Self-Updatable Encryption: Time Constrained Access Control with Hidden Attributes and Better Efficiency

ASIACRYPT 2013

Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park and Moti Yung

Korea University, US Naval Academy, Korea University, Sangmyung University, Google Inc. and Columbia University

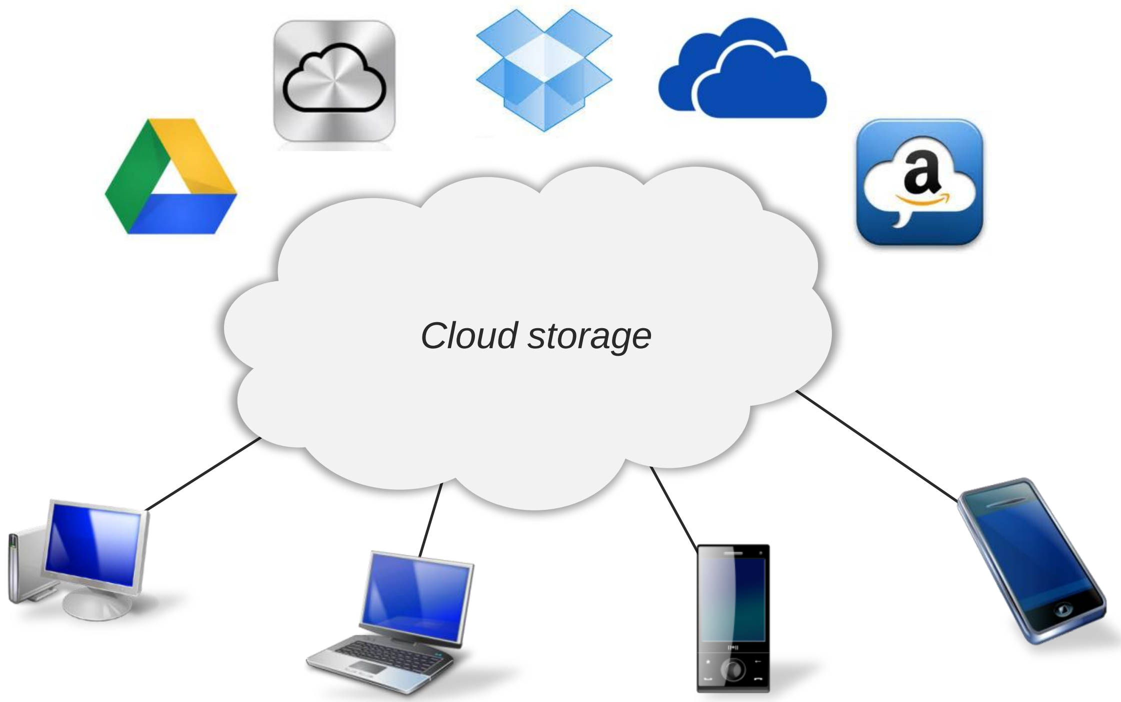
Overview

- Motivation
  - A revocable-storage attribute-based encryption (RS-ABE) is a good access control mechanism for cloud storage by supporting *key-revocation* and *ciphertext-update*
  - We ask whether it is possible to have a modular approach for RS-ABE by using a primitive for time-evolution mechanism

- Results
  - We introduce a *self-updatable encryption (SUE)* for a time evolution mechanism, and construct an efficient SUE scheme
  - We present a new *revocable-storage attribute-based encryption (RS-ABE)* scheme with shorter ciphertexts
  - We also obtain a *revocable-storage predicate encryption (RS-PE)* scheme that supports attribute-hiding property

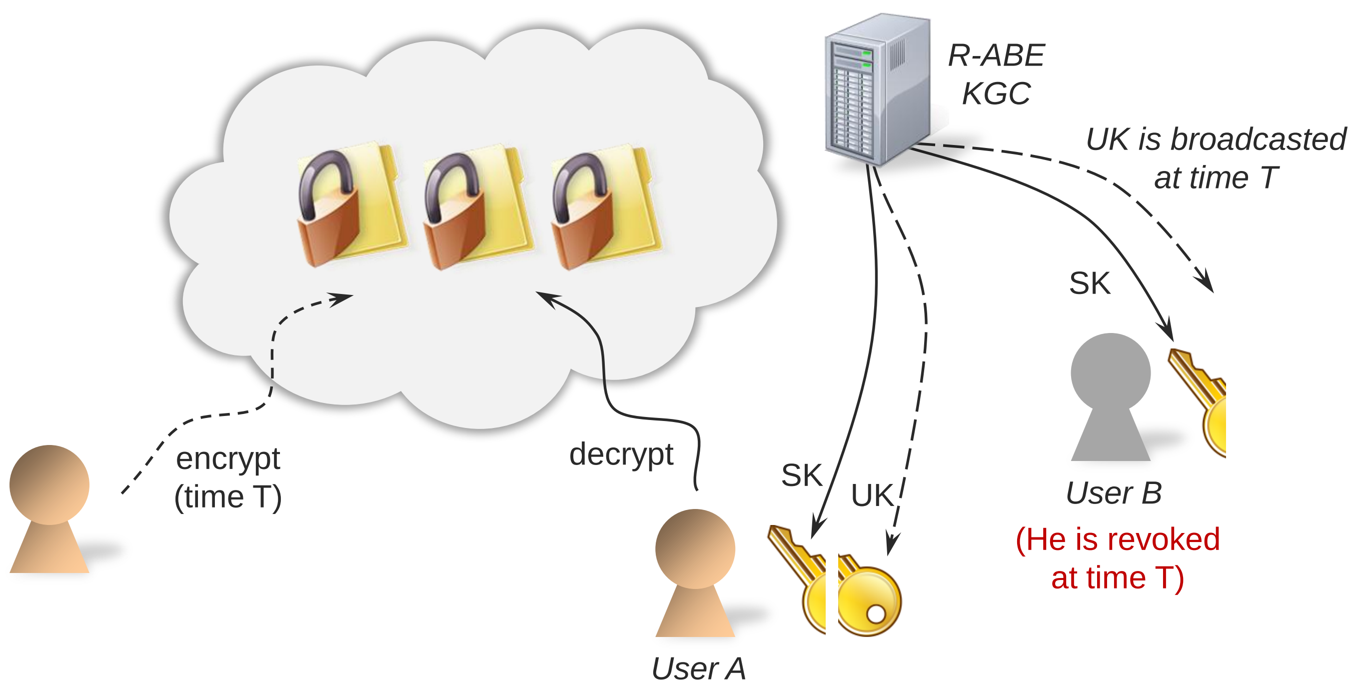
Introduction

- Cloud Storage
  - Cloud data storage has many advantages: A virtually unlimited amount of space can be allocated, and storage management can be easier
  - Moreover, it provides great accessibility: Users in any geographic location can access their data through the Internet



