public-key, secret-key. Using the techniques of Lewko et al. [14], the efficiency of our encryption/decrytion scheme and size of the ciphertext from $\mathcal{O}(logN)$ to $\mathcal{O}(1)$. Therefore, considering the security and the performance efficiency, our scheme is much better than previous schemes.

## 6 Conclusions

In this paper, motivated by the work of Libert [16], we give formal definition for uufsPKE scheme, as well as a general framework for constructing uufsPKE from BTE and SE scheme. We have proved our scheme is IND-uufs-CCA secure in standard model. Furthermore, we give a concrete construction of BTE scheme and prove the SN-CPA security under three static assumptions. Finally, we presented the first completely uufsPKE scheme that is forward security and update security against chosen ciphertext attack without random oracle. Compared with existing scheme, our scheme performs more faster.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (NO. 61370203), the Key Technology Research and Development Program of Sichuan Province and Chengdu Municipality (NO. szjj2015-054), the doctoral research fund of Weifang university (No.2015BS11).

## References

[1] M. Abdalla, L. Reyzin, "A new forward-secure digital signature scheme", in *Advances in Cryptology (ASIACRYPT'00)*, pp. 116–129, Kyoto, Japan, 2000.

[2] M. Bellare, S. K. Miner, "A forward-secure digital signature scheme", in *Advances in Cryptology (CRYPTO'99)*, pp. 431–448, Santa Barbara, California, USA, 1999.

[3] M. Bellare, B. Yee, "Forward security in private-key cryptography", in *The Cryptographer's Track at RSA Conference (CT-RSA 2003)*, pp. 1–18, San Francisco, CA, USA, 2003.

[4] D. Boneh, X. Boyen, and E. J. Goh, "Hierarchical identity based encryption with constant size ciphertext", in *Advances in Cryptology (EUROCRYPT'05)*, pp. 440–456, Aarhus, Denmark, 2005.

[5] D. Boneh, M. Franklin, "Identity-based encryption from the weil paring", in *Advances in Cryptology (CRYPTO'01)*, pp. 213–229, Santa Barbara, California, USA, 2001.

[6] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts", in *The second Theory of Cryptography Confference (TCC 2005)*, pp. 325–341, Cambridge, MA, USA, 2005.

[7] X. Boyen, H. Shacham, and E. Shen, et. al., "Forward-secure signatures with untrusted update", in *The 13th ACM Conference on Computer and Communications Security (CCS'06)*, pp. 191–200, Alexandria, VA, USA, 2006.

[8] R. Canetti, S. Halevi, J. Katz, "A forward-secure public-key encryption scheme", in *Advances in Cryptology (EUROCRYPT'03)*, pp. 255–271, Warsaw, Poland, 2003.

[9] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from Identity-based encryption", in *Advances in Cryptology (EUROCRYPT'04)*, pp. 207–222, Interlaken, Switzerland, 2004.
[10] G. Itkis, L. Reyzin, "Forward-secure signatures with optimal signing and verifying", in *Advances in Cryptology (CRYPTO'01)*, pp. 499–514, Santa Barbara, California, USA, 2001.
[11] A. Kozlov, L. Reyzin, "Forward-secure signatures with fast key update", in *Security in Communication Networks*, pp. 241-256, Amalfi, Italy, 2003.
[12] H. Krawczyk, "Simple forward-secure signatures from any signature scheme", in *The 10th ACM Confference on Computer and Communications Security*, pp. 108–115, New York, 2000.
[13] S. Kunwar, C. Pandurangan, A. K. Banerjee, "Lattice based forward-secure identity based encryption scheme", *Journal of Internet Services and Information Security*, vol. 3, no. 1/2, pp. 5-19, 2013.
[14] A. Lewko, B. Waters, "New technique for dual system encryption and fully secure HIBE with short ciphertexts", in *The Seventh Theory of Cryptography Conference (TCC 2010)*, pp. 455–479, Zurich, Switzerland, 2010.
[15] B. Libert, J. Quisquater, M. Yung, "Forward-secure signatures in untrusted update environments: Efficient and generic constructions", in *The 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 266–275, Alexandria, VA, USA, 2007.
[16] B. Libert, J. Quisquater, and M. Yung, "Key evolution systems in untrusted update environments", *ACM Transactions on Information and Systems Security*, vol. 13, no. 4, pp. 1-34, 2010.
[17] T. Malkin, D. Micciancio, and S. K. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods", in *Advances in Cryptology (EUROCRYPT'02)*, pp. 400–417, Amsterdam, The Netherlands, 2002.
[18] J. Nieto, M. Manulis, D. D. Sun, "Forward-secure hierarchical predicate encryption", in *Pairing-Based Cryptography (Pairing'12)*, pp. 83–101, Cologne, Germany, 2012.
[19] Y. L. Ren, Z. H. Niu, X. P. Zhang, "Fully anonymous identity-based broadcast encryption without random oracles", *International Journal of Network Security*, vol. 16, no. 4, pp. 256–264, 2014.
[20] D. Yao, Y. Dodis, and N. Fazio, et. al., "ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption", in *The eleven ACM Conference on Computer and Communication Security*, pp. 354–363, Washington, DC, USA, 2004.
[21] J. Yu, F. Y. Kong, and X. G. Cheng, et. al., "Forward-secure identity-based public-key encryption without random oracles", *Fundamenta Informatica*, vol. 111, no. 2, pp. 241–256, 2011.

**Xiujie Zhang** biography. Xiujie Zhang is a lecturer in Weifang University. She received her PhD from University of Electronic Science and Technology of China(UESTC). Her research interests include leakage resilient cryptosystems, applied cryptography and information security.

**Chunxiang Xu** biography. Chunxiang Xu received her BS, MS and PhD from Xidian University, China, in 1985, 1988 and 2004, respectively. She is a professor at UESTC. Her research interests include information security, cloud computing security and cryptography.