SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2025, Vol. 68, Iss. 3, 132109:1–132109:16
https://doi.org/10.1007/s11432-024-4214-0

# Witness encryption with updatable ciphertexts

Yuzhu WANG[1] & Mingwu ZHANG[1,2*]

[1]*School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*School of Computer Science, Hubei University of Technology, Wuhan 430068, China*

Received 27 July 2024/Revised 21 October 2024/Accepted 11 November 2024/Published online 10 February 2025

**Abstract** Witness encryption (WE) is a novel type of cryptographic primitive that enables a message to be encrypted via an NP instance. Anyone who possesses a solution to this instance (i.e., a witness) can then recover the message from the ciphertext. We introduce a variant of WE that allows ciphertext updates, referred to as ciphertext updateable WE (CUWE). With CUWE, a user can encrypt a message using an instance $x$ and a tag $t$, and those who possess a valid witness $w$ for $x$ and match the access policy defined by tag $t$ can decrypt the message. Furthermore, CUWE allows for the use of an update token to change the tag $t$ of ciphertext to a different tag. This feature enables fine-grained access control, even after the ciphertext has been created, thereby significantly increasing the usefulness of the WE scheme. We demonstrate that such a WE framework with an updatable ciphertext scheme can be constructed using our puncturable instance-based deterministic encryption (PIDE) and indistinguishability obfuscation ($i\mathcal{O}$). We also propose an instantiation of PIDE utilizing puncturable pseudorandom functions (PRFs) that provide (selectively) indistinguishable security. Finally, we expand our CUWE to ciphertext-updatable functional WE (CUFWE), which offers enhanced data access control.

**Keywords** witness encryption, NP relation, updatable ciphertext, access control, indistinguishable obfuscation

**Citation** Wang Y Z, Zhang M W. Witness encryption with updatable ciphertexts. Sci China Inf Sci, 2025, 68(3): 132109, https://doi.org/10.1007/s11432-024-4214-0

## 1 Introduction

Witness encryption (WE) is a powerful cryptography primitive proposed by Garg et al. [1] that allows a message to be encrypted under an instance $x$ of an NP language $\mathcal{L}$ with $\mathcal{L} = \{x \mid \exists\ w : R(x, w) = 1\}$. Any entity possessing a valid witness $w$ satisfying $R(x, w) = 1$ can recover the encrypted message. For security purposes, if the instance $x \notin \mathcal{L}$, no PPT adversary can distinguish between encryptions of any two messages of equal length.

WE has many interesting applications. It can be used to construct powerful cryptographic primitives such as public key encryption (PKE) [1], fully secure identity-based encryption [1], attribute-based encryption for circuits [2], and broadcast encryption [3]. It also enables a variety of cryptographic protocols, including time-lock encryption [4, 5], oblivious transfer [6], conditional payments in blockchain-based cryptocurrencies [7], multiparty reusable non-interactive secure computation (mrNISC) [8], (verifiable) encryption to the future (EtF) [9, 10], and paid data subscriptions [11]. Furthermore, WE allows for applications that were previously considered impossible. For example, as shown in [1], it is possible to encrypt a message concerning a puzzle such that only those who solve the puzzle can decrypt it. This puzzle could represent various challenges, including NP-complete problems like crossword puzzles, the SAT problem [4], the subset-sum problem [4], or even proofs of certain mathematical conjectures.

Intuitively, WE can be seen as an extension of PKE [12]. A user can encrypt a message $m$ under an NP instance $x$ (analogous to a public key $\mathrm{pk} = x$), while the witness $w$ serves as the decryption key (similar to a private key $\mathrm{sk} = w$) [13]. However, unlike classical PKE, WE does not require a key generation phase. This eliminates the need for the secure transmission of a secret key and ensures that decryption is accessible to anyone who possesses a valid witness $w$. Specifically, in classic PKE, the sender uses the recipient's public key (or identity, attributes, etc.) to encrypt the message, and only the recipient possessing the corresponding private key can decrypt it. WE departs from this model by allowing decryption for anyone who knows a $w$ such that $R(x, w) = 1$. Essentially, WE compensates for

* Corresponding author (email: csmwzhang@gmail.com)

 

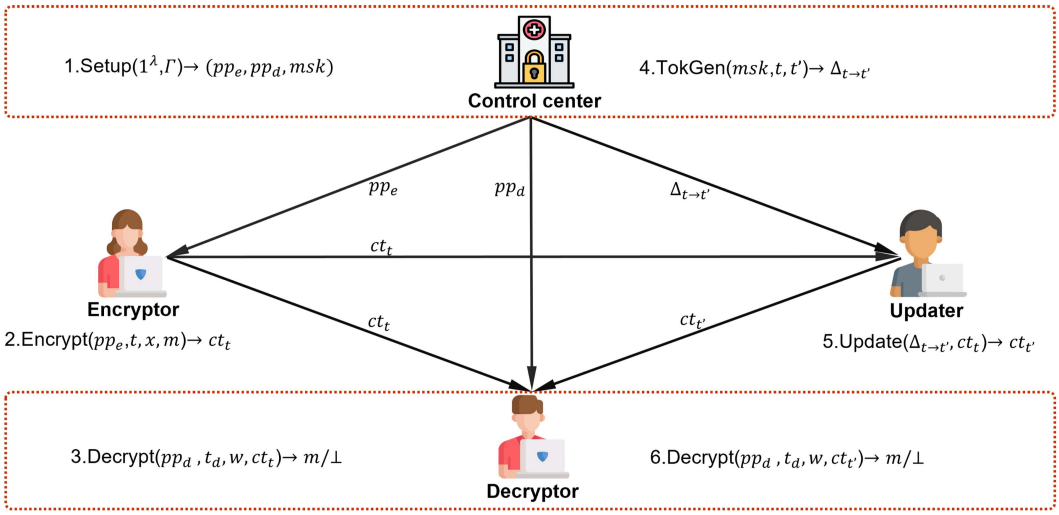


**Figure 1** (Color online) Example of witness encryption with updatable ciphertexts.

the absence of a key distribution process by enabling decryption through the possession of a valid witness $w$.

**Motivation.** Despite the significant progress made over the last few years [14–16], limitations in WE still persist. For example, as described in [1], the original motivation for WE was to facilitate cryptography-based prize competitions. Here, the prize (e.g., a Bitcoin address) is encrypted using a mathematical puzzle such that anyone who solves the puzzle can successfully decrypt and get a prize. However, in most prize contests, the solution to the puzzle must be obtained within a specific timeframe to claim the prize. We now envision WE with fine-grained access control, referred to as tag-based WE. Specifically, the message $m$ is encrypted using an NP instance $x$ and a tag $t$ (e.g., a timestamp, label, or other identifier). Decryption then requires a valid $w$, where $R(x, w) = 1$, and satisfies the access policy defined by tag $t$ (i.e., $\Gamma(t) = 1$).

In addition, Freitag et al. [2] demonstrated how to build flexible broadcast encryption (FBE) by employing WE and function-binding hash functions. If their FBE is derived from our tag-based witness encryption, it can be seamlessly extended to attribute-based broadcast encryption [17], which enables users to broadcast encrypted messages to a specified group of users while enforcing specific access policies. However, challenges arise when newly authorized users require access to the data. Addressing this typically involves either re-encrypting all data to align with the policy that the newly authorized users satisfy or issuing additional keys to these users.

We now envision a system that not only offers fine-grained control but also allows competition platforms or broadcast platforms to update the policy (e.g., timestamp, attribute, identity) in the existing ciphertext, adding a layer of flexibility. Simultaneously, we maintain the inherent advantage of witness encryption, which eliminates the need for secure key transmission. Currently, no WE schemes in the cryptographic literature offer these advanced features. Therefore, our study seeks to address the following critical question:

Can we define and build a witness encryption framework that incorporates fine-grained control and ciphertext updatability?

**Our work.** To address the above question, we propose defining and constructing ciphertext updateable WE (CUWE) that supports fine-grained access control. To illustrate our CUWE concept, we consider a toy example (as shown in Figure 1) set in a prize competition. This competition involves four types of entities: the control center (e.g., the competition platform), the encryptor, the decryptor, and the updater.

As the first step, the control center generates the encryption parameters $\mathrm{pp}_e$, decryption parameters $\mathrm{pp}_d$, and the master private key msk (kept secret) based on the security parameter and the competition rules $\Gamma$ (e.g., the competition time or registration identity). For instance, if $\Gamma(t) = 1$, it means that the time condition is met, whereas $\Gamma(t) = 0$ signifies non-compliance. Notably, unlike the Setup algorithm of PKE, both the encryption and decryption parameters here can be made publicly available. Next, the encryptor encrypts a message $m$ using the encryption parameters, NP instance $x$, and a tag $t$ of

decryption time, producing a ciphertext $\mathrm{ct}_t$. Using the decryption parameter $\mathrm{pp}_e$, the current timestamp $t_d$ ($t_d$ satisfies $t$, such as $t_d < t$), and a witness $w$ where $R(x, w) = 1$, the decryptor can decrypt $\mathrm{ct}_t$ to obtain $m$. Unlike existing WE schemes, decryption requires the current timestamp $t_d$. In addition to the above three algorithms, which are the same as the existing WE [18, 19], our CUWE introduces two additional algorithms to support updatable ciphertext. To update tag $t$ in the ciphertext $\mathrm{ct}_t$, the control center generates an update token $\triangle_{t\to t'}$ using the master private key msk. The updater, which can be any honest-but-curious party, utilizes this update token to transform the ciphertext $\mathrm{ct}_t$ to $\mathrm{ct}_{t'}$. We limit the update of the ciphertext to a single operation, and the update token is designed to function in a single direction, specifically from tag $t$ to $t'$, rather than reverse.

Correctness ensures that when the current tags $t_d$ of the decryptor and the ciphertext match, a valid witness $w$ can be used to successfully decrypt and obtain $m$. Concerning security, we aim to update the ciphertext according to a tag of the update token while ensuring that the updated ciphertext cannot undergo additional updates. Specifically, the token $\triangle_{t\to t'}$ is able to transfer tags from $t$ to $t'$ rather than the reverse direction. For our CUWE constructions, we adopt the notation of Gorbunov et al. [20], who developed attribute-based encryption (ABE) for circuits of arbitrary polynomial size. Here, tag $t \in \{0,1\}^l$ represents the $l$-bit public index (used for Encrypt), and $\Gamma$ denotes the Boolean predicate associated with the decryption parameter. Decryption should only work if $\Gamma(t) = 1$ and there exists a valid witness $w$ such that $R(x, w) = 1$. We can simply associate the decryption parameter with a predicate $\Gamma$ (encode them as $\mathrm{pp}_d$) and use tag $t$ (i.e., attribute) as the public tag of the puncturable instance-based deterministic encryption (PIDE) scheme. In the decryption circuit $P_d$, we only need to check whether tag $t$ and the hard-coded $\Gamma$ satisfy $\Gamma(t) = 1$ and whether the witness $w$ satisfies $R(x, w) = 1$.

Since the concept of update tokens does not exist in the current WE, we need to deal with other aspects of the notion of security. Specifically, we must recognize that these tokens can be utilized not only to update the ciphertext from tag $t \in \mathcal{T}$ to $t' \in \mathcal{T}\backslash\{t\}$ but also to reverse the ciphertext update. This issue closely resembles the challenge of offering adequate and robust safety in proxy re-encryption (PRE) [21, 22]. With these considerations in mind, we have defined a concept of indistinguishability IND-CUWE-CPA, which ensures an adversary cannot distinguish between the ciphertext of a specific challenge target tag, the NP instance, and the message chosen by the adversary.

Our goal is to enable a one-time change of identifiers for ciphertexts restricted to a single direction. To capture these restrictions, we offer access to four oracles ($\mathsf{TokGen}_c, \mathsf{TokGen}_h, \mathsf{Encrypt}^*, \mathsf{Update}_h$ for the adversary). These oracles are detailed below for comprehensive discussion. We enable the adversary to adaptively query the corrupted update token $\mathsf{TokGen}_c$ and the honest update token $\mathsf{TokGen}_h$, and then deliver encrypt oracle $\mathsf{Encrypt}^*$ and honest ciphertext-update oracle $\mathsf{Update}_h$.