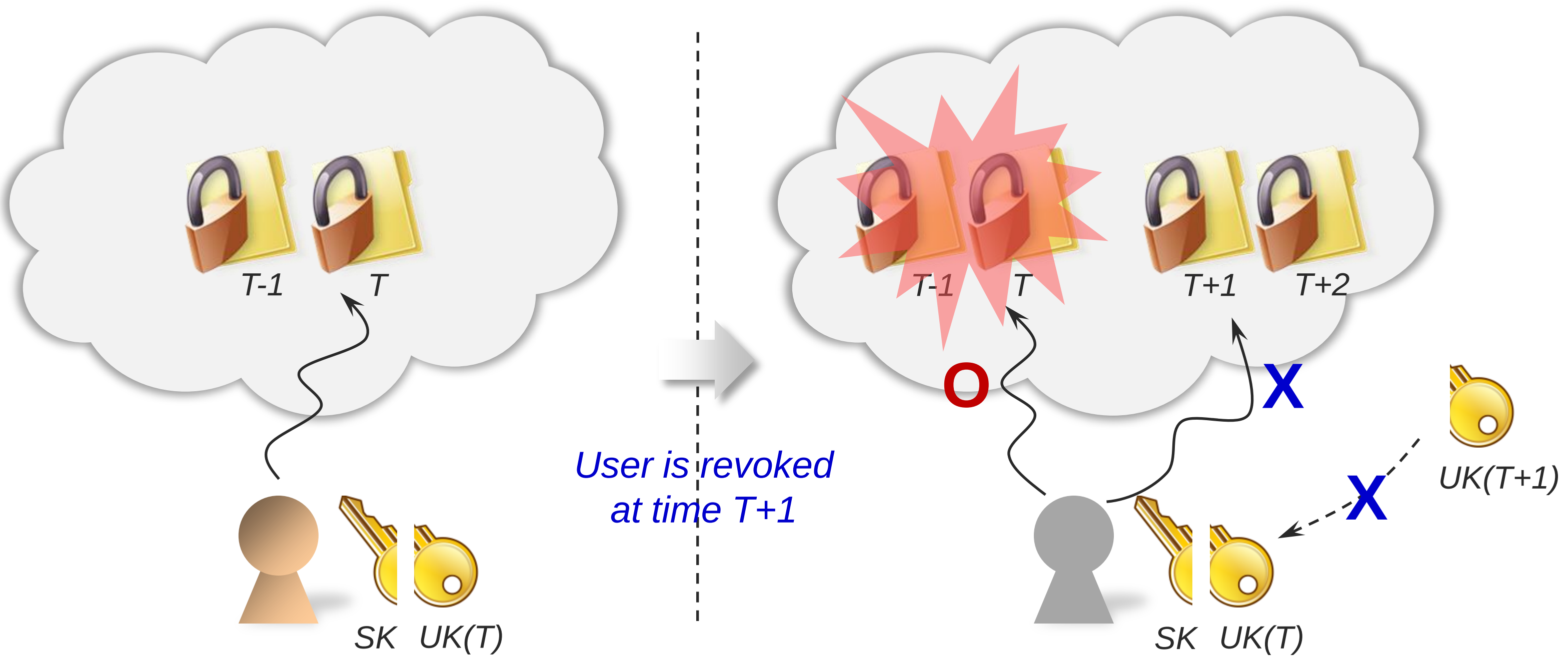
Introduction

- Access Control for Cloud Storage
  - Access control is one of greatest concerns: the senstive data should be protected from any illegal access from outsiders or from insiders
  - A revocable ABE (R-ABE) can be used for access control in cloud storage by revoking a user's private key if his credential is expired



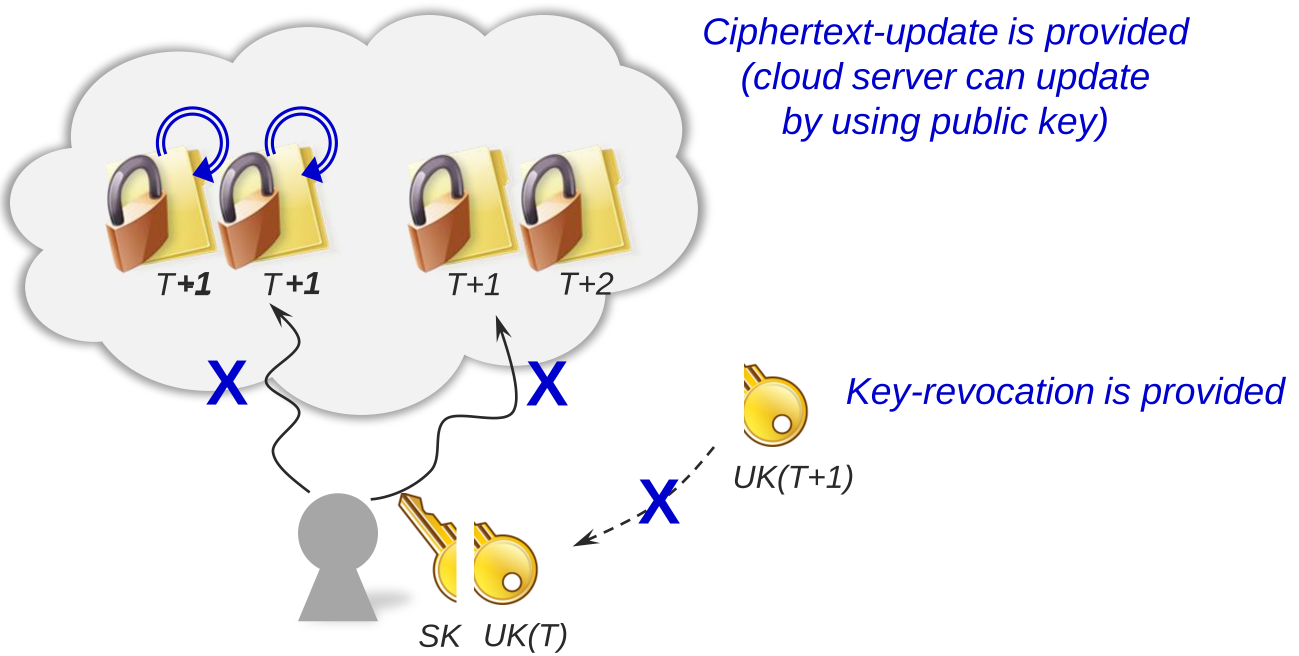
Introduction

- Novel Concern in Cloud Storage
  - Sahai, Seyalioglu, and Waters (Crypto 2012) pointed out that R-ABE alone does not suffice in managing dynamic credentials for cloud storage
  - R-ABE cannot prevent *a revoked user from accessing ciphertexts that were created before the revocation*, since the old private key is enough for decryption



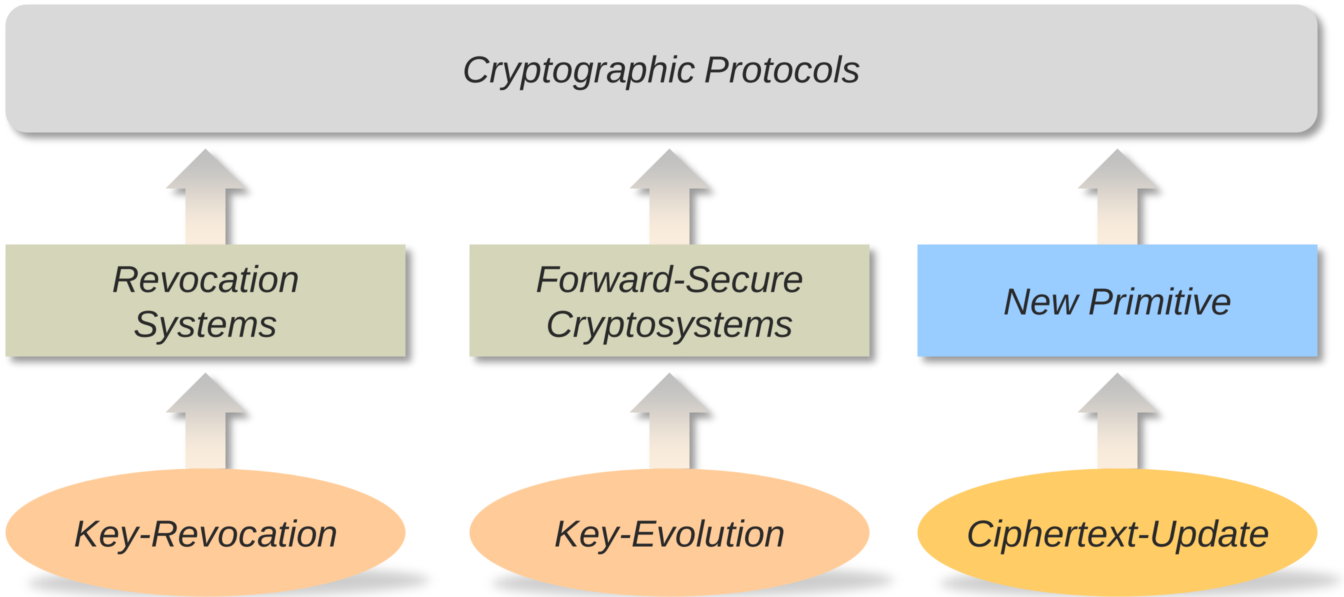
Introduction

- Revocable-Storage ABE
  - To solve the previous issue, Sahai et al. introduced a novel RS-ABE that supports not only *key-revocation* but also *ciphertext update*
  - That is, a ciphertext at any time *T* can be updated to a new ciphertext at time *T+1* by any party *just using the public key* (by the cloud server)



