Trapdoor Privacy in Asymmetric Searchable Encryption Schemes

Afonso Arriaga[1] and Qiang Tang[1]

APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
{afonso.delerue,qiang.tang}@uni.lu

Abstract. Asymmetric searchable encryption allows searches to be carried over ciphertexts, through delegation, and by means of trapdoors issued by the owner of the data. Public Key Encryption with Keyword Search (PEKS) is a primitive with such functionality that provides delegation of *exact-match* searches. As it is important that ciphertexts preserve data privacy, it is also important that trapdoors do not expose the user's search criteria. The difficulty of formalizing a security model for trapdoor privacy lies in the verification functionality, which gives the adversary the power of verifying if a trapdoor encodes a particular keyword. In this paper, we provide a broader view on what can be achieved regarding trapdoor privacy in asymmetric searchable encryption schemes, and bridge the gap between previous definitions, which give limited privacy guarantees in practice against search patterns. We propose the notion of *Strong Search Pattern Privacy* for PEKS and construct a scheme that achieves this security notion.

Keywords: Asymmetric Searchable Encryption, PEKS, Trapdoor Privacy, Anonymous IBE, Function Privacy, Search Pattern Privacy, Key Unlinkability.

1 Introduction

As cloud services become increasingly popular, security concerns arise from exposing the users data to third-party service providers. Encryption can be used to protect the user's data privacy, but usability is sacrificed if not even the most basic operations, such as searching over the user's data, can be delegated to the service provider. In the public key setting, Boneh et al. [1] were the first to propose a primitive to tackle this problem. They called it Public Key Encryption with Keyword Search (PEKS), a primitive that provides delegation of *exact-match* searches over ciphertexts. A typical scenario where this primitive can bring great benefits to users (and consequently to service providers wishing to increase their customer base as well) is that of any email system.

Suppose user Alice stores her emails in the servers of some email service provider, so that she can access them from either her laptop or her smartphone. Alice does not trust the service provider or fears that government agencies may require the service provider to hand over all her data. Using standard public key encryption, any user with Alice's public key can send her encrypted emails that only she can decrypt. For Alice to find a particular email later on, the sender could also attach to the email some *searchable ciphertexts*, produced from a PEKS scheme, with keywords that Alice might use when searching for this email. These ciphertexts are searchable upon delegation, meaning that only Alice can authorize the email service provider to search on her behalf by issuing a trapdoor that encodes Alice's search criteria (e.g. ciphertexts that encrypt the keyword "project xx123 - meeting"), generated from her own secret key. The service provider searches through all Alice's emails for those containing searchable ciphertexts that match the issued trapdoor, and returns to her only those with a positive match.

Many efforts have been put in asymmetric searchable encryption in general, as surveyed in [2], most towards more efficient PEKS schemes (or relying on weaker assumptions) or towards primitives with more flexible search queries, such as conjunctive, disjunctive, subset and inner product types of queries. Until recently [3–5], the concern was always to preserve data privacy in the ciphertexts and no attention was paid to possible information leakage from the trapdoors. In fact, some schemes, as the statistically consistent scheme proposed in [6], include the keyword itself in the

trapdoor. In this paper we focus on defining trapdoor privacy for PEKS and constructing a scheme that provably stands up to the definition. Nevertheless, the definition can easily be extended to asymmetric searchable encryption in general.

The difficulty of formalizing a security model for trapdoor privacy lies in the verification functionality of PEKS, which in the public key setting depends on the trapdoor itself and ciphertexts created from publicly known parameters. This provides to any adversary the power to verify if a trapdoor encodes a particular keyword. (The adversary encrypts the chosen keyword under the public key associated with the trapdoor; if the ciphertext matches the trapdoor then the trapdoor encodes the chosen keyword.) Therefore, an offline dictionary attack can always be launched, putting aside the possibility of formalizing the security notion of trapdoor privacy as a traditional choose-then-guess indistinguishability game. In many cases, the keywords encoded in trapdoors are *sufficiently unpredictable* for a dictionary attack to be infeasible. So, defining the right notion of trapdoor privacy is crucial to guarantee that the user's privacy is fully protected.