**Technical overview.** Our construction of CUWE is inspired by Waters's functional encryption (FE) construction [23], which utilizes indistinguishable obfuscation ($i\mathcal{O}$) [24] and punctured programming. The key components of Waters' FE framework include a primitive known as puncturable deterministic encryption (PDE), which is a deterministic symmetric encryption scheme consisting of four PPT algorithms: $\mathsf{PDE.Setup}(1^\lambda)$, $\mathsf{PDE.Encrypt}(k, m)$, $\mathsf{PDE.Decrypt}(k, ct)$, and $\mathsf{PDE.Puncture}(k, m_0, m_1)$. The ciphertext of PDE takes the following form:

$$\mathrm{ct}_{\mathsf{PDE}} = (\mathrm{ct}_1 = \mathsf{PRF.F}_1(k_1, m), \mathrm{ct}_2 = \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1) \oplus m), \tag{1}$$

where $\mathsf{PRF.F}_1$ and $\mathsf{PRF.F}_2$ represent puncturable pseudorandom functions (PRFs). The PDE.Decrypt algorithm requires initially computing $m' = \mathrm{ct}_2 \oplus \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1)$ and then verifying that $\mathsf{PRF.F}_1(k_1, m') = \mathrm{ct}_1$. The PDE.Puncture algorithm takes as input a key $k$ and two messages $(m_0, m_1)$, producing a punctured key $k\{m_0, m_1\}$ which can decrypt all ciphertexts excluding those with encrypted $m_0$ or $m_1$.

Next, we first build a semi-adaptive security WE scheme from PDE. Subsequently, we extend this to a tag-based WE with fine-grained access control, ultimately evolving it into witness encryption with updatable ciphertexts. The semi-adaptive security witness encryption constructed from the PDE construct is described as follows.

The WE.Setup algorithm first generates a puncturable PRF key $k_p$, then constructs a program $P_e$ and obfuscates it as the encryption parameter $\mathrm{pp}_e := i\mathcal{O}_{P_e}$. The program $P_e$ takes a random input $r$, calculates $r_g = \mathsf{PRG.G}(r)$, generates a PDE key as $k_p = \mathsf{PRF.F}(k_p, r_g)$, and then outputs the tuple $(r_g, k_p)$. Then, the WE.Encrypt algorithm selects a random value $r$ to encrypt the message $m$, runs the obfuscation program $P_e$ on $r$ to generate $(r_g, k_p)$, and finally produces the ciphertext as $\mathrm{ct}_{\mathsf{WE}} := (x, \mathrm{ct}_{\mathsf{PDE}} = \mathsf{PDE.Encrypt}(k_p, (x \mid\mid m)), r_g)$. The decryption parameters $\mathrm{pp}_d$ are generated by obfuscating

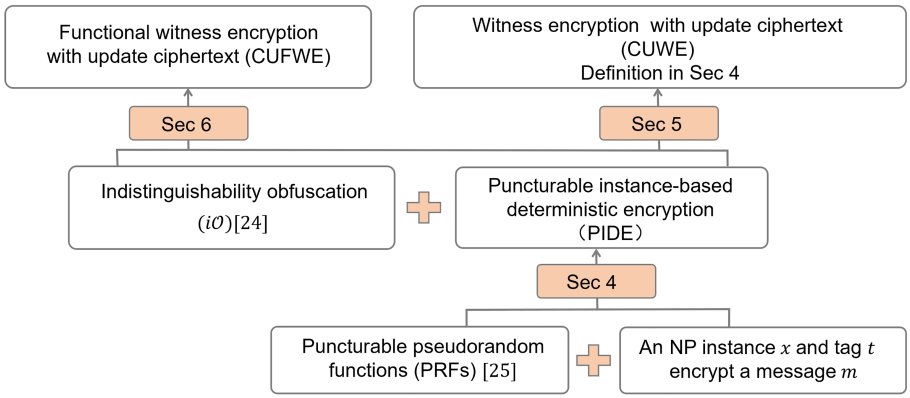


**Figure 2** (Color online) Landscape of our contributions.

the program $P_d$, which accepts as input the ciphertext $\mathrm{ct}_{\mathsf{WE}} := (x, \mathrm{ct}_{\mathsf{PDE}} = \mathsf{PDE.Encrypt}(k_p, (x \,||\, m)), r_g)$, obtains the key $k_p$ via $r_g$, decrypts $\mathrm{ct}_{\mathsf{PDE}}$ using $k_p$ to derive the message $x' \,||\, m'$, and outputs $m'$ when $x' = x$ as well $R(x, w) = 1$, otherwise outputs $\perp$. Therefore, the $\mathsf{WE.Decrypt}$ algorithm simply involves executing $i\mathcal{O}_{P_d}$ on the ciphertext $\mathrm{ct}_{\mathsf{WE}}$ and witness $w$.

The witness encryption described above is similar to the offline $\mathsf{WE}$ proposed by Abusalah et al. [11]. Next, we examine tag-based witness encryption with fine-grained access control. Since our goal is to study CUWE, the tag in the ciphertext can be updated only once. Thus, we cannot directly include $\mathsf{NP}$ instance $x$ and tag $t$ in the encrypted message as in the previous $\mathsf{WE}$ scheme. To add tags in ciphertext, the initial step involves modifying $\mathsf{PDE}$ into an instance-based variant known as $\mathsf{PIDE}$.

$\mathsf{PIDE}$ is similar to $\mathsf{PDE}$, except that the ciphertext is related with an $\mathsf{NP}$ instance $x$ and a tag $t$, and the puncturing process applies not only to a pair of messages $m_0$ and $m_1$ but also to an $\mathsf{NP}$ instance $x$ and a tag $t$. Therefore, the puncturing key $k_p\{t, x, m_0, m_1\}$ can decrypt all ciphertexts excluding $m_0$ or $m_1$ encrypted under $\mathsf{NP}$ instance $x$ and tag $t$. $\mathsf{PDE}$ ciphertexts take on the following form:

$$ct = (\mathrm{ct}_1 = \mathsf{PRF.F}_1(k_1, m) \oplus t, \mathrm{ct}_2 = \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1 \oplus x) \oplus m), \tag{2}$$

where $\mathsf{PRF.F}_1$ and $\mathsf{PRF.F}_2$ represent $\mathsf{PRF}$s. We replace $\mathsf{PDE}$ in the above $\mathsf{WE}$ scheme with our proposed $\mathsf{PIDE}$. The decryptor requires a valid witness $w$ and a tag $t_d$ to decrypt the ciphertext $\mathrm{ct}_t := (x, t, \mathrm{ct} = \mathsf{PDE.Encrypt}(k_p, t, x, m), r_g)$.

The $\mathsf{Decrypt}$ algorithm first checks whether witness $w$ satisfies $R(x, w) = 1$, then checks whether $\Gamma(t) = 1$ and $t_d \subseteq t$. If these conditions hold, it uses $r_g$ to obtain the key $k_p$, which is then used to decrypt $ct$ to retrieve the message $m$ and outputs $m$. If any condition is not met, the algorithm outputs $\perp$. It is important to note that the tag $t \in \{0,1\}^l$ is the $l$-bit public tag, and $\Gamma$ is the Boolean predicate. Decryption is only successful if $\Gamma(t) = 1$ and there exists a valid witness $w$ with $R(x, w) = 1$. In the decryption circuit $P_d$, validation involves verifying whether tag $t$ and the hard-coded $\Gamma$ satisfy $\Gamma(t) = 1$ and that the witness $w$ satisfies $R(x, w) = 1$.

Next, the most challenging aspect is updating the ciphertext. To ensure that an updated ciphertext cannot be updated again, we employ two distinct $\mathsf{PRF}$ keys $k_o$, $k_u$ as the master private key msk. Here, $k_o$ decrypts the original ciphertext, and $k_u$ decrypts the updated ciphertext. To update a ciphertext, we need to update the ciphertext encrypted within the $\mathsf{NP}$ instance $x$, tag $t$, and key $k_p$ (derived from $k_o$) to a new ciphertext under the $\mathsf{NP}$ instance $x$, tag $t' \in \mathcal{T}\backslash\{t\}$, and key $k'_p$ (derived from $k_u$). To accomplish this, we create a program $P_u$ that accepts a ciphertext $\mathrm{ct}_t := (x, \mathrm{ct} = \mathsf{PIDE.Encrypt}(k_p, t, x, m), r_g)$ and a random value $r$, decrypts the ciphertext $\mathrm{ct}_t$ to retrieve message $m$, subsequently re-encrypts $m$ with update key $k'_p$ and a tag $t' \in \mathcal{T}\backslash\{t\}$ to generate a new ciphertext $\mathrm{ct}_{t'}$. In addition to the determinism of the primitives employed (e.g., $\mathsf{PRF}$and $\mathsf{PIDE}$), we rely on (ordinary) $i\mathcal{O}$ for the update operation without requiring probabilistic $i\mathcal{O}$ [25]. It is worth noting that in our $\mathsf{CUWE}$ scheme, $i\mathcal{O}$ can hide the key of $\mathsf{PIDE}$.

**Our contributions.** The main contributions of this paper (as shown in Figure 2) are outlined below.

• **New cryptographic primitive.** We introduce a novel cryptographic primitive, $\mathsf{CUWE}$, and formally define it in Subsection 5.1. Within $\mathsf{CUWE}$, ciphertexts are identified with tags and an $\mathsf{NP}$ instance, ensuring that the decryption is successful only when the tag in the decryption parameters matches the tag of the ciphertext and there is a valid witness $w$. Our $\mathsf{CUWE}$ scheme can utilize the update token to

change tag $t$ of a ciphertext to any other tag, enabling fine-grained control when a ciphertext has been created.

• **Construction of CUWE from PIDE.** We propose a general construction of CUWE (Subsection 5.2) that offers IND-CUWE-CPA security, relying on our PIDE and an indistinguishability obfuscation ($i\mathcal{O}$). Furthermore, we follow our CUWE scheme and extend it to ciphertext-updatable function witness encryption (CUFWE) (Section 6), which only works only when the tag included in the decryption parameters and ciphertext match, and a valid witness $w$ is provided. Decryption outputs $f(m, w)$, thereby enabling enhanced data access control.

• **Construction of PIDE.** To construct our CUWE, we propose a PIDE primitive (Section 4) and demonstrate that it can be constructed from a puncturable PRF. It encrypts a message $m$ over an NP instance $x$ and a tag $t$. We show that our PIDE scheme is (selectively) indistinguishably secure if PRFs exist.

## 2 Related work

**Witness encryption.** The concept of witness encryption was first introduced at STOC'13 by Garg et al. [1], who proposed a candidate construct of WE under the exact cover problem (NP-complete problem). Garg et al. [24] demonstrated that $i\mathcal{O}$ implies witness encryption. Subsequently, Garg et al. [26] have shown that extractable WE for all languages does not exist in NP assuming special-purpose confusion.

While we are not aware of any existing witness encryption schemes designed to achieve ciphertext updatability, related concepts include verifiable WE based on threshold signatures (VweTS) [7]. In VweTS, a payer can encrypt the signature of a payment, which can only be decrypted by the receiver when the total amount of signers who signed another message reaches a threshold. The functionality of VweTS resembles multi-authority attribute-based encryption [27], where a user can encrypt a message that can only be decrypted if the decryptor possesses attributes from the authorities that satisfy the required threshold.

Another related work (through conceptual difference) is to update policies within ABE [28]. Their work combines the ciphertext policy ABE with the proxy re-encryption [21, 22], allowing to update the policies related with ciphertexts. Nevertheless, the above studies do not address WE schemes and are not well-suited for the applications we envision. Our research can be considered a hybrid of IBE/ABE and WE, with the additional advantage of updatability.

**Functional witness encryption.** Boyle et al. [29] initially proposed the concept of function-witness encryption (FWE). FWE allows the encryption of a message $(f, m)$ using an NP instance $x$ and a function $f$, such that anyone who knows a witness $w$ of $x \in \mathcal{L}$ can compute $f(m, w)$. Importantly, a party with a witness $w$ of $x$ does not learn the message $m$ itself but only the value of function $f(m, w)$. Subsequently, Abusalah et al. [11] presented offline functional witness encryption (OFWE) schemes, which offload the computationally expensive encryption steps to an offline setup phase. Building on this, other similar studies [30, 31] inspired by [11, 32] proposed OFWE schemes with selective security or semi-adaptive security for all NP languages.