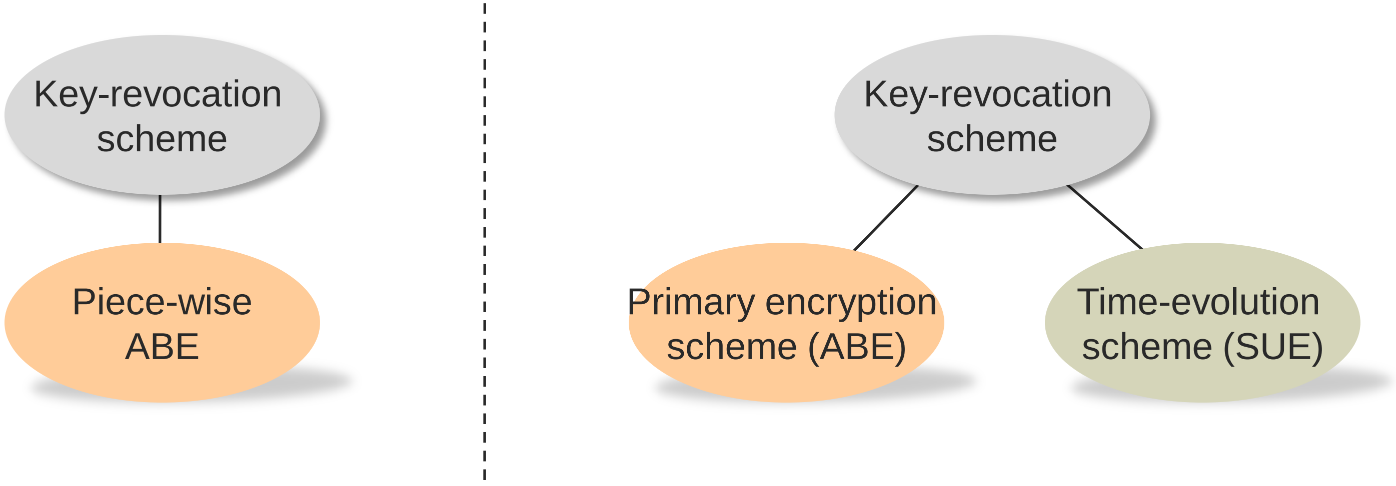
Introduction

- Our Motivation
  - Key-revocation and key-evolution are importance issues in cryptosystem design, and *ciphertext-update (time-evolution)* can be useful elsewhere
  - We want to achieve ciphertext-update (time-evolution) in other encryption scheme and use it as an underlying primitive



Introduction

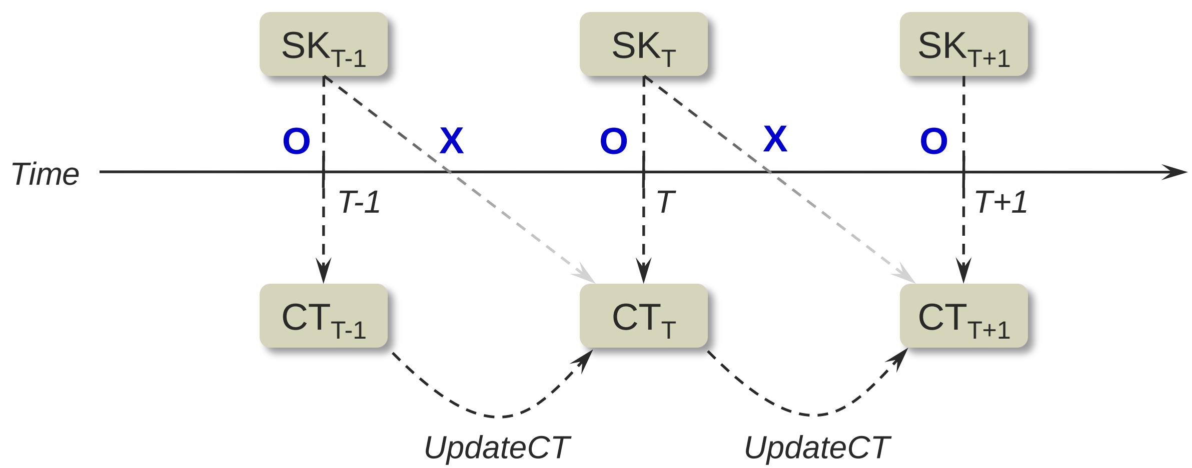
- Our Approach
  - We take a modular approach for RS-ABE by combining three components: a primary encryption scheme, a key-revocation mechanism, and a time-evolution mechanism
  - This approach has potential benefits since each mechanism may have independent interest and it may open the door to optimizations



The previous approach *Our modular approach*

Self-Updatable Encryption

- Overview
  - Self-updatable encryption (SUE) is a new cryptographic primitive that realizes a time-evolution mechanism
  - A private key and a ciphertext are associated with time T_k and T_c, and a private key for T_k can decrypt a ciphertext for T_c if $T_c \leq T_k$
  - Additionally, anyone can update a ciphertext with time T_c to a new ciphertext with new time T_c+1



Self-Updatable Encryption

- Definition
  - SUE is a new type of PKE with the ciphertext updating property (time-evolution mechanism)
  - An SUE scheme consists of algorithms: Setup, GenKey, Encrypt, UpdateCT, RandCT, and Decrypt