1.1 Related work

Abdalla et al. [6] proposed a general black-box transformation from *anonymous* Identity-Based Encryption (IBE) to PEKS, where the resulting PEKS scheme is secure in the traditional ciphertext indistinguishability sense. Identities and their secret keys in the original IBE scheme map to keywords and trapdoors in resulting PEKS scheme, respectively. The anonymity requirement informally states that ciphertexts leak no information regarding the identity of the recipient, leading to the commonly desired keyword-privacy guarantees over ciphertexts in PEKS. The standard notion of ciphertext indistinguishability in IBE leads to *computational consistency* in the resulting PEKS scheme, which informally means that it is hard for computationally bounded adversaries to find two distinct keywords such that the trapdoors for the first keyword positively match the ciphertexts of the second keyword. (Note that if keywords are hashed before used in the scheme, *inconsistency* happens at least every time $\mathsf{H}(\mathsf{w}_1) = \mathsf{H}(\mathsf{w}_2)$, where $\mathsf{w}_1 \neq \mathsf{w}_2$.) We refer the reader to Section 2 for precise details on this transformation and to [6] for formal proofs.

This black-box transformation allows us to define a "dual" security notion for IBE that will lead to the desired trapdoor privacy notion in PEKS, and motivates the construction of IBE schemes that can provably satisfy it. This approach was also followed by [3–5], which, to the best of our knowledge, are the only works to address the concerns on trapdoor privacy in asymmetric searchable encryption.

Two distinct scenarios have to be considered to model trapdoor privacy. One in the presence of ciphertexts that positively match the trapdoors, and the other in the absence of such ciphertexts. Consider a toy example where the service provider possesses one ciphertext that belongs to Alice and two trapdoors that Alice issued for searches to be performed on her behalf. The service provider executes the test-search and one of the following cases occurs:

(a) Both trapdoors positively match the stored ciphertext, in which case the trapdoors encode the same keyword.
(b) Only one of the trapdoors match the ciphertext, in which case the trapdoors encode different keywords.
(c) None of the trapdoors positively match the stored ciphertext.

From cases (a) and (b), we can see that, in the presence of ciphertexts that match the trapdoors, an equality relation between the keywords encoded under the trapdoors can be determined trivially. In such cases, the notion of trapdoor privacy focus on revealing as little information as possible on the keywords themselves[1]. Very recently, Boneh et al. [4] put forward two formal definitions of different strengths for IBE, inspired by the security definition given for deterministic encryption in [7]: *Function Privacy* and *Enhanced Function Privacy*. The latter leads to a security notion in PEKS (after the black-box transformation in [6]), which addresses this scenario.

[1] Note that some information is inevitably leaked because of the verification functionality. E.g. if the trapdoor does not match a ciphertext which encrypts a particular known keyword, it means that the trapdoor does not encode this keyword.

Case (c) covers the scenario where trapdoors do not match any ciphertext. It is in Alice's best interest to hide her search pattern from the service provider. If the search pattern is revealed, the attacker could concentrate its resources on breaking the privacy of trapdoors encoding the most frequent keywords, which a priori are the most relevant to Alice. This issue is particularly important for PEKS due to the possibility of launching dictionary attacks. Nishioka [3] proposed a model denoted *Search Pattern Privacy*, which partially addresses this scenario. However, the model limits the distinguishing game to two trapdoors, which provides insufficient privacy guarantees in practice, considering that an actual attacker may have access to a much larger number of (possibly related) trapdoors. As we show in Section 4, and contrarily to intuition, the so-called hybrid argument does not apply here, unless trapdoors can be efficiently re-randomized. Also, after the transformation from IBE to PEKS, Function Privacy leads to a security definition that provides limited privacy guarantees against search patterns, since the resulting model prevents the adversary from being challenged with trapdoors encoding the same keyword.