A similar primitive to FWE is FE [33, 34], which allows fine-grained access control to encrypted data. Specifically, each decryption key $\mathrm{sk}_f$ is connected to a function $f$, and when $\mathrm{sk}_f$ is used to decrypt a ciphertext of $m$, the decryption yields only $f(m)$. Recently, function encryption with updatable ciphertext (CUFE) proposed by Cini et al. [34] has been closely related to our CUFWE. In CUFE, both a function key $\mathrm{sk}_{t,f}$ and a ciphertext $\mathrm{ct}_t$ are tagged. Decryption is only successful when the tag of the decryption key $\mathrm{sk}_{t,f}$ matches the tag of the ciphertext $\mathrm{ct}_t$, resulting in the output $f(m)$. In addition, CUFE allows ciphertexts to switch their tags to any other tag by updating the token. The concept is very similar to our CUFWE, where the encryption algorithm takes as input a tag $t$, an NP instance $x$, and a message $(f, m)$ to produce a ciphertext $\mathrm{ct}_t$. For security, it is required that the ciphertext computations of $(t, x, (f_0, m_0))$ and $(t, x, (f_1, m_1))$ are indistinguishable, where $f_0(m_0, w) = f_1(m_1, w)$ and $w$ satisfies $R(x, w) = 1$.

However, the key difference between CUFE and our CUFWE lies in their reliance on trust and communication channels. CUFE requires secure key transmission between users and the authority, whereas our approach does not necessitate a secret channel. Furthermore, CUFE requires the authority to be always online, which could pose full trust challenges.

## 3 Preliminaries

In this section, we present the definitions of the techniques necessary to construct our CUWE, including the pseudorandom generator [35], PRFs [36], and indistinguishable obfuscation [24].

### 3.1 Pseudorandom generator

**Definition 1** (Pseudorandom generator)**.** A pseudorandom generator is a function PRG: $\{0,1\}^n \to \{0,1\}^{m(n)}$, which is computational by an uniforms PPT machine, for all PPT adversary $\mathcal{A}$, there exists a negligible function negl, for all $n \in \mathbb{N}$, we have

$$\left| \Pr_{r \leftarrow \{0,1\}^n} [\mathcal{A}(\mathsf{PRG.G}(r)) = 1] - \Pr_{z \leftarrow \{0,1\}^m} [\mathcal{A}(z) = 1] \right| = \mathsf{negl}(\lambda). \tag{3}$$

### 3.2 Puncturable pseudorandom functions

**Definition 2** (Puncturable pseudorandom functions)**.** A puncturable family of pseudorandom functions PRF is defined by a tuple of three algorithms (PRF.G, PRF.F, PRF.Puncture) and a couple of computable functions $n = n(\lambda)$ and $m = m(\lambda)$, having the following properties.

• **Functionality preserved under puncturing.** For any PPT adversary $\mathcal{A}$ to produce a set $S \subseteq \{0,1\}^n$, and that any $y \in \{0,1\}^n$ in which $y \notin S$, the following holds:

$$\Pr\left[\mathsf{PRF.F}(k,y) = \mathsf{PRF.F}(k\{S\},y) \mid k \leftarrow \mathsf{PRF.G}(1^\lambda), k\{S\} \leftarrow \mathsf{PRF.Puncture}(k,S)\right] = 1. \tag{4}$$

• **Pseudorandom at punctured points.** For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, when $\mathcal{A}_1$ produces a set $S \subseteq \{0,1\}^n$ and a value $\nu$, take the experiment where $k \leftarrow \mathsf{PRF.G}(1^\lambda)$ and $k\{S\} \leftarrow \mathsf{PRF.Puncture}(k,S)$, the following holds:

$$\left|\Pr\left[\mathcal{A}_2(\nu, k\{S\}, S, \mathsf{PRF.F}(k,S)) = 1\right] - \Pr\left[\mathcal{A}_2(\nu, k\{S\}, S, D_{m\cdot|S|}) = 1\right]\right| = \mathsf{negl}(\lambda), \tag{5}$$

where $\mathsf{PRF.F}(k,S)$ indicates a combination of $\mathsf{PRF.F}(k,y_1),\ldots,\mathsf{PRF.F}(k,y_k)$, in which $S = \{y_1,\ldots,y_k\}$ is the enumeration of each element of $S$ in lexicographic orders, and $D_l$ represents a consistent distribution of $l$ bits.

### 3.3 Indistinguishability obfuscation

**Definition 3** (Indistinguishability obfuscation)**.** A PPT algorithm $i\mathcal{O}$ is an indistinguishable obfuscator on the class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, which having the following properties.

• For any $\lambda \in \mathbb{N}$, for any $\mathcal{C} \in \mathcal{C}_\lambda$, for any $y \in \{0,1\}^\lambda$, the following holds:

$$\Pr\left[C'(y) = C(y) : C' \leftarrow i\mathcal{O}(\lambda, C)\right] = 1. \tag{6}$$

• For any PPT distinguisher $\mathcal{D}$, any security parameters $\lambda \in \mathbb{N}$, and any pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, which for any input $y$, $C_0(y) = C_1(y)$ and $|C_0| = |C_1|$, the following holds:

$$\left|\Pr\left[\mathcal{D}\left(i\mathcal{O}\left(1^\lambda, C_0\right)\right) = 1\right] - \Pr\left[\mathcal{D}\left(i\mathcal{O}\left(1^\lambda, C_1\right)\right) = 1\right]\right| \leqslant \mathsf{negl}(\lambda). \tag{7}$$

## 4 Puncturable instance-based deterministic encryption

Our updatable ciphertext witness encryption scheme depends on the primitive of PIDE. In the following, we provide a formal definition of PIDE and demonstrate how to construct it from puncturable PRFs.

### 4.1 Definition of puncturable instance-based deterministic encryption

**Definition 4** (Puncturable instance-based deterministic encryption)**.** A PIDE scheme for NP instance $x \in \mathcal{L}$ is a tuple of four PPT algorithms (PIDE.Setup, PIDE.Encrypt, PIDE.Decrypt, PIDE.Puncture) defined as follows.

• $k_p \leftarrow \mathsf{PIDE.Setup}(1^\lambda)$ : on input a security parameter $1^\lambda$, PIDE.Setup outputs a key $k_p$.

• $\mathrm{ct} \leftarrow \mathsf{PIDE.Encrypt}(k_p, t, x, m)$ : on input a key $k_p$, a tag $t$, an instance $x \in \mathcal{L}$ and a message $m \in \mathcal{M}$, PIDE.Encrypt outputs a ciphertext ct.

• $m \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$ : on input a key $k_p$, a tag $t$, an instance $x \in \mathcal{L}$ and a ciphertext ct, $\mathsf{PIDE.Decrypt}$ outputs $m \in \mathcal{M} \cup \{\perp\}$.

• $k'_p \leftarrow \mathsf{PIDE.Puncture}(k_p, t, x, m_0, m_1)$ : on input a key $k_p$, a tag $t \in \mathcal{T}$, an instance $x \in \mathcal{L}$, messages $m_0, m_1 \in \mathcal{M}$, $\mathsf{PIDE.Puncture}$ outputs a new key $k'_p := k_p\{t, x, m_0, m_1\}$, where $k'_p$ denotes the puncturable key at the points $t, x, m_0, m_1$.

**Definition 5** (Correctness)**.** A PIDE scheme ($\mathsf{PIDE.Setup}$, $\mathsf{PIDE.Encrypt}$, $\mathsf{PIDE.Decrypt}$, $\mathsf{PIDE.Puncture}$) is correct, if for any $\lambda \in \mathbb{N}$, for any $t \in \mathcal{T}$, for any $x \in \mathcal{L}$, for any messages $m_0, m_1 \in \mathcal{M}$, for any $k_p \leftarrow \mathsf{PIDE.Setup}(1^\lambda)$, for any $k'_p \leftarrow \mathsf{PIDE.Puncture}(1^\lambda)$, we have

$$\Pr[\mathsf{PIDE.Decrypt}(k'_p, t, x, \mathsf{PIDE.Encrypt}(k_p, t, x, m)) \neq m] = \mathsf{negl}(\lambda). \tag{8}$$

In addition, for any $m \in \mathcal{M}$ (including $m_0, m_1$), we have

$$\Pr[\mathsf{PIDE.Decrypt}(k_p, t, x, \mathsf{PIDE.Encrypt}(k_p, t, x, m)) \neq m] = \mathsf{negl}(\lambda). \tag{9}$$

**Definition 6** (Security of PIDE)**.** A PIDE scheme is (selective) indistinguishability secure, if for any PPT adversaries $\mathcal{A}$ it holds that

$$\mathrm{Adv}_{\mathsf{PIDE},\mathcal{A}}(\lambda) := \left| \Pr \left[ \begin{array}{c} k_p \leftarrow \mathsf{PIDE.Setup}(1^\lambda), \\ (t, x, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda), \\ k_p\{t, x, m_0, m_1\} \leftarrow \mathsf{PIDE.Puncture}(k_p, t, x, m_0, m_1) \\ b \leftarrow \{0,1\} \\ \mathrm{ct}_0 \leftarrow \mathsf{PIDE.Encrypt}(k_p, t, x, m_b), \mathrm{ct}_1 \leftarrow \mathsf{PIDE.Encrypt}(k_p, t, x, m_{1-b}) \\ b^* \leftarrow \mathcal{A}(k_p\{t, x, m_0, m_1\}, \mathrm{ct}_0, \mathrm{ct}_1) : \\ b = b^* \end{array} \right] - \frac{1}{2} \right| \tag{10}$$

is negligible.

### 4.2 Construction of PIDE from puncturable PRFs.

Next, we present our PIDE construction, which can be observed as a variant of the work of Waters [23], where we additionally incorporate an NP instance $x$ and a tag $t$. Our PIDE scheme exploits two families of PRFs. The first is an injectable PRF $F_1$, which accepts an input of length $\lambda$ bytes and outputs a string of length $l = l(\lambda)$. The second is $F_2 : \{0,1\}^l \to \{0,1\}^\lambda$.

• $\mathsf{PIDE.Setup}(1^\lambda)$ :

– Generate two random PRF keys $k_1 \leftarrow \mathsf{PRF.G}_1(1^\lambda)$ and $k_2 \leftarrow \mathsf{PRF.G}_2(1^\lambda)$.

– Output the key $k_p := (k_1, k_2)$.

• $\mathsf{PIDE.Encrypt}(k_p := (k_1, k_2), t, x, m)$ :

– Compute $\mathrm{ct}_1 := \mathsf{PRF.F}_1(k_1, m) \oplus t$.

– Compute $\mathrm{ct}_2 := \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1 \oplus x) \oplus m$.

– Output $\mathrm{ct} = (\mathrm{ct}_1, \mathrm{ct}_2)$ as the ciphertext.

• $\mathsf{PIDE.Decrypt}(k_p := (k_1, k_2), t, x, \mathrm{ct} := (\mathrm{ct}_1, \mathrm{ct}_2))$ :

– Compute $m' := \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1 \oplus x) \oplus \mathrm{ct}_2$.

– Check $\mathsf{PRF.F}_1(k_1, m') \oplus t = \mathrm{ct}_1$; if so, output $m'$, otherwise output $\perp$.

• $\mathsf{PIDE.Puncture}(k_p := (k_1, k_2), t, x, m_0, m_1)$ :

– Calculate $d_{\mathrm{ct}} := \mathsf{PRF.F}_1(k_1, m_0) \oplus t$ and $e_{\mathrm{ct}} := \mathsf{PRF.F}_1(k_1, m_1) \oplus t$.

– Calculate $k_1\{m_0, m_1\} \leftarrow \mathsf{PRF.Puncture}_1(k_1, \{m_0, m_1\})$ and $k_2\{t, x\} \leftarrow \mathsf{PRF.Puncture}_2(k_2, \{d_{\mathrm{ct}} \oplus x, e_{\mathrm{ct}} \oplus x\})$.

– Output $k_p\{t, x, m_0, m_1\} := (k_1\{m_0, m_1\}, k_2\{t, x\})$.