Setup(T_{max}) → MK,PP

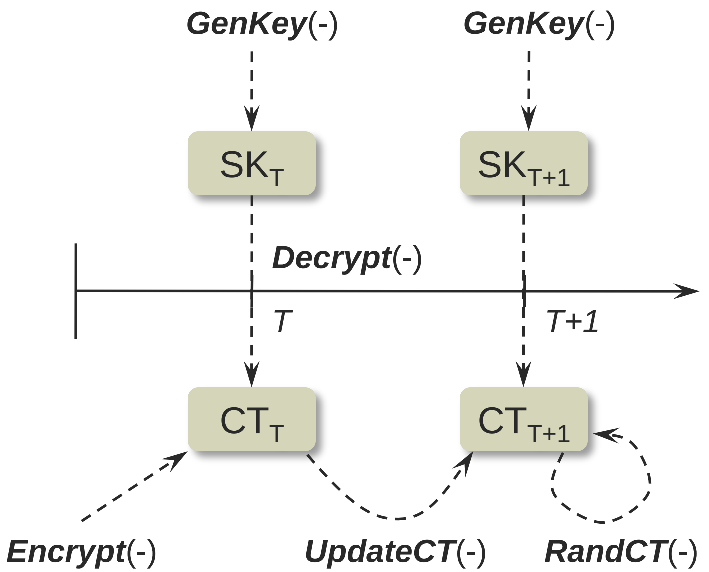
GenKey(T,MK,PP) → SK_T

Encrypt(T,M,PP) → CT_T

UpdateCT(CT_T,T+1,PP) → CT_{T+1}

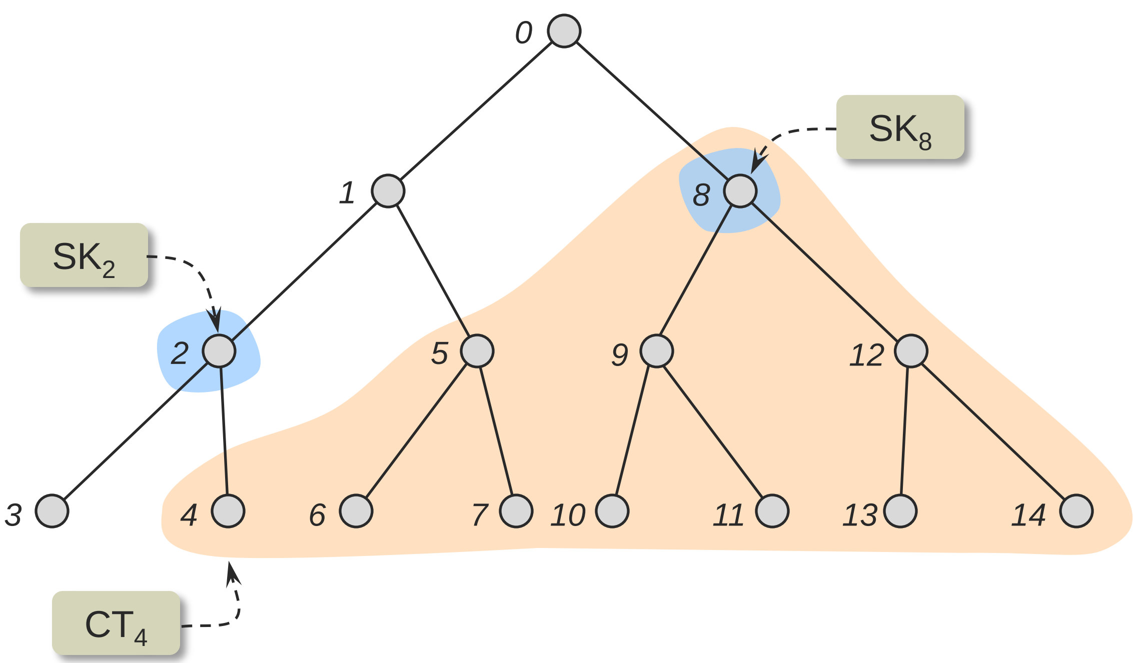
RandCT(CT_T,PP) → CT_T

Decrypt(CT_T,$SK_{T'}$,PP) → M



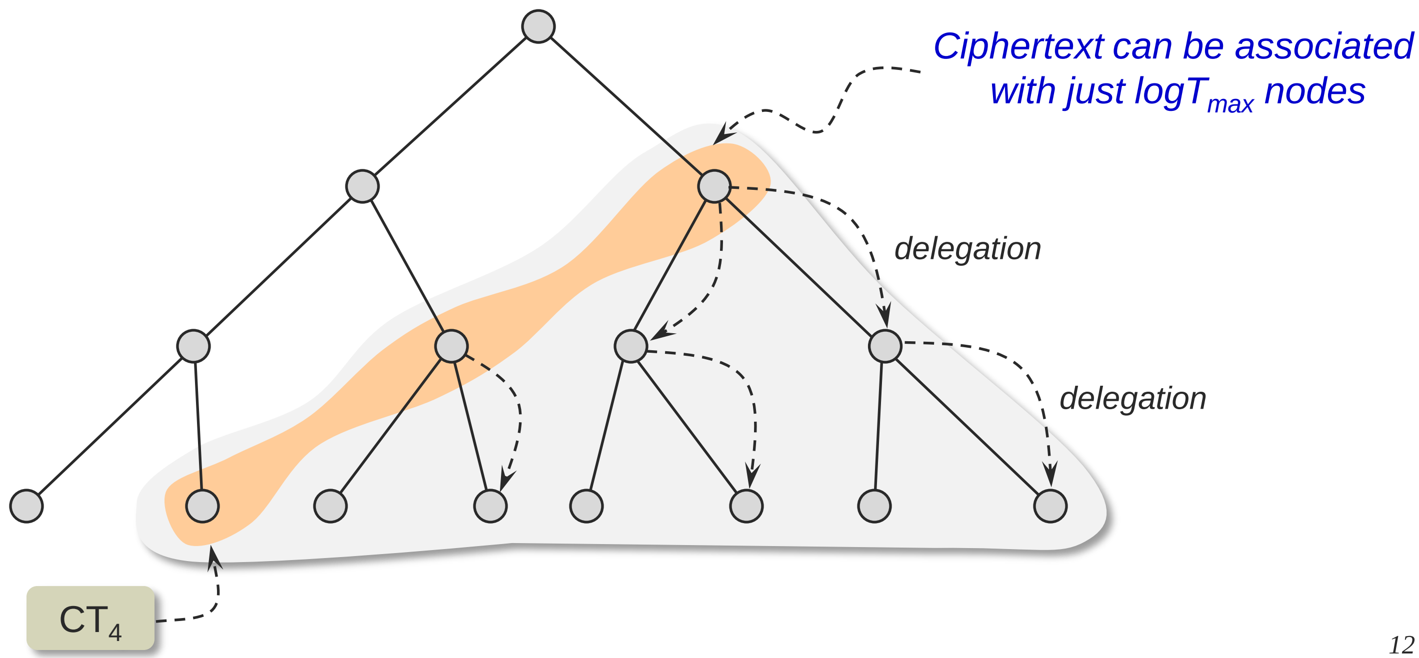
Self-Updatable Encryption

- Design Principle
  - A full binary tree is used to represent time by assigning time periods to tree nodes in pre-order traversal
  - A private key for time T_k is associated with a node v_k and a ciphertext for time T_c is associated with nodes $\{v_i\}$ for all time $T_i \geq T_c$



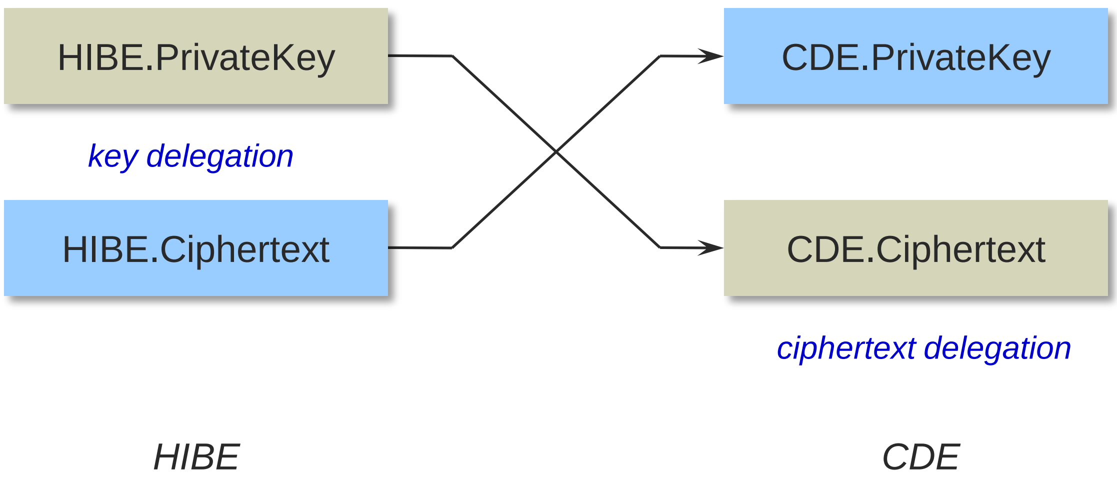
Self-Updatable Encryption

- Design Principle
  - If a ciphertext has *the delegation property* such that it's association can be changed from a node to it's chid node, then ciphertext can be shorten
  - The design idea of SUE is similar to that of forward-secure encryption, but ciphertexts are delegated in SUE (not private keys)



Self-Updatable Encryption

- Ciphertext Delegatable Encryption
  - CDE is a new type of PKE that has the ciphertext delegation property, and it can be used to build an SUE scheme
  - A CDE scheme could be derived from an HIBE scheme by switching the structure of private keys and that of ciphertexts