1.2 Our contributions

In this paper, we provide a broader view on what can be achieved regarding trapdoor privacy in asymmetric searchable encryption, and bridge the gap between currently existent definitions. In more detail, our contributions are the following:

- We formulate the "dual" notion of Search Pattern Privacy, which we call *Weak Key Unlinkability* for IBE. If an IBE scheme is provably secure in the Weak Key Unlinkability model, the resulting PEKS scheme from the black-box transformation in [6] has Search Pattern Privacy. We then show that Search Pattern Privacy, as defined in [3], is insufficient in practice. We do so by constructing a new anonymous IBE scheme with Weak Key Unlinkability, based on the anonymous IBE scheme by Boyen and Waters [8]. Applying the black-box transformation to this scheme therefore results in a PEKS scheme with Search Pattern Privacy. However, we show that the resulting scheme fails to hide search patterns when more than two trapdoors have been issued. (Of independent interest, the new anonymous IBE scheme is more efficient than that in [8], and eliminates the *selective-ID* constraint, but the security proofs rely on random oracles.)
- We propose a new security model, strictly stronger than Weak Key Unlinkability, which we call *Strong Key Unlinkability* for IBE. The model allows the adversary to be challenged with multiple secret keys, which leads to a new notion that we refer to as *Strong Search Pattern Privacy* for PEKS, in which the adversary is allowed to be challenged with multiple trapdoors, instead of just two.
- We compare the different notions of security and show that *Key Unlinkability* and *Function Privacy* are independent security notions. We provide counter-examples to show that Strong Key Unlinkability does not imply Function Privacy, neither does Enhanced Function Privacy implies Weak Key Unlinkability.
- We introduce a new hardness assumption for composite order groups. We call this assumption the *Composite Decisional Diffie-Hellman* (CDDH) and show that it is weaker than the *Composite 3-party Diffie-Hellman* (C3DH) assumption made in [9] by Boneh and Waters. Assuming CDDH is intractable, we extend our new IBE scheme to groups of composite order, and prove that the scheme now satisfies Strong Key Unlinkability security.

1.3 Structure of the paper

In the following section we recall some background on bilinear groups of prime order and composite order, introduce the new CDDH assumption, and show that this assumption is weaker than the well-established C3DH assumption. We also define consistency and standard security notions for IBE and PEKS, and describe the black-box transformation from anonymous IBE to PEKS. In Section 3, we recall Nishioka's Search Pattern Privacy model for PEKS, and introduce the strictly stronger *Strong Search Pattern Privacy* model. We also formulate the "dual" proprieties for IBE, *Weak Key Unlinkability* and *Strong Key Unlinkability*, which lead to *Search Pattern Privacy* and

Strong Search Pattern Privacy for PEKS, respectively, after the black-box transformation from IBE to PEKS. In Section 4, we construct a new anonymous IBE scheme and prove it secure in the Weak Key Unlinkability model. We then present a concrete attack to show that the scheme does not satisfy the notion of Strong Key Unlinkability. Finally, we extend the newly introduced scheme to groups of composite order, and prove that the scheme is now secure in the Strong Key Unlinkability model. We end by some concluding remarks and point out possible directions for future work, in Section 5.

2 Preliminaries

NOTATION. We write $\mathsf{a} \leftarrow \mathsf{b}$ to denote the algorithmic action of assigning the value of b to the variable a. We use $\perp \notin \{0,1\}^\star$ to denote a special failure symbol. If $\mathbb{S}$ is a set, we write $\mathsf{a} \leftarrow_\$ \mathbb{S}$ for sampling a from $\mathbb{S}$ uniformly at random. If $\mathcal{A}$ is a probabilistic algorithm we write $\mathsf{a} \leftarrow_\$ \mathcal{A}(\mathsf{i}_1, \mathsf{i}_2, \ldots, \mathsf{i}_n)$ for the action of running $\mathcal{A}$ on inputs $\mathsf{i}_1, \mathsf{i}_2, \ldots, \mathsf{i}_n$ with random coins, and assigning the result to a. If a is a variable, $|\mathsf{a}|$ denotes the length in bits of its representation. We denote by $\mathsf{a}||\mathsf{b}$ the concatenation of variables a and b, represented as bit-strings.