**Correctness.** Correctness follows straightforwardly from the correctness of the PRFs. First, the non-punctured keys are correct by observation. For the punctured key, correctness follows for key $k_p\{t, x, m_0, m_1\}$ on all messages $m \neq m_0, m_1$. Specifically, $\mathsf{PRF.F}_1(k_1, m) \oplus t \neq \mathsf{PRF.F}_1(k_1, m_0) \oplus t$ or $\mathsf{PRF.F}_1(k_1, m_1) \oplus t$, which holds because $\mathsf{PRF.F}_1$ is injective.

**Theorem 1.** If PIDE is instantiated with two secure puncturable pseudorandom functions $\mathsf{F}_1$ and $\mathsf{F}_2$, then the PIDE scheme is (selectively) indistinguishably secure.

*Proof.* We proved the theorem via a series of hybrid games. Below we show that each succeeding hybrid game is computationally indistinguishable from the former game. Following that, let $\mathsf{Hybrid}_i \approx \mathsf{Hybrid}_{i+1}$ indicate $|\Pr[\mathsf{Hybrid}_i = 1] - \Pr[\mathsf{Hybrid}_{i+1} = 1]| \leqslant \mathsf{negl}(\lambda)$.

• $\mathsf{Hybrid}_0$ : It is defined as the original selective indistinguishability game of PIDE.

• $\mathsf{Hybrid}_1$ : In this hybrid $\mathrm{ct}_1^b, \mathrm{ct}_1^{1-b} \in \{0,1\}^l$ gets sampled uniformly, rather than being calculated as $\mathrm{ct}_1^b := \mathsf{PRF.F}_1(k_1, m_b) \oplus t$ and $\mathrm{ct}_1^{1-b} := \mathsf{PRF.F}_1(k_1, m_{1-b}) \oplus t$. The other part of that hybrid is the same as the former hybrid, $\mathsf{Hybrid}_0$.

• $\mathsf{Hybrid}_2$ : In this hybrid $\mathrm{ct}_2^b, \mathrm{ct}_2^{1-b} \in \{0,1\}^l$ gets sampled uniformly, rather than being calculated as $\mathrm{ct}_2^b := \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1^b) \oplus x$ and $\mathrm{ct}_2^{1-b} := \mathsf{PRF.F}_2(k_2, \mathrm{ct}_1^{1-b}) \oplus x$. The other part of that hybrid is the same as the former hybrid, $\mathsf{Hybrid}_1$.

**Claim 1.** Assuming $\mathsf{F}_1$ is a secure puncturable PRF, we have $\mathsf{Hybrid}_0 \approx \mathsf{Hybrid}_1$.

*Proof.* If there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hybrid}_0$ from $\mathsf{Hybrid}_1$, then a PPT reduction algorithm $\mathcal{B}$ that is capable of breaking the PRF security game can be constructed. The reduction for the security of PRF is as follows: First, $\mathcal{B}$ receives a PRF game challenge $(t, x, m_0, m_1)$. Next, it executes the adversary $\mathcal{A}$ while engaging with the PIDE security game (as explained in $\mathsf{Hybrid}_0$). $\mathcal{B}$ obtains a punctured PRF key $k_{\mathrm{prf}}\{m_b, m_{1-b}\}$ and challenge values $v_0, v_1$. Then, it sets $k_p := (k_{\mathrm{prf}}\{m_b, m_{1-b}\}, k_2)$, $\mathrm{ct}_0(v_b, \mathsf{PRF.F}_2(k_2, v_b \oplus x) \oplus m_b)$, and $\mathrm{ct}_1(v_{1-b}, \mathsf{PRF.F}_2(k_2, v_{1-b} \oplus x) \oplus m_{1-b})$ and returns $(k_p, \mathrm{ct}_0, \mathrm{ct}_1)$ to $\mathcal{A}$. If the adversary $\mathcal{A}$ wins, then $\mathcal{B}$ outputs 1 indicating that $v_0 := \mathsf{PRF.F}_1(k_1, m_0) \oplus t$, $v_1 := \mathsf{PRF.F}_1(k_1, m_1) \oplus t$; otherwise, it returns 0 to demonstrate $v_0, v_1$ were selected randomly.

Now, we analyze these two cases in sequence. If the PRF challenger generates $v_0 := \mathsf{PRF.F}_1(k_1, m_0) \oplus t$ and $v_1 := \mathsf{PRF.F}_1(k_1, m_1) \oplus t$, then $\mathcal{B}$ provides an exact view of $\mathsf{Hybrid}_0$ to $\mathcal{A}$. Otherwise, if $v_0, v_1$ are selected randomly, the view corresponds to $\mathsf{Hybrid}_1$. To summarize, if adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_0$ from $\mathsf{Hybrid}_1$ with non-negligible probability, then the reduction algorithm $\mathcal{B}$ compromises the security of PRFs with non-negligible probability.

**Claim 2.** Assuming $\mathsf{F}_2$ is a secure puncturable PRF, we have $\mathsf{Hybrid}_1 \approx \mathsf{Hybrid}_2$.

*Proof.* The proof is analogous to the proof of Claim 1.

## 5 Witness encryption with updatable ciphertexts

In this section, we first define the main object of our study, CUWE. Then, we present a construction of CUWE from PIDE. Finally, to better support fine-grained access control, we demonstrate how to derive a CUFWE scheme from CUWE.

### 5.1 Definition of witness encryption with updatable ciphertexts

**Definition 7** (Witness encryption with updatable ciphertexts)**.** A CUWE scheme based on a tag $t \in \mathcal{T}$ for an NP language $\mathcal{L}$ with witness relation $R$ consists of five PPT algorithms (Setup, Encrypt, Decrypt, TokGen, Update), defined as follows.

• $(\mathrm{pp}_e, \mathrm{pp}_d, \mathrm{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \Gamma)$ : on input a security parameter $1^\lambda$ and a predicate $\Gamma$, Setup outputs a encryption parameter $\mathrm{pp}_e$, a decryption parameter $\mathrm{pp}_d$, and a master secret key msk.

• $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(\mathrm{msk}, t, t')$ : on input a master secret key msk, a tags $t, t' \in \mathcal{T}$, TokGen outputs an update token $\triangle_{t\to t'}$.

• $\mathrm{ct}_t \leftarrow \mathsf{Encrypt}(\mathrm{pp}_e, t, x, m)$ on input encryption parameter $\mathrm{pp}_e$, a tag $t$, an instance $x \in \mathcal{L}$, and a message $m \in \mathcal{M}$, Encrypt outputs a ciphertext $\mathrm{ct}_t$.

• $\mathrm{ct}_{t'} \leftarrow \mathsf{Update}(\triangle_{t\to t'}, \mathrm{ct}_t)$ : on input an update token $\triangle_{t\to t'}$, a original ciphertext $\mathrm{ct}_t$, Update outputs an update ciphertext $\mathrm{ct}_{t'}$.

• $m \leftarrow \mathsf{Decrypt}(\mathrm{pp}_d, t_d, w, \mathrm{ct}_t/\mathrm{ct}_{t'})$ : on input decryption parameter $\mathrm{pp}_d$, a tag $t_d$, a witness $w \in \mathcal{W}$, and a ciphertext $\mathrm{ct}_t$ (either a nonupdated one $\mathrm{ct}_t$ or an updated one $\mathrm{ct}_{t'}$), Decrypt outputs $m$ if $\Gamma(t) = 1$ or $\Gamma(t') = 1$, else outputs $\perp$.

Next, we focus on the fundamental definition of correctness. Essentially, the correctness guarantees that if the tag of the decryption parameter $\mathrm{pp}_d$ matches the tag in the (updated) ciphertext, and the witness $w$ associated with the NP instance $x$ when $R(x, w) = 1$, then decryption succeeds.

**Definition 8** (Correctness)**.** A CUWE scheme (Setup, Encrypt, Decrypt, TokGen, Update) is correct, if for any $\lambda \in \mathbb{N}$, for any $(\mathrm{pp}_e, \mathrm{pp}_d, \mathrm{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \Gamma)$, for any $t \in \mathcal{T}$, for any $m \in \mathcal{M}$, for any $x \in \mathcal{L}$, and

$\underline{\mathsf{Exp}^{\mathsf{IND-CUWE-CPA}}_{\mathsf{CUWE},\mathcal{A}}(\lambda)}$:

1. $(pp_e, pp_d, msk) \leftarrow \mathsf{Setup}(1^\lambda, \Gamma)$
2. $\mathcal{C} := \mathcal{C}_u := \emptyset, \mathcal{T}_h := \mathcal{T}_c := \emptyset, ct_o := ct_u := 1, t_h := t_c := 1$
3. $(t^*, x, m_0^*, m_1^*, \mathrm{st}) \leftarrow \mathcal{A}(1^\lambda, pp_e, R)$
4. $b \leftarrow \{0,1\},\ ct_t^* \leftarrow \mathsf{Encrypt}(pp_e, t^*, x, m_b^*)$
5. $\mathcal{C} := \mathcal{C} \cup \{0, ct_t^*, t^*\}$
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}}(ct_t^*, st)$
7. If $x \in \mathcal{L}$, return 0
8. If $b' = b$, then return 1 else return 0

Oracles $\mathcal{O}$ :

- $\mathsf{TokGen}_h(t, t')$ : If $t' \notin \mathcal{T},\ t \notin \mathcal{T}$, or $(\cdot, t, t') \in \mathcal{T}_c$, then return $\perp$. Run $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$ and set $\mathcal{T}_h := \mathcal{T}_h \cup \{(t_h, t, t', \triangle_{t\to t'})\},\ t_h := t_h + 1$.
- $\mathsf{TokGen}_c(t, t')$ : If $t' \notin \mathcal{T},\ t \notin \mathcal{T}$, or $(\cdot, t, t') \in \mathcal{T}_h$, then return $\perp$. Run $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$ and set $\mathcal{T}_c := \mathcal{T}_c \cup \{(t_c, t, t')\},\ t_c := t_c + 1$, and return $\triangle_{t\to t'}$.
- $\mathsf{Encrypt}^*(t, x, m)$ : Compute $ct_t \leftarrow \mathsf{Encrypt}(pp_e, t, x, m)$, set $\mathcal{C} := \mathcal{C} \cup \{(ct_o, ct_t, t)\}$, $ct_o := ct_o + 1$ and return $ct_t$.
- $\mathsf{Update}_h(t, t', i, j)$ : If $(i, \cdot, t) \notin \mathcal{T}_c$ or $(\cdot, t, t') \in \mathcal{T}_c$ , then return $\perp$. If $(j, t, t', \cdot) \notin \mathcal{T}_h$, compute $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(msk, t, t')$ and set $\mathcal{T}_h := \mathcal{T}_h \cup \{(t_h, t, t', \triangle_{t\to t'})\},\ t_h := t_h + 1$; otherwise, retrieve $(j, t, t', \triangle_{t\to t'})$ from $\mathcal{T}_h$. Retrieve $(i, ct_t, t)$ from $\mathcal{C}$ and compute $ct_{t'} \leftarrow \mathsf{Update}(\triangle_{t\to t'}, ct_t)$. Set $\mathcal{C}_u := \mathcal{C}_u \cup \{(ct_u, i, t')\}$, $ct_u := ct_u + 1$, and return $ct_{t'}$ .

**Figure 3** IND-CUWE-CPA-security game.

for any witnesses $w$ such that $R(x, w) = 1$, the following holds:

$$\Pr[\mathsf{Decrypt}(\mathrm{pp}_d, t_d, w, \mathsf{Encrypt}(\mathrm{pp}_e, t, x, m)) = m] = 1. \tag{11}$$

In addition, for all $\mathrm{ct}_t \leftarrow \mathsf{Encrypt}(\mathrm{pp}_e, t, x, m)$, for all $t' \in \mathcal{T}\backslash\{t\}$, for all $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(\mathrm{msk}, t, t')$, we have

$$\Pr[\mathsf{Decrypt}(\mathrm{pp}_d, t_d, w, \mathsf{Update}(\triangle_{t\to t'}, \mathrm{ct}_t)) = m] = 1. \tag{12}$$

We observe that owing to the updatability of ciphertexts, existing WE schemes lack the concept of update tokens. Therefore, it is necessary to provide additional features to ensure the security of CUWE. To address this, we define a concept of indistinguishability, referred to as IND-CUWE-CPA, which ensures that an adversary cannot distinguish between the ciphertext corresponding to a specific tag $t^*$ and the messages $(x, m_0^*)$ or $(x, m_1^*)$ chosen by the adversary.