Self-Updatable Encryption

- Ciphertext Delegatable Encryption
  - We start from the HIBE scheme of Boneh and Boyen (Eurocrypt 2004) to derive a CDE scheme
  - The ciphertext delegation property of CDE could be obtained from the key delegation property of HIBE

BB_HIBE

$$SK = [g^{\beta} F_1(I_1)^{r_1}, g^{r_1}]$$

$$SK' = [g^{\beta} F_1(I_1)^{r_1} F_2(I_2)^{r_2}, g^{r_1}, g^{r_2}]$$

$$CT = [g^{s}, F_1(I_1)^{s}, F_2(I_2)^{s}]$$

CDE

$$SK = [g^{\beta} w^{r}, g^{r}, F_1(L_1)^{r}, F_2(L_2)^{r}]$$

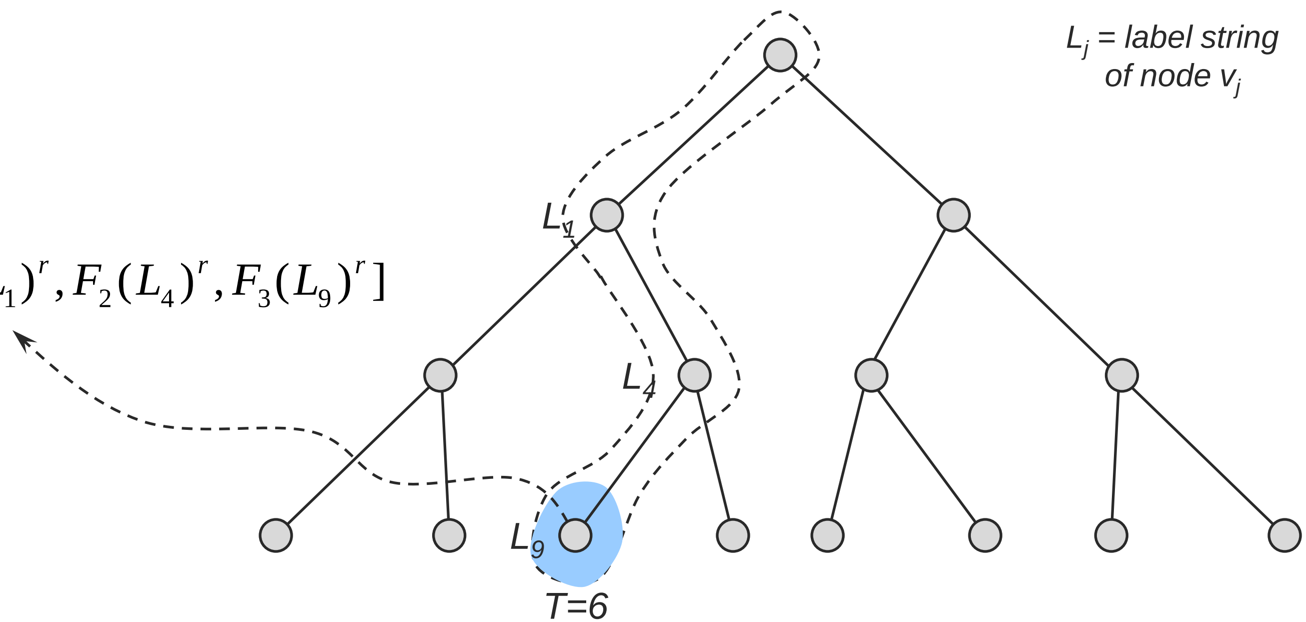
$$CT = [g^{s}, w^{s} F_1(L_1)^{s_1}, g^{s_1}]$$

$$CT' = [g^{s}, w^{s} F_1(L_1)^{s_1} F_2(L_2)^{s_2}, g^{s_1}, g^{s_2}]$$

Self-Updatable Encryption

- SUE Construction
  - $SK_T \leftarrow$ **GenKey**(T, MK, PP): The private key of SUE for time T is associated with path nodes Path(v) from the root node to a tree node v where v is associated with T

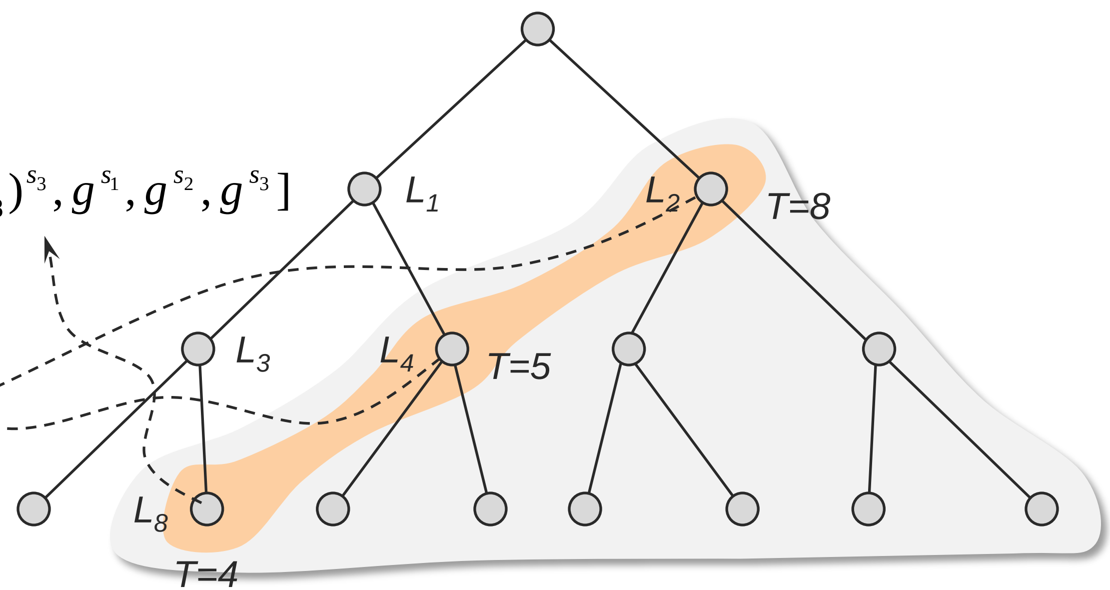
$$SK_6 =$$
$$[g^{\beta} w^{r}, g^{r}, F_1(L_1)^{r}, F_2(L_4)^{r}, F_3(L_9)^{r}]$$



Self-Updatable Encryption

- SUE Construction
  - CT_T←**Encrypt**(T, PP): The ciphertext of SUE for time T consists of ciphertexts of CDE for root nodes of all subtrees that cover all time $T_i \geq T$
  - The number of group elements in SUE can be reduced from $O(\log^2 T_{max})$ to $O(\log T_{max})$ by carefully reusing the randomness of CDE

$$CT_4 =$$
$$[g^s, w^s F_1(L_1)^{s_1} F_2(L_3)^{s_2} F_3(L_8)^{s_3}, g^{s_1}, g^{s_2}, g^{s_3}]$$
$$[g^s, w^s F_1(L_1)^{s_1} F_2(L_4)^{s'_2}, g^{s'_2}]$$
$$[g^s, w^s F_1(L_2)^{s'_1}, g^{s'_1}]$$