GAMES. In this paper we use the code-based game-playing language [10]. Each game has an Initialize and a Finalize procedure. It also has specifications of procedures to respond to an adversary's various queries. A game is run with an adversary $\mathcal{A}$ as follows. First Initialize runs and its outputs are passed to $\mathcal{A}$. Then $\mathcal{A}$ runs and its oracle queries are answered by the procedures of the game. When $\mathcal{A}$ terminates, its output is passed to Finalize, which returns the outcome of the game. In each game, we restrict attention to legitimate adversaries, which is defined specifically for each game. We use lists as data structures to keep relevant state in the games. The empty list is represented by empty square brackets $[\,]$. We denote by $\mathsf{list} \leftarrow \mathsf{a} : \mathsf{list}$ the action of appending element a to the head of list. To access the value stored in index i of list and assign it to a, we write $\mathsf{a} \leftarrow \mathsf{list}[\mathsf{i}]$. To denote the number of elements in list, we use $|\mathsf{list}|$. Unless stated otherwise, lists are initialized empty and variables are first assigned with $\perp$.

2.1 Bilinear groups

We first revise pairings over prime-order groups and the associated *Decision Bilinear Diffie-Hellman* (DBDH) and *Decision Linear* (DLIN) assumptions [11, 12]. We then revise pairings over composite-order groups [13], introduce the new *Composite Decision Diffie-Hellman* (CDDH) assumption, and show that this assumption is weaker than the well-established *Composite 3-party Diffie-Hellman* (C3DH) assumption made in [9].

Bilinear groups of prime order

Definition 1. *A prime-order bilinear group generator is an algorithm $\mathcal{G}_\mathcal{P}$ that takes as input a security parameter λ and outputs a description $\Gamma = (\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ where:*

- *$\mathbb{G}$ and $\mathbb{G}_\mathsf{T}$ are groups of order p with efficiently-computable group laws, where p is a λ-bit prime.*
- *g is a generator of $\mathbb{G}$.*
- *$\mathbf{e}$ is an efficiently-computable bilinear pairing $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathsf{T}$, i.e., a map satisfying the following properties:*
  - *Bilinearity: $\forall \mathsf{a}, \mathsf{b} \in \mathbb{Z}_\mathsf{p}, \mathbf{e}(\mathsf{g}^\mathsf{a}, \mathsf{g}^\mathsf{b}) = \mathbf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{ab}}$;*
  - *Non-degeneracy: $\mathbf{e}(\mathsf{g}, \mathsf{g}) \neq 1$.*

Definition 2. *Let $\Gamma = (\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ be the description output by $\mathcal{G}_\mathcal{P}(\lambda)$. We say the DBDH assumption holds for description Γ if, for every PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ.*

$$\mathbf{Adv}^{\mathsf{DBDH}}_{\Gamma,\mathcal{A}} := 2 \cdot \Pr[\mathsf{DBDH} \Rightarrow \mathsf{True}] - 1,$$

where game DBDH *is described in Fig. 1.*

procedure Initialize(λ):
$\Gamma \leftarrow_\$ \mathcal{G}_{\mathcal{P}}(\lambda)$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{z}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{z}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{z}_3 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{Z} \leftarrow_\$ \mathbb{G}_\mathsf{T}$
$\mathsf{bit} \leftarrow_\$ \{0,1\}$
if $\mathsf{bit} = 0$ return $(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_3}, \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3})$
else return $(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_3}, \mathsf{Z})$

procedure Finalize(bit'):
if $\mathsf{bit} = \mathsf{bit}'$ return True
else return False

Fig. 1. Game DBDH.

procedure Initialize(λ):
$\Gamma \leftarrow_\$ \mathcal{G}_{\mathcal{P}}(\lambda)$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{z}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{z}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{z}_3 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{z}_4 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
$\mathsf{Z} \leftarrow_\$ \mathbb