We restrict the ability to update access policies (i.e., tag $t$) in the ciphertext using a token, limiting updates only from the original ciphertext to the updated ciphertext. To achieve this, we offer the adversary access to four oracles $(\mathsf{TokGen}_c, \mathsf{TokGen}_h, \mathsf{Encrypt}^*, \mathsf{Update}_h)$. Specifically, two of the oracles pertain to the generation of update tokens. Through $\mathsf{TokGen}_c$, the adversary is granted adaptive access to query corrupted tokens, while $\mathsf{TokGen}_h$ provides access to query honestly generated tokens. We also provide $\mathsf{Encrypt}^*$ and $\mathsf{Update}_h$ to ensure the security associated with the updatability of honestly produced ciphertexts. $\mathsf{Encrypt}^*$ allows the generation of honest ciphertexts, while $\mathsf{Update}_h$ facilitates the update of ciphertexts generated by $\mathsf{Encrypt}^*$, without exposing the update token to the adversary. Through the $\mathsf{TokGen}_h$ oracle, the adversary can request the generation of honest tokens, which the experiment can then leverage to perform honest updates on the ciphertexts.

We now present the security definition of our witness encryption with updatable ciphertexts.

**Definition 9** (Security)**.** A CUWE scheme is semi-adaptive IND-CUWE-CPA-secure, if for any PPT adversaries $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{IND\text{-}CUWE\text{-}CPA}}_{\mathsf{CUWE},\mathcal{A}}(\lambda)$ (Figure 3), it holds that

$$\mathsf{Adv}^{\mathsf{IND\text{-}CUWE\text{-}CPA}}_{\mathsf{CUWE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{IND\text{-}CUWE\text{-}CPA}}_{\mathsf{CUWE},\mathcal{A}}(\lambda) = 1\right] - 1/2\right| \tag{13}$$

is negligible, where $\mathsf{Exp}^{\mathsf{IND\text{-}CUWE\text{-}CPA}}_{\mathsf{CUWE},\mathcal{A}}(\lambda)$ is described in Figure 3.

### 5.2 Construction of witness encryption with updatable ciphertext from PIDE

Next, we describe our witness encryption with updatable ciphertext schemes from puncturable instance-based deterministic encryption that offers semi-adaptive IND-CUWE-CPA security. The construction of CUWE is described below.

- $\mathsf{Setup}(1^\lambda, \Gamma)$ :

– Generate two random PRF keys $k_o \leftarrow \mathsf{PRF.G}(1^\lambda)$ and $k_u \leftarrow \mathsf{PRF.G}(1^\lambda)$.

– Construct an obfuscation $P_e \leftarrow i\mathcal{O}_{P_e:1[k_o]}$ for the program $P_e : 1[k_o]$, where $P_e : 1[k_o]$ is padded to the size equal to max $|P_e : 1[k_o]|, |P_e : 2[k_o\{r_g^*\}]|$.

– Construct an obfuscation $P_d \leftarrow i\mathcal{O}_{P_d:1[k_o,k_u,\Gamma]}$ for the program $P_d : 1[k_o, k_u, \Gamma]$, where $P_d : 1[k_o, k_u, \Gamma]$ is padded to the size equal to max $|P_d : 1[k_o, k_u, \Gamma]|, |P_d : 2[k_o\{r_g^*\}, k_u, \Gamma, r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, k_p']|$.

– Output the encryption parameter $\mathrm{pp}_e := P_e$, the decryption parameter $\mathrm{pp}_d = P_d$, and the master secret key $\mathrm{msk} := (k_o, k_u)$.

• $\mathsf{TokGen}(\mathrm{msk} := (k_o, k_u), t, t')$ :

– Produce an obfuscated program $P_u \leftarrow i\mathcal{O}_{P_u:1[k_o,k_u,t,t']}$ for the program $P_u : 1[k_o, k_u, t, t']$, where $P_d : 1[k_o, k_u, \Gamma]$ is padded to the size equal to max $|P_u : 1[k_o, k_u, t, t']|, |P_u : 2[k_o\{r_g^*\}, k_u, t, t', r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}_0', \mathrm{ct}_1', k_p']|$.

– Output $\triangle_{t\to t'} := P_u$ as the update token.

• $\mathsf{Encrypt}(\mathrm{pp}_e := P_e, t, x, m)$ :

– Select a random $r \in \{0,1\}^\lambda$.

– Take $r$ as the input of obfuscation program $P_e : 1[k_o]$ and execute it to obtain $(r_g, k_p) \leftarrow P_e(r)$.

– Calculate $\mathrm{ct} \leftarrow \mathsf{PIDE.Encrypt}(k_p, t, x, m)$.

– Output $\mathrm{ct}_t = (x, t, \mathrm{ct}, r_g)$ as the ciphertext.

• $\mathsf{Update}(\triangle_{t\to t'} := P_u, \mathrm{ct}_t)$ :

– Select a random $r' \in \{0,1\}^\lambda$.

– Execute the obfuscated program $\mathrm{ct}_{t'} \leftarrow P_u(\mathrm{ct}_t, r')$.

– Output $\mathrm{ct}_{t'}$ as the update ciphertext.

• $\mathsf{Decrypt}(\mathrm{pp}_d := P_d, t_d, w, \mathrm{ct}_t)$ :

– Run the obfuscated program $m \leftarrow P_d(t_d, w, \mathrm{ct}_t)$.

– Output $m$ as the answer.

**Correctness.** Correctness follows straightforwardly from the correctness of the pseudorandom generator $\mathsf{PRG}$, puncturable instance-based deterministic encryption scheme $\mathsf{PIDE}$, indistinguishability obfuscation $i\mathcal{O}$, and puncturable pseudorandom function $\mathsf{PRF}$, along with the correctness of the program $P_e : 1[k_o], P_d : 1[k_o, k_u, \Gamma], P_u : 1[k_o, k_u, t, t']$.

$P_e : 1[k_o]$

**Constants:** $\mathsf{PRF}$ key $k_o$
**Input:** $r \in \{0,1\}^\lambda$

1. Generate $r_g = \mathsf{PRG.G}(r)$.
2. Let $k_p = \mathsf{PRF.F}(k_o, r_g)$.
3. Output $(r_g, k_p)$.

$P_d : 1[k_o, k_u, \Gamma]$

**Constants:** $\mathsf{PRF}$ keys $k_o,\ k_u$, Predicate $\Gamma$
**Input:** $t_d, w, \mathrm{ct}_t = (x, t, \mathrm{ct}, r_g)$

1. If $R(x, w) = 0$, or $t_d \not\subseteq t$, or $\Gamma(t) = 0$, output $\perp$.
2. If $\mathrm{ct}_t$ is updated ciphertext, set $k = k_u$, else set $k = k_o$.
3. Compute $k_p = \mathsf{F}(k, r_g)$.
4. Compute $m' \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$.
5. Output $m'$.

$P_u : 1[k_o, k_u, t, t']$

**Constants:** $\mathsf{PRF}$ keys $k_o, k_u$, tag $t, t' \in \mathcal{T}$
**Input:** $\mathrm{ct}_t = (x, t, \mathrm{ct}, r_g), r \in \{0,1\}^\lambda$

1. Compute $k_p = \mathsf{PRF.F}(k_o, r_g)$.
2. Compute $m \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$.
3. Generate $r_g' \leftarrow \mathsf{PRG.G}(r)$.
4. Compute $k_p' = \mathsf{PRF.F}(k_u, r_g')$.
5. Compute $\mathrm{ct}' \leftarrow \mathsf{PIDE.Encrypt}(k_{\mathrm{p}}', t', x, m)$.
6. Output $\mathrm{ct}_{t'} := (x, t', \mathrm{ct}', r_g')$.

**Theorem 2.** The $\mathsf{CUWE}$ scheme is semi-adaptively $\mathsf{IND\text{-}CUWE\text{-}CPA}$ secure if it is instantiated with a secure $\mathsf{PRG}$, a secure $\mathsf{PRF}$, an indistinguishability secure $\mathsf{PIDE}$, and an indistinguishability secure obfuscator $i\mathcal{O}$.

*Proof.* We proved the theorem via a series of hybrid games. Below we show that each succeeding hybrid game is computationally indistinguishable from the former game. Following that, let $\mathsf{Hybrid}_i \approx \mathsf{Hybrid}_{i+1}$ indicate $|\Pr[\mathsf{Hybrid}_i = 1] - \Pr[\mathsf{Hybrid}_{i+1} = 1]| \leqslant \mathsf{negl}(\lambda)$.

• $\mathsf{Hybrid}_0$ : It is defined as the original security experiment presented in Figure 3.

• $\mathsf{Hybrid}_1$ : In this hybrid $r_g^* \in \{0,1\}^{2\lambda}$ gets sampled uniformly, rather than being calculated as $r_g^* \leftarrow \mathsf{PRG.G}(r^*)$, where $r^* \in \{0,1\}^{\lambda}$.

• $\mathsf{Hybrid}_2$ : In this hybrid $P_e$ is computed as $i\mathcal{O}_{P_e:2[k_o\{r_g^*\}]}$, where $k_o\{r_g^*\} \leftarrow \mathsf{PRF.Puncture}(k_o, r_g^*)$, instead of being computed as $P_e \leftarrow i\mathcal{O}_{P_e:1[k_o]}$ for the program $P_e : 1[k_o]$.

• $\mathsf{Hybrid}_3$ : In this hybrid $P_d$ is computed as $i\mathcal{O}_{P_d:2[k_o\{r_g^*\},k_u,\Gamma,r_g^*,\mathrm{ct}_0,\mathrm{ct}_1,k'_p]}$, where $k_p^* \leftarrow \mathsf{PRF.F}(k_o, r_g^*)$, $k'_p \leftarrow \mathsf{PIDE.Puncture}(k_p^*, t^*, x, m_0^*, m_1^*)$, $\mathrm{ct}'_0 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_0^*)$, $\mathrm{ct}'_1 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_1^*)$, $|m_0^*| = |m_1^*|$, let $\mathrm{ct}_0, \mathrm{ct}_1$ consist of $\mathrm{ct}'_0, \mathrm{ct}'_1$ in lexicographic order[1)], instead of being computed as $P_d \leftarrow i\mathcal{O}_{P_d:1[k_o,k_u,\Gamma]}$.

• $\mathsf{Hybrid}_4$ : In this hybrid $P_u$ is computed as $i\mathcal{O}_{P_u:2[k_o\{r_g^*\},k_u,t,t',r_g^*,\mathrm{ct}_0,\mathrm{ct}_1,\mathrm{ct}'_0,\mathrm{ct}'_1,k'_p]}$, in which $|m_0^*| = |m_1^*|$, $k_p^* \leftarrow \mathsf{PRF.F}(k_o, r_g^*)$, $k'_p \leftarrow \mathsf{PIDE.Puncture}(k_p^*, t^*, x, m_0^*, m_1^*)$, $\mathrm{ct}_0 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_0^*)$, $\mathrm{ct}_1 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_1^*)$, $k''_p \leftarrow \mathsf{PRF.F}(k_u, r)$, randomly select $r \in \{0,1\}^{\lambda}$, $\mathrm{ct}'_0 \leftarrow \mathsf{PIDE.Encrypt}(k''_p, t, x, m_0^*)$, $\mathrm{ct}'_1 \leftarrow \mathsf{PIDE.Encrypt}(k''_p, t, x, m_1^*)$, let $\mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}'_0, \mathrm{ct}'_1$ be arranged in lexicographic orde[2)], instead of being computed as $P_u \leftarrow i\mathcal{O}_{P_u:1[k_o,k_u,t,t']}$.

• $\mathsf{Hybrid}_5$ : In this hybrid $k_p^*$ samples uniformly, rather than being calculated as $k_p^* \leftarrow \mathsf{PRF.F}(k_o, r_g^*)$.