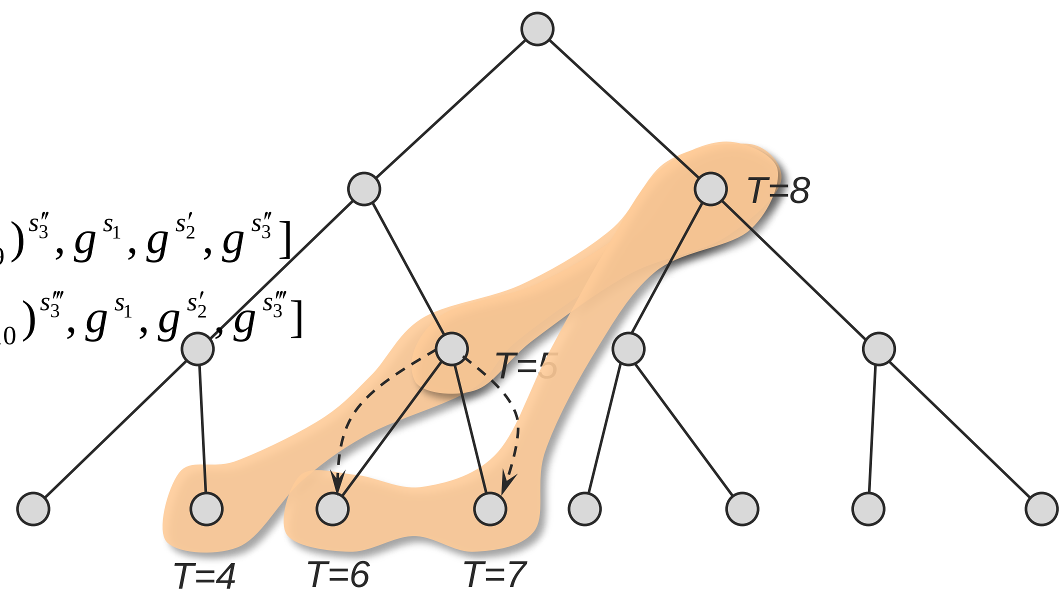


Self-Updatable Encryption

- SUE Construction
  - CT_{T+1}←**UpdateCT**(CT_T, $T+1$, PP): The ciphertext of SUE can be updated to next time by using the ciphertext delegation algorithm of CDE

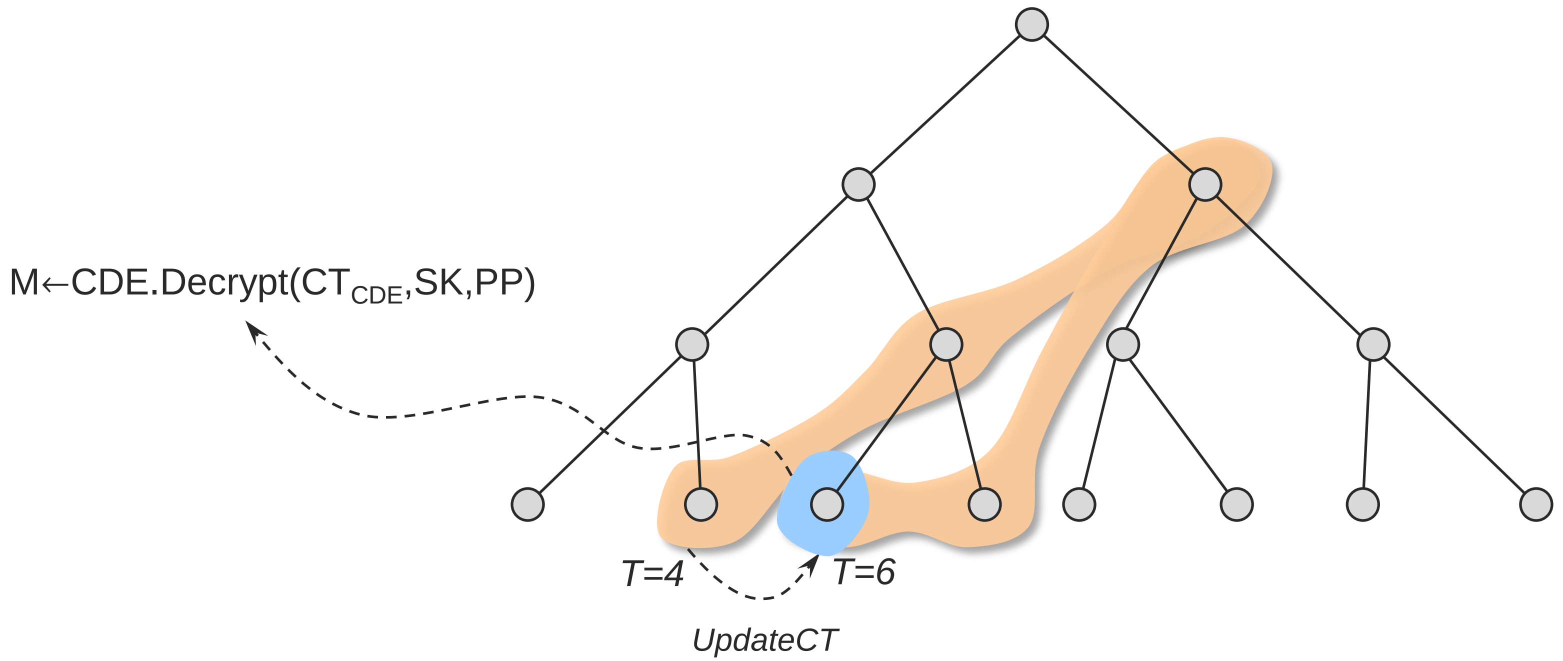
$$CT_5 =$$
$$[g^{s}, w^{s} F_1(L_1)^{s_1} F_2(L_4)^{s'_2}, g^{s_1}, g^{s'_2}]$$
$$[g^{s}, w^{s} F_1(L_2)^{s'_1}, g^{s'_1}]$$

$$CT_6 =$$
$$[g^{s}, w^{s} F_1(L_1)^{s_1} F_2(L_4)^{s'_2} F_3(L_9)^{s''_3}, g^{s_1}, g^{s'_2}, g^{s''_3}]$$
$$[g^{s}, w^{s} F_1(L_1)^{s_1} F_2(L_4)^{s'_2} F_3(L_{10})^{s'''_3}, g^{s_1}, g^{s'_2}, g^{s'''_3}]$$
$$[g^{s}, w^{s} F_1(L_2)^{s'_1}, g^{s'_1}]$$



Self-Updatable Encryption

- SUE Construction
  - $M \leftarrow$ **Decrypt**$(CT_T, SK_{T'}, PP)$: If $T \leq T'$, then a CDE ciphertext in SUE ciphertext can be decrypted by using the decryption algorithm of CDE



Self-Updatable Encryption

- Discussions
  - **Efficiency**: The number of group elements in *SK* is O(logT_{max}) and the number of group elements in *CT* is O(logT_{max})
  - **Exponential Number of Time Periods**: Our SUE scheme can support an exponential number (2^{λ}) of time periods by setting the tree depth to be the security parameter
  - **Time Interval**: By combining two SUE schemes (one for future SUE and another for past SUE), we expect to build an SUE scheme for time interval [T_L,T_R]
  - **Differenct Constructions**: We expect that different HIBE schemes will result different SUE schemes with different efficiency tradeoffs

Revocable-Storage ABE

- Definition
  - RS-ABE is an attribute-based encryption (ABE) that additionally supports both *key revocation* and *ciphertext update*
  - RS-ABE consits of algorithms: Setup, GenKey, UpdateKey, Encrypt, UpdateCT, RandCT, and Decrypt

Setup(…) → MK,PP

GenKey(S,u,MK,PP) → $SK_{S,u}$

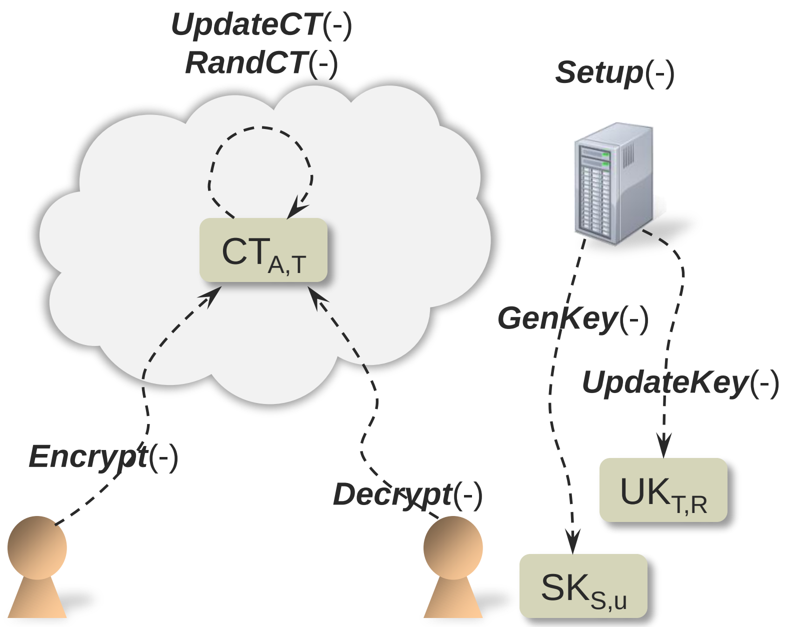
UpdateKey(T,R,MK,PP) → $UK_{T,R}$

Encrypt($\mathbb{A}$,T,M,PP) → $CT_{\mathbb{A},T}$

UpdateCT($CT_{\mathbb{A},T}$,T+1,PP) → $CT_{\mathbb{A},T+1}$

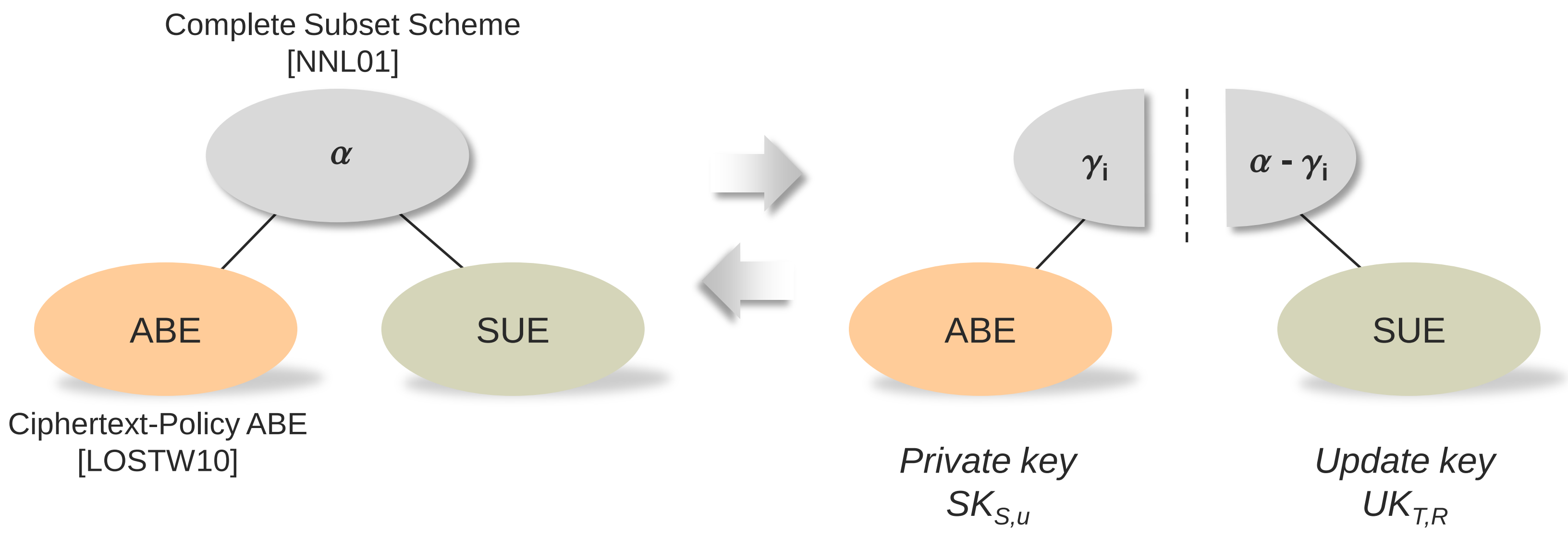
RandCT($CT_{\mathbb{A},T}$,PP) → $CT_{\mathbb{A},T}$

Decrypt($CT_{\mathbb{A},T}$,$SK_{S,u}$,$UK_{T',R}$,PP) → M



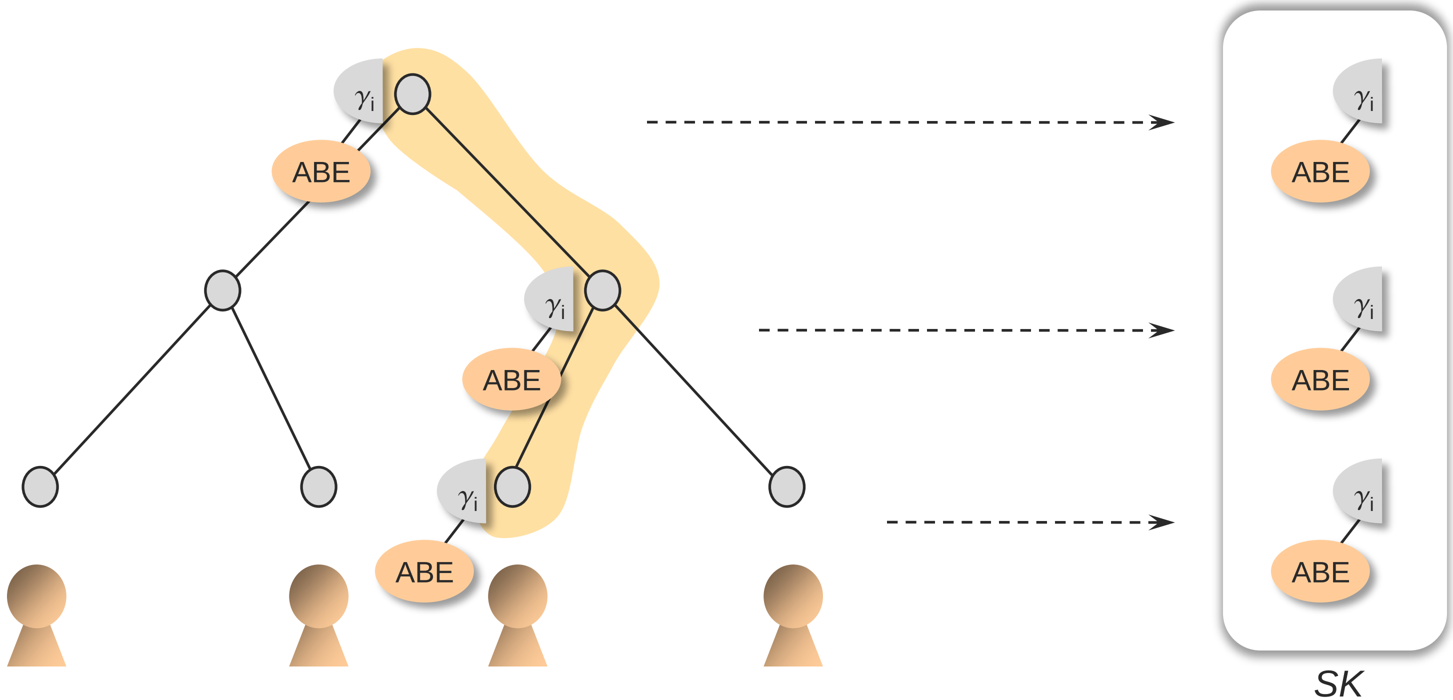
Revocable-Storage ABE

- Design Principle
  - Our scheme combines three components: a primary encryption scheme (CP-ABE), a key-revocation scheme, and a time-evolution scheme (SUE)
  - To prevent collusion-attacks, the key-revocation scheme uses a secret-sharing method when it combines two encryption components