• $\mathsf{Hybrid}_6$ : In this hybrid $\mathrm{ct}^*$ is computed as $\mathsf{PIDE.Encrypt}(k_p^*, t^*, x, m_0^*)$, rather than being computed as $\mathsf{PIDE.Encrypt}(k_p^*, t^*, x, m_b^*)$. The other part of that hybrid is the same as the former hybrid, $\mathsf{Hybrid}_5$.

**Claim 3.** Assuming PRG is a secure pseudorandom generator, we have $\mathsf{Hybrid}_0 \approx \mathsf{Hybrid}_1$.

*Proof.* If there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hybrid}_0$ from $\mathsf{Hybrid}_1$, then a PPT reduction algorithm $\mathcal{B}$ which is capable of breaking the PRG security game can be constructed. We can build this reduction for the security of PRG: First, $\mathcal{B}$ obtains a PRG challenge $r_g \in \{0,1\}^{2\lambda}$. Next, it executes the $\mathcal{A}$ and plays the CUWE challgenge game(as explained by $\mathsf{Hybrid}_0$), except that it sets $r_g^* := r_g$ when generating the challenge ciphertext. If the adversary $\mathcal{A}$ wins, $\mathcal{B}$ returns 1 indicating that $r_g$ corresponds to the image of PRG; otherwise, $\mathcal{B}$ returns 0 demonstrating that $r_g$ is selected arbitrarily.

Now we analyze these two cases in sequence. When the PRG challenger generates $r_g = \mathsf{PRG}(r)$, for some $r \in \{0,1\}^{\lambda}$, $\mathcal{B}$ presents exactly views of $\mathsf{Hybrid}_0$ to $\mathcal{A}$. If not, $r_g$ is selected randomly views of $\mathsf{Hybrid}_1$. In other words, if the adversary $\mathcal{A}$ is capable of distinguishing $\mathsf{Hybrid}_0$ from $\mathsf{Hybrid}_1$ by non-negligible probability, then our reduction algorithm $\mathcal{B}$ destroys the security of PRG by non-negligible probability.

**Claim 4.** Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, we have $\mathsf{Hybrid}_1 \approx \mathsf{Hybrid}_2$.

*Proof.* If there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hybrid}_1$ from $\mathsf{Hybrid}_2$, then a PPT reduction algorithm $\mathcal{B}$, which can break the $i\mathcal{O}$ security game, can be constructed. We can build this reduction for the security of $i\mathcal{O}$ as follows: First, $\mathcal{B}$ runs as in $\mathsf{Hybrid}_1$. Next, the reduction algorithm $\mathcal{B}$ calculates the punctured PRF key $k_o\{r_g^*\} \leftarrow \mathsf{PRF.Puncture}(k_o, r_g^*)$ and constructs two circuits defined as $C_0 = P_e : 1[k_o]$ and $C_1 = P_e : 2[k_o\{r_g^*\}]$. Next, $\mathcal{B}$ submits both circuits $C_0$, $C_1$ to the $i\mathcal{O}$ challenger and receives a returned program $P$, which it designates as $\mathrm{pp}_e := P$ and sends it to $\mathcal{A}$. The rest of the procedure follows the steps in $\mathsf{Hybrid}_1$. If the adversary $\mathcal{A}$ wins, $\mathcal{B}$ returns 0, indicating that $P$ is the obfuscation of $C_0$; otherwise, $\mathcal{B}$ returns 1, demonstrating that $P$ is the obfuscation of $C_1$.

Now, we analyze these two cases in sequence. When the $i\mathcal{O}$ challenger generates $P$ as the obfuscation of $C_1$, $\mathcal{B}$ provides an exact view of $\mathsf{Hybrid}_2$ to $\mathcal{A}$. Otherwise, if $P$ is the obfuscation of $C_0$, it provides a view of $\mathsf{Hybrid}_1$. Furthermore, the programs were functionally equivalent with negligible probability of difference, as the probability that $r_g^*$ lies beyond the image of the PRG is at least $1 - 2^{\lambda}$. In other words, if the adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_1$ from $\mathsf{Hybrid}_2$ with non-negligible probability, our reduction algorithm $\mathcal{B}$ compromises the security of $i\mathcal{O}$ with non-negligible probability.

$P_e : 2[k_o\{r_g^*\}]$

**Constants:** Punctured PRF key $k_o\{r_g^*\}$

**Input:** $r \in \{0,1\}^{\lambda}$

1. Generate $r_g = \mathsf{PRG.G}(r)$. 2. Let $k_p = \mathsf{PRF.F}(k_o\{r_g^*\}, r_g)$. 3. Output $(r_g, k_p)$.

**Claim 5.** Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, we have $\mathsf{Hybrid}_2 \approx \mathsf{Hybrid}_3$.

*Proof.* If there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hybrid}_2$ from $\mathsf{Hybrid}_3$, then a PPT reduction algorithm $\mathcal{B}$ which is capable of breaking the $i\mathcal{O}$ security game can be constructed. We can build this reduction for the security of $i\mathcal{O}$: First, $\mathcal{B}$ runs as in $\mathsf{Hybrid}_2$. Then the reduction algorithm $\mathcal{B}$ computes $k'_p \leftarrow$

1) If $\mathrm{ct}'_0 < \mathrm{ct}'_1$, then $\mathrm{ct}_0 = \mathrm{ct}'_0, \mathrm{ct}_1 = \mathrm{ct}'_1$, otherwise, $\mathrm{ct}_0 = \mathrm{ct}'_1, \mathrm{ct}_1 = \mathrm{ct}'_0$.

2) Here we sort $\mathrm{ct}_0$ and $\mathrm{ct}_1$ lexicographically and then we order $\mathrm{ct}'_0$ and $\mathrm{ct}'_1$ according to this sort.

$\mathsf{PIDE.Puncture}(k_p^*, t^*, x, m_0^*, m_1^*)$, for $k_p^* \leftarrow \mathsf{PRF.F}(k_o, r_g^*)$, calculates $\mathrm{ct}_0' \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_0^*)$, $\mathrm{ct}_1' \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_1^*)$ when $|m_0^*| = |m_1^*|$. Then let $\mathrm{ct}_0, \mathrm{ct}_1$ include $\mathrm{ct}_0', \mathrm{ct}_1'$ in lexicographic order. It creates two circuits as $C_0 = P_d : 1[k_o, k_u, \Gamma]$ and $C_1 = P_d : 2[k_o\{r_g^*\}, k_u, \Gamma, r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, k_p']$. Next, $\mathcal{B}$ submits both of the circuits $C_0,\ C_1$ to the $i\mathcal{O}$ challenger and obtains a program $P$. It lets $\mathrm{pp}_d := P$, and sends it to the $\mathcal{A}$. The other part of the procedure is the same as the $\mathsf{Hybrid}_2$. If the adversary $\mathcal{A}$ wins, $\mathcal{B}$ returns 0 indicating that $P$ is the obfuscation of $C_0$; if not, $\mathcal{B}$ returns 1 to demonstrate that $P$ is the obfuscation of $C_1$.

$P_d : 2[k_o\{r_g^*\}, k_u, \Gamma, r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, k_p']$

**Constants:** (Punctured) PRF key $k_o\{r_g^*\}, k_u$, predicate $\Gamma$, point $r_g^* \in \{0,1\}^{2\lambda}$, PIDE ciphertexts $\mathrm{ct}_0, \mathrm{ct}_1$, punctured PIDE key $k_p' = k_p\{t^*, x, m_0^*, m_1^*\}$

**Input:** $t_d, w, \mathrm{ct}_t = (x, t, \mathrm{ct}, r_g)$

1. If $R(x, w) = 0$, or $t_d \not\subseteq t$, or $\Gamma(t) = 0$, output $\perp$.
2. If $r_g = r_g^*$, and $\mathrm{ct} \neq \mathrm{ct}_0, \mathrm{ct}_1$, compute $(x', m') \leftarrow \mathsf{PIDE.Decrypt}(k_p', t, \mathrm{ct})$.
3. If $r_g = r_g^*$, $\mathrm{ct} = \mathrm{ct}_b$, set $x' := x,\ m' = m_b^*$.
4. If $\mathrm{ct}_t$ is updated ciphertext, set $k = k_u$, else set $k = k_o\{r_g^*\}$.
5. Otherwise compute $k_p = \mathsf{PRF.F}(k, r_g)$, $m' \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$.
6. Output $m'$.

Now, we analyze these two cases in sequence. When the $i\mathcal{O}$ challenger generates $P$ as the obfuscation of $C_1$, $\mathcal{B}$ provides an exact view of $\mathsf{Hybrid}_3$ to $\mathcal{A}$. Otherwise, if $P$ is the obfuscation of $C_0$, it provides a view corresponding to $\mathsf{Hybrid}_2$. Once again, the programs were nearly functionally equivalent, with differences appearing only in the hardwired responses for two inputs. Correctness holds across messages with very limited exceptions. Therefore, if the adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_2$ from $\mathsf{Hybrid}_3$ with non-negligible probability, then our reduction algorithm $\mathcal{B}$ destroys the security of $i\mathcal{O}$ by non-negligible probability.

**Claim 6.** Assuming $i\mathcal{O}$ is a secure indistinguishability obfuscator, we obtain $\mathsf{Hybrid}_3 \approx \mathsf{Hybrid}_4$.

*Proof.* To prove this claim, we consider a hybrid argument. Let $Q = Q(\lambda)$ represent the total amount of token-generating queries made by the CUWE adversary $\mathcal{A}$, where $Q$ is the sum of $Q_h = Q_h(\lambda)$ (the total number of honest token-generating queries), and $Q_c = Q_c(\lambda)$ (the amount of corrupt token-generating queries). For $i \in [0, Q]$, we define $\mathsf{Hybrid}_{3,i}$ to be identical to $\mathsf{Hybrid}_3$, except that the first $i$ token-generating queries are handled as in $\mathsf{Hybrid}_4$, while the remaining $Q_i$ queries are processed as in $\mathsf{Hybrid}_3$. We observe that $\mathsf{Hybrid}_{3,0}$ is identical to $\mathsf{Hybrid}_3$, while $\mathsf{Hybrid}_{3,Q}$ is the identical to $\mathsf{Hybrid}_4$. To prove security, we assume that there exists an adversary $\mathcal{A}$ for which the advantage of distinguishing $\mathsf{Hybrid}_{3,i}$ and $\mathsf{Hybrid}_{3,i+1}$, for $i \in [0, Q-1]$ is non-negligible. Based on this assumption, we construct a PPT reduction algorithm $\mathcal{B}$ capable of breaking the security game of $i\mathcal{O}$.

We can build this reduction for the security of $i\mathcal{O}$: First, $\mathcal{B}$ runs similarly to $\mathsf{Hybrid}_3$, except that the initial $i$ token-generating queries are responded to as in $\mathsf{Hybrid}_4$. For query $i+1$, the reduction algorithm $\mathcal{B}$ computes $k_p' \leftarrow \mathsf{PIDE.Puncture}(k_p^*, t^*, x, m_0^*, m_1^*)$, for $k_p^* \leftarrow \mathsf{PRF.F}(k_o, r_g^*)$, computes $\mathrm{ct}_0 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_0^*)$, $\mathrm{ct}_1 \leftarrow \mathsf{PIDE.Encrypt}(k_p^*, t, x, m_1^*)$, and $\mathrm{ct}_0' \leftarrow \mathsf{PIDE.Encrypt}(k_p'', t, x, m_0^*)$, $\mathrm{ct}_1' \leftarrow \mathsf{PIDE.Encrypt}(k_p'', t, x, m_1^*)$, for $k_p'' \leftarrow \mathsf{PRF.F}(k_u, r)$ and randomly selects $r \in \{0,1\}^\lambda$, where $t, t'$ are the queried index. Then, $\mathcal{B}$ arranges and orders $\mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}_0', \mathrm{ct}_1'$ lexicographically. It then creates two circuits $\mathrm{ct}_0 = P_u : 1[k_o, k_u, t, t']$, and $\mathrm{ct}_1 = P_u : 2[k_o\{r_g^*\}, k_u, t, t', r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}_0', \mathrm{ct}_1', k_p']$. Next, $\mathcal{B}$ submits both circuits $C_0,\ C_1$ to a $i\mathcal{O}$ challenger and receives back a program $P$, which it designates as $\triangle_{t \to t'} := P_u := P$. If the query proves to be a corrupt token-generating query, $\mathcal{B}$ returns $\triangle_{t \to t'}$ to the $A$ as the response to the query. Otherwise, it keeps the result local. If the adversary $\mathcal{A}$ wins, $\mathcal{B}$ returns 0, indicating that $P$ is the obfuscation of $C_0$; if $\mathcal{A}$ loses, $\mathcal{B}$ returns 1, demonstrating that $P$ is the obfuscation of $C_1$.