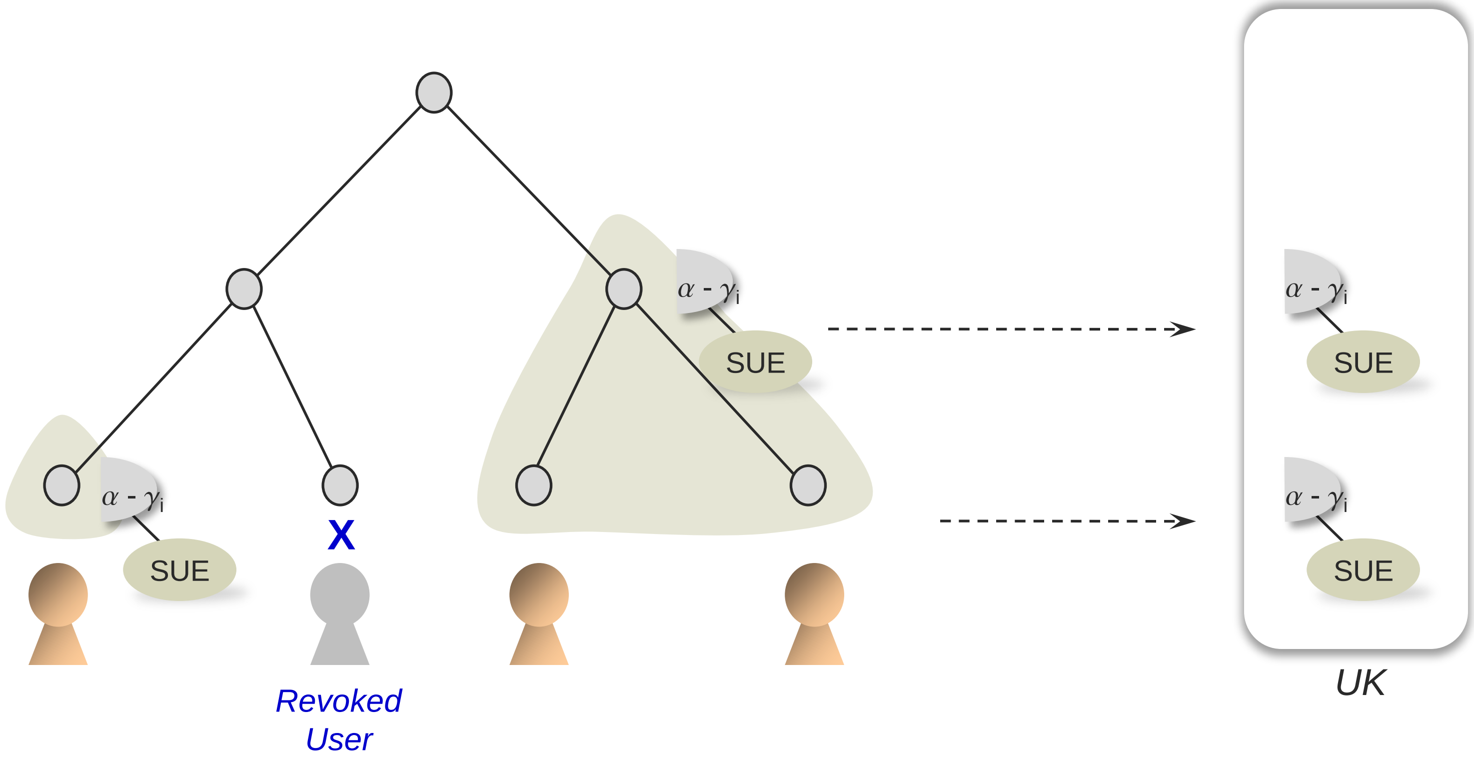
Revocable-Storage ABE

- RS-ABE Construction
  - **GenKey**: A private key (*SK*) consists of ABE private keys associated with path nodes of a user where the user is assigned to a leaf node of a binary tree of the CS scheme



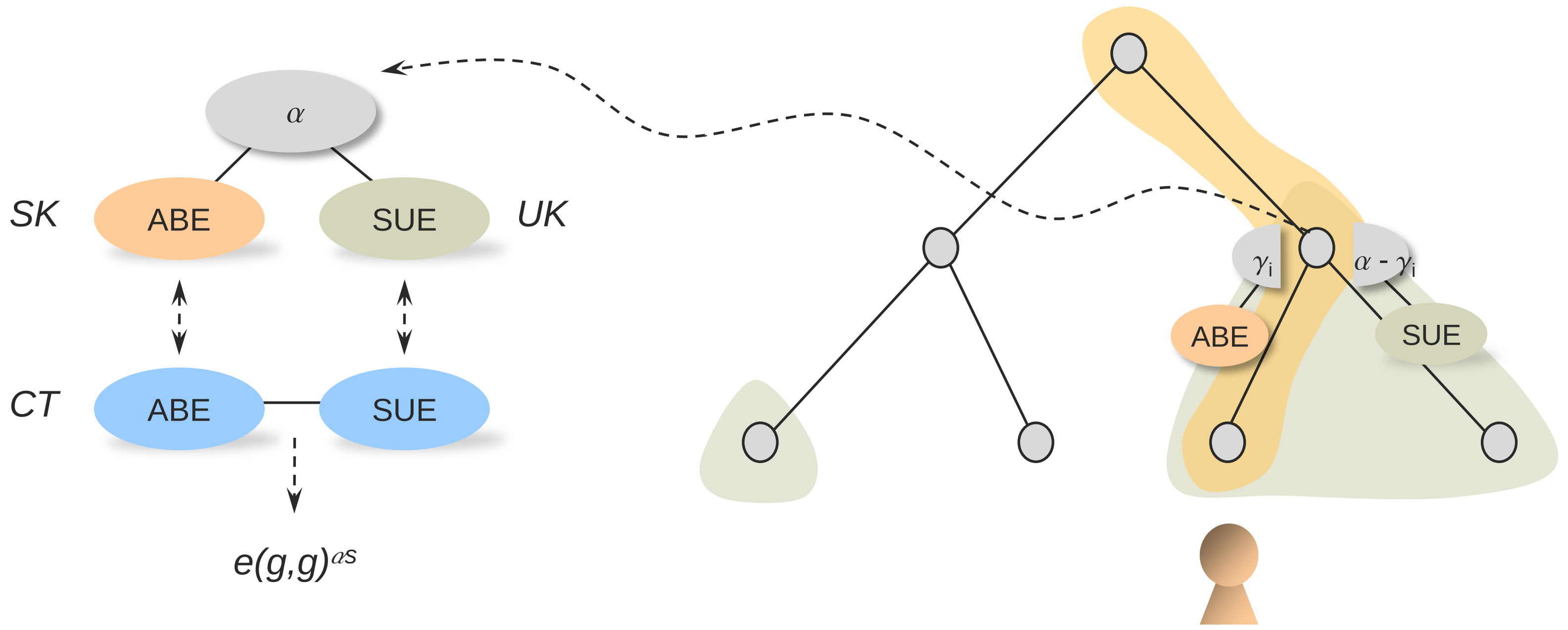
Revocable-Storage ABE

- RS-ABE Construction
  - **UpdateKey**: An update key (*UK*) consists of SUE private keys associated with covering subsets for non-revoked users (i.e. root nodes of subtrees that cover non-revoked users)



Revocable-Storage ABE

- RS-ABE Construction
  - **Encrypt**: A ciphertext (*CT*) consits of an SUE ciphertext and an ABE ciphertext with the same random exponent for secret sharing
  - **Decrypt**: If a user is not revoked ($u \notin R$) at time T, then a ciphertext with time T can be decrypted by an SUE private key from *SK* and an ABE private key from *UK*



Conclusion

- Other Applications
  - **Revocable-Storage Predicate Encryption** (RS-PE): By using an inner-product encryption (IPE) scheme as a primary encryption scheme, we can build an RS-PE scheme that provides *the attribute-hiding property* in ciphertexts
  - **Timed-Release Encryption** (TRE): TRE is a PKE such that a ciphertext with time T can be decrypted after T. An SUE scheme can be used to build a TRE scheme
  - **Key-Insulated Encryption** (KIE) with Ciphertext Forward Security: KIE is a PKE that provides tolerance against key exposures. By combining KIE and SUE schemes, we can build a KIE scheme with forward-secure storage

Thank You