$P_u : 2[k_o\{r_g^*\}, k_u, t, t', r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}_0', \mathrm{ct}_1', k_p']$

**Constants:** (Punctured) PRF keys $k_o\{r_g^*\}, k_u$, tag $t, t' \in \mathcal{T}$, point $r_g^* \in \{0,1\}^{2\lambda}$, PIDE ciphertexts $\mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}_0', \mathrm{ct}_1'$, punctured PIDE key $k_p' = k_p\{t^*, x, m_0^*, m_1^*\}$

**Input:** $\mathrm{ct}_t = (x, t, \mathrm{ct}, r_g), r \in \{0,1\}^\lambda$

1. If $r_g = r_g^*$ and $\mathrm{ct} = \mathrm{ct}_b$, output $\mathrm{ct}_{t'} := (x, t', \mathrm{ct}_b', r_g)$.

2. If $r_g = r_g^*$ and $\mathrm{ct} \neq \mathrm{ct}_0, \mathrm{ct}_1$, compute $m \leftarrow \mathsf{PIDE.Decrypt}(k'_p, t, r_g)$.
3. Else compute $m \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, \mathrm{ct})$, for $k_p = \mathsf{PRF.F}(k_o\{r_g^*\}, r_g)$.
4. Compute $r'_g \leftarrow \mathsf{PRG}(r)$.
5. Compute $k'_p \leftarrow \mathsf{PRF.F}(k_u, r'_g)$.
6. Compute $\mathrm{ct}' \leftarrow \mathsf{PIDE.Encrypt}(k'_p, t', x, m)$, and output $\mathrm{ct}_{t'} := (x, t', \mathrm{ct}', r'_g)$.

Now, we analyze these two cases in sequence. When the $i\mathcal{O}$ challenger generates $P$ as the obfuscation of $C_1$, $\mathcal{B}$ provides an exact view of $\mathsf{Hybrid}_{3,i+1}$ to $\mathcal{A}$. Otherwise, if $P$ is selected as the obfuscation of $C_0$, it provides a view corresponding to $\mathsf{Hybrid}_{3,i}$. Furthermore, the programs are functionally comparable with negligible probability of deviation. Correctness holds across messages with minimal exceptions, the only distinction being the hardwired responses for two inputs. In other words, if the adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_{3,i}$ from $\mathsf{Hybrid}_{3,i+1}$ with non-negligible probability, then our reduction algorithm $\mathcal{B}$ breaks the security of $i\mathcal{O}$ by non-negligible probability.

**Claim 7.** Assuming PRF is a selectively secure PRF, we have $\mathsf{Hybrid}_4 \approx \mathsf{Hybrid}_5$.

*Proof.* If there exists an adversary $\mathcal{A}$ that can distinguish $\mathsf{Hybrid}_4$ from $\mathsf{Hybrid}_5$, then a PPT reduction algorithm $\mathcal{B}$ can be constructed to break the PRF security game. The reduction for the security of PRF is as follows: First, $\mathcal{B}$ interacts with the CUWE adversary $\mathcal{A}$ in the same way as in $\mathsf{Hybrid}_4$ except that it selects a random $r_g^* \in \{0,1\}^{2\lambda}$ and sends it to the punctured PRF challenger. The challenger then returns a punctured PRF key $k_p\{r_g^*\}$ and a challenge value $v$. $\mathcal{B}$ lets $k_p^* := v$. We remark that the punctured PRF key $k_p\{r_g^*\}$ is sufficient to calculate the challenge ciphertext and respond to the oracle queries of $\mathcal{A}$ as in $\mathsf{Hybrid}_4$. If the adversary $\mathcal{A}$ wins, $\mathcal{B}$ returns 1, indicating that $v = \mathsf{PRF.F}(k, r_g^*)$ for some PRF key $k$; otherwise, $\mathcal{B}$ returns 0 to demonstrate that $v$ was chosen randomly.

Now, we analyze these two cases in sequence. When the PRF challenger generates $v$ as $\mathsf{PRF.F}(k, r_g^*)$, $\mathcal{B}$ provides a view to $\mathcal{A}$ corresponding exactly to $\mathsf{Hybrid}_4$. On the other hand, if $v$ is selected randomly, the view corresponds to $\mathsf{Hybrid}_5$. Therefore, if the adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_4$ from $\mathsf{Hybrid}_5$ with non-negligible probability, then our reduction algorithm $\mathcal{B}$ breaks the security of PRF with non-negligible probability.

**Claim 8.** Assuming PIDE is a selectively secure PIDE scheme, we have $\mathsf{Hybrid}_5 \approx \mathsf{Hybrid}_6$.

*Proof.* We note that the only distinction between $\mathsf{Hybrid}_5$ and $\mathsf{Hybrid}_6$ is that the encrypted message in $\mathsf{Hybrid}_6$ is always $m_0^*$, while in $\mathsf{Hybrid}_5$ the message may be $m_0^*$ or $m_1^*$, depending on the coin flip $b$. Additionally, when $b = 0$, the views of the combined games are identical. Therefore, if there is an advantage for adversary $\mathcal{A}$ in predicting $b$ between $\mathsf{Hybrid}_5$ and $\mathsf{Hybrid}_6$, it must be focused merely on $b = 1$. Assuming that there is an adversary $\mathcal{A}$ where the advantage in distinguishing $\mathsf{Hybrid}_5$ from $\mathsf{Hybrid}_6$ is non-negligible, we build a PPT reduction algorithm $\mathcal{B}$ that can break the PIDE secure game.

Then we can build this reduction for the security of PIDE: First, $\mathcal{B}$ runs as in $\mathsf{Hybrid}_5$ when interacting with the CUWE adversary $\mathcal{A}$, with the exception that it sends the challenge messages $m_0^*$, $m_1^*$, instance $x$, and tag $t^*$ to the PIDE challenger. This challenger responds with the punctured PIDE key $k'_p \leftarrow \mathsf{PIDE.Puncture}(k_p^*, t^*, x, m_0^*, m_1^*)$ and two ciphertexts $\mathrm{ct}'_0, \mathrm{ct}'_1$. $\mathcal{B}$ sets the challenge CUWE ciphertext as $\mathrm{ct}_{t^*} := (x, t, \mathrm{ct}^* := \mathrm{ct}'_0, r_g^*)$. Let $\mathrm{ct}_0, \mathrm{ct}_1$ include $\mathrm{ct}'_0, \mathrm{ct}'_1$ in lexicographic order. The significance of requiring lexicographic ordering in prior games is apparent here, as the reduction algorithm does not know whether $m_0^*$ is in $\mathrm{ct}'_0$ or $\mathrm{ct}'_1$. Then, $\mathcal{B}$ creates $C_0 = P_d : 2[k_o\{r_g^*\}, k_u, \Gamma, r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, k'_p]$.

Similarly, when responding all token-generating queries, $\mathcal{B}$ randomly chooses a $\beta \in \{0,1\}$ and calculates $\mathrm{ct}''_\beta \leftarrow \mathsf{PIDE.Encrypt}(k''_p, t', x, m_\beta^*)$, $\mathrm{ct}''_{1-\beta} \leftarrow \mathsf{PIDE.Encrypt}(k''_p, t', x, m_{1-\beta}^*)$, for $k''_p \leftarrow \mathsf{PRF.F}(k_u, r)$ and randomly selects $r \in \{0,1\}^\lambda$. Next, $\mathcal{B}$ creates $C_1 := P_u : 2[k_o\{r_g^*\}, k_u, t, t', r_g^*, \mathrm{ct}_0, \mathrm{ct}_1, \mathrm{ct}''_0, \mathrm{ct}''_1, k'_p]$ and uses it to respond to the token-generating query. We observe that the choice of $\beta$ incurs a $1/2$ security loss. Encryption queries are responded in an uncomplicated form via the program $P_e : 2[k_o\{r_g^*\}]$. Finally, if the adversary $\mathcal{A}$ wins, then $\mathcal{B}$ returns 1, indicating that $\mathrm{ct}^* = \mathrm{ct}'_0$ is an encryption of $m_1^*$; if $\mathcal{A}$ loses, $\mathcal{B}$ returns 0, demonstrating that $\mathrm{ct}^* = \mathrm{ct}'_0$ is an encryption for $m_0^*$.

Now, we analyze these two cases in sequence. If the PIDE challenger generates $\mathrm{ct}^* = \mathrm{ct}'_0$ as $\mathsf{PIDE.Encrypt}(k_p^*, m_1^*)$, then $\mathcal{B}$ presents an exact view of $\mathsf{Hybrid}_5$ to $\mathcal{A}$. Otherwise, if $\mathrm{ct}^* = \mathrm{ct}'_0$ is generated as $\mathsf{PIDE.Encrypt}(k_p^*, m_0^*)$, the views correspond to $\mathsf{Hybrid}_6$. Therefore, if the adversary $\mathcal{A}$ can distinguish $\mathsf{Hybrid}_5$ from $\mathsf{Hybrid}_6$ with non-negligible probability, then our reduction algorithm $\mathcal{B}$ compromises the security of PIDE with non-negligible probability.

Finally, from the above hybrid argument, we can finalize that the advantage of any PPT adversary in the hybrid game $\mathsf{Hybrid}_1$ is negligibly greater than its advantage in the hybrid game $\mathsf{Hybrid}_6$. However, since no information about bit $b$ is provided in $\mathsf{Hybrid}_6$, the advantage of any adversary is 0. Therefore, our CUWE scheme is secure.

## 6 Functional witness encryption with updatable ciphertext

To enhance access control, we consider an updatable ciphertext function witness encryption scheme. Unlike the previous CUWE scheme, decryption can only retrieve the function value $f(m, w)$ of message $m$ and cannot access any other information about message $m$.

**Definition 10** (Functional witness encryption with updatable ciphertext)**.** A tag-based $t \in \mathcal{T}$ functional witness encryption with updatable ciphertext (CUFWE) scheme for an NP language $\mathcal{L}$ with witness relation $R$ and a class of functions $\{f_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple of five PPT algorithms (Setup, Encrypt, Decrypt, TokGen, Update) defined as follows.

• $(\mathrm{pp}_e, \mathrm{pp}_d, \mathrm{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \Gamma)$ : on input a security parameter $1^\lambda$ and a predicate $\Gamma$, Setup outputs encryption parameter $\mathrm{pp}_e$, a decryption parameter $\mathrm{pp}_d$, and the master secret key msk.

• $\triangle_{t\to t'} \leftarrow \mathsf{TokGen}(\mathrm{msk}, t, t')$ : on input a master secret key msk, tags $t, t' \in \mathcal{T}$, TokGen outputs update token $\triangle_{t\to t'}$.

• $\mathrm{ct}_t \leftarrow \mathsf{Encrypt}(\mathrm{pp}_e, t, x, (f, m))$ : on input encryption parameter $\mathrm{pp}_e$, a tag $t$, an instance $x \in \mathcal{L}$ and a message of the form $(f, m) \in f_\lambda \times \mathcal{M}$, Encrypt outputs ciphertext $\mathrm{ct}_t$.

• $\mathrm{ct}_{t'} \leftarrow \mathsf{Update}(\triangle_{t\to t'}, \mathrm{ct}_t)$ : on input an update token $\triangle_{t\to t'}$, a original ciphertext $\mathrm{ct}_t$, Update outputs an update ciphertext $\mathrm{ct}_{t'}$.

• $f(m, w) \leftarrow \mathsf{Decrypt}(\mathrm{pp}_d, t_d, w, \mathrm{ct}_t/\mathrm{ct}_{t'})$ : on input decryption parameter $\mathrm{pp}_d$, a tag $t_d$, a witness $w \in \mathcal{W}$, and a ciphertext $\mathrm{ct}_t$ (either a nonupdated one $\mathrm{ct}_t$ or an updated one $\mathrm{ct}_{t'}$), Decrypt outputs $f(m, w)$ if $\Gamma(t) = 1$ or $\Gamma(t') = 1$, else outputs $\perp$.

**Construction of CUFWE.** Now, we convert our CUWE scheme into an CUFWE scheme. Our construction of CUFWE requires the following cryptographic primitives: $\mathsf{PRF} = (\mathsf{PRF.G}, \mathsf{PRF.F}, \mathsf{PRF.Puncture})$, $\mathsf{CUWE} = (\mathsf{CUWE.Setup}, \mathsf{CUWE.TokGen}, \mathsf{CUWE.Encrypt}, \mathsf{CUWE.Update}, \mathsf{CUWE.Decrypt})$ and $i\mathcal{O}$. The construction of CUFWE is described below.

- $\mathsf{Setup}(1^\lambda, \Gamma)$ :
  - $k_o \leftarrow \mathsf{PRF.G}(1^\lambda)$, $k_u \leftarrow \mathsf{PRF.G}(1^\lambda)$.
  - $P'_e \leftarrow i\mathcal{O}_{P_e:1[k_o]}$, $P'_d \leftarrow i\mathcal{O}_{P'_d:1[k_o,k_u,\Gamma]}$.
  - Set $\mathrm{msk} = (k_o, k_u), \mathrm{pp}_e = P'_e, \mathrm{pp}_d = P'_d$.
  - Return $(\mathrm{pp}_e, \mathrm{pp}_d, \mathrm{msk})$.
- $\mathsf{TokGen}(\mathrm{msk} := (k_o, k_u), t, t')$ :
  - $\triangle_{t\to t'} \leftarrow i\mathcal{O}_{P'_u:1[k_o,k_u,t,t']}$.
  - Return $\triangle_{t\to t'}$.
- $\mathsf{Encrypt}(\mathrm{pp}_e := P'_e, t, x, M = (f, m))$ :
  - $\mathrm{ct}_t \leftarrow \mathsf{CUWE.Encrypt}(\mathrm{pp}_e, t, x, M)$.
  - Return $\mathrm{ct}_t$.
- $\mathsf{Update}(\triangle_{t\to t'}, \mathrm{ct}_t)$ :
  - $\mathrm{ct}_{t'} \leftarrow \mathsf{CUWE.Update}(\triangle_{t\to t'}, \mathrm{ct}_t)$.
  - Return $\mathrm{ct}_{t'}$.
- $\mathsf{Decrypt}(\mathrm{pp}_d := P_d, t_d, w, \mathrm{ct}_t)$ :
  - $f(m, w) \leftarrow \mathsf{CUWE.Decrypt}(\mathrm{pp}_d, t_d, w, \mathrm{ct}_t)$.
  - Return $f(m, w)$.

$P'_d : 1[k_o, k_u, \Gamma]$

**Constants:** PRF keys $k_o$, $k_u$, Predicate $\Gamma$
**Input:** $t_d, w, \mathrm{ct}_t = (x, t, \mathrm{ct}, r_g)$

1. If $R(x, w) = 0$, or $t_d \not\subseteq t$, or $\Gamma(t) = 0$, output $\perp$.
2. If $\mathrm{ct}_t$ is updated ciphertext, set $k = k_u$, else set $k = k_o$.
3. Compute $k_p = \mathsf{F}(k, r_g)$.
4. Compute $M' \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$.
5. Output $f(m, w)$.

$$P'_u : 1[k_o, k_u, t, t']$$

**Constants:** PRF keys $k_o, k_u$, tag $t, t' \in \mathcal{T}$
**Input:** $\mathrm{ct}_t = (x, t, \mathrm{ct}, r_g), r \in \{0,1\}^\lambda$

1. Compute $k_p = \mathsf{PRF.F}(k_o, r_g)$.
2. Compute $M' \leftarrow \mathsf{PIDE.Decrypt}(k_p, t, x, \mathrm{ct})$.
3. Generate $r'_g \leftarrow \mathsf{PRG.G}(r)$.
4. Compute $k'_p = \mathsf{PRF.F}(k_u, r'_g)$.
5. Compute $\mathrm{ct}' \leftarrow \mathsf{PIDE.Encrypt}(k'_{\mathrm{p}}, t', x, M)$
6. Output $\mathrm{ct}_{t'} := (x, t', \mathrm{ct}', r'_g)$.

The correctness and the security of CUFWE depend on the same assumptions as in the case of CUWE.

## 7 Conclusion

In this work, we investigate the primitive of witness encryption with updatable ciphertext, where the ciphertext is identified by a tag $t$ and an NP instance. Decryption is successful only when the tag in the decryption parameters matches the ciphertext and there is a valid witness. The CUWE scheme can utilize an update token to change tag $t$ of the ciphertext to any other tag, enabling fine-grained control after the ciphertext has been created, supporting dynamic updates post-creation. In particular, we introduce a new cryptographic primitive called deterministic encryption of puncturable instances and propose a practical and efficient construction based on PRFs. Furthermore, we formally prove the security guarantees of our constructions. Finally, we extend the CUWE scheme to CUFWE, which functions only when tag $t$ in the decryption parameters matches the ciphertext and there is a valid witness $w$. In this case, decryption can yield $f(m, w)$, enhancing data access control.

**Acknowledgements** This work was supported in part by National Natural Science Foundation of China (Grant Nos. 62072134, 62472150, U2001205), Major Research Plan of Hubei Province (Grant No. 2023BAA027), and Key Research and Development Program of Hubei Province (Grant No. 2021BEA163).

**References**

1 Garg S, Gentry C, Sahai A, et, al. Witness encryption and its applications. In: Proceedings of Symposium on Theory of Computing Conference, New York, 2013. 467–476
2 Freitag C, Waters B, Wu D J. How to use (plain) witness encryption: registered abe, flexible broadcast, and more. In: Proceedings of Advances in Cryptology-CRYPTO, Santa Barbara, 2023. 498–531
3 Goyal R, Vusirikala S, Waters B. Collusion resistant broadcast and trace from positional witness encryption. In: Proceedings of Public-Key Cryptography, Beijing, 2019. 3–33
4 Liu J, Jager T, Kakvi S A, et al. How to build time-lock encryption. Des Codes Cryptogr, 2018, 86: 2549–2586
5 Wang Y Z, Wang X B, Zhang M W. Homomorphic witness encryption from indistinguishable obfuscation. In: Proceedings of Provable and Practical Security, Wuhan, 2023. 231–250
6 Badrinarayanan S, Garg S, Ishai Y, et al. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In: Proceedings of Advances in Cryptology-ASIACRYPT, Hong Kong, 2017. 275–303
7 Madathil V, Thyagarajan S A K, Vasilopoulos D, et, al. Cryptographic oracle-based conditional payments. In: Proceedings of the 30th Annual Network and Distributed System Security Symposium (NDSS 2023), San Diego, 2023
8 Benhamouda F, Lin H. Mr NISC: multiparty reusable non-interactive secure computation. In: Proceedings of Theory of Cryptography (TCC 2020), Durham, 2020. 349–378
9 Campanelli M, David B, Khoshakhlagh H, et, al. Encryption to the future: a paradigm for sending secret messages to future (anonymous) committees. In: Proceedings of Advances in Cryptology-ASIACRYPT, Taipei, 2022. 151–180
10 Dottling N, Hanzlik L, Magri B, et, al. McFly: verifiable encryption to the future made practical. In: Proceedings of Financial Cryptography and Data Security, Willemstad, 2024. 252–269
11 Abusalah H, Fuchsbauer G, Pietrzak K. Offline witness encryption. In: Proceedings of Applied Cryptography and Network Security, Guildford, 2016. 285–303
12 Zhang M W, Chen S J, Shen J, et al. PrivacyEAFL: privacy-enhanced aggregation for federated learning in mobile crowdsensing. IEEE Trans Inform Forensic Secur, 2023, 18: 5804–5816
13 Campanelli M, Fiore D, Khoshakhlagh H. Witness encryption for succinct functional commitments and applications. In: Proceedings of Public-Key Cryptography, Sydney, 2024. 132–167
14 Barta O, Ishai Y, Ostrovsky R, et, al. On succinct arguments and witness encryption from groups. In: Proceedings of Advances in Cryptology-CRYPTO, 2020. 776–806
15 Tsabary R. Candidate witness encryption from lattice techniques. In: Proceedings of Advances in Cryptology-CRYPTO, Santa Barbara, 2022. 535–559
16 Vaikuntanathan V, Wee H, Wichs D. Witness encryption and null-IO from evasive LWE. In: Proceedings of Advances in Cryptology-ASIACRYPT, Taipei, 2022. 195–221
17 Phuong T V X. Anonymous attribute-based broadcast encryption with hidden multiple access structures. Des Codes Cryptogr, 2024, 92: 1925–1945
18 Derler D, Slamanig D. Practical witness encryption for algebraic languages or how to encrypt under Groth-Sahai proofs. Des Codes Cryptogr, 2018, 86: 2525–2547
19 Wang Y Z, Zhang M W. Witness encryption from smooth projective hashing system. In: Proceedings of the 25th International Conference on Information Security, Bali, 2022. 53–72
20 Gorbunov S, Vaikuntanathan V, Wee H. Attribute-based encryption for circuits. In: Proceedings of Symposium on Theory of Computing Conference, New York, 2013. 545–554

21 Cohen A. What about Bob? The inadequacy of CPA security for proxy reencryption. In: Proceedings of Public-Key Cryptography, Beijing, 2019. 287–316
22 Derler D, Krenn S, Lorunser T, et al. Revisiting proxy re-encryption: forward secrecy, improved security, and applications. In: Proceedings of Public-Key Cryptography, Rio de Janeiro, 2018. 219–250
23 Waters B. A punctured programming approach to adaptively secure functional encryption. In: Proceedings of Advances in Cryptology-CRYPTO, Santa Barbara, 2015. 678–697
24 Garg S, Gentry C, Halevi S, et, al. Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of IEEE 54th Annual Symposium on Foundations of Computer Science, Berkeley, 2013. 40–49
25 Canetti R, Lin H, Tessaro S, et al. Obfuscation of probabilistic circuits and applications. In: Proceedings of Theory of Cryptography (TCC 2015), Warsaw, 2015. 468–497
26 Garg S, Gentry C, Halevi S, et, al. On the implausibility of differing inputs obfuscation and extractable witness encryption with auxiliary input. In: Proceedings of Advances in Cryptology-CRYPTO 2014, Santa Barbara, 2014. 518–535
27 Garg R, Goyal R, Lu G. Dynamic collusion functional encryption and multiauthority attribute-based encryption. In: Proceedings of Public-Key Cryptography, Sydney, 2024. 69–104
28 Fugkeaw S, Sato H. Updating policies in CP-ABE-based access control: an optimized and secure service. In: Proceedings of Service-Oriented and Cloud Computing (ESOCC 2016), Vienna, 2016. 9846
29 Boyle E, Chung K M, Pass R. On extractability obfuscation. In: Proceedings of Theory of Cryptography, San Diego, 2014. 52–73
30 Pal T, Dutta R. Offline witness encryption from witness PRF and randomized encoding in CRS model. In: Proceedings of Information Security and Privacy (ACISP 2019), Christchurch, 2019. 78–96
31 Chvojka P, Jager T, Kakvi S A. Offline witness encryption with semi-adaptive security. In: Proceedings of Applied Cryptography and Network Security (ACNS 2020), Rome, 2020. 231–250
32 Zhandry M. How to avoid obfuscation using witness PRFs. In: Proceedings of Theory of Cryptography (TCC 2016), Beijing, 2016. 421–448
33 Nguyen D D, Phan D H, Pointcheval D. Verifiable decentralized multiclient functional encryption for inner product. In: Proceedings of Advances in Cryptology-ASIACRYPT, Guangzhou, 2023. 33–65
34 Cini V, Ramacher S, Slamanig D, et al. (Inner-Product) functional encryption with updatable ciphertexts. J Cryptol, 2024, 37: 8–47
35 Nisan N, Wigderson A. Hardness vs randomness. J Comput Syst Sci, 1994, 49: 149–167
36 Sahai A, Waters B. How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of Symposium on Theory of Computing Conference (STOC'14), New York, 2014. 475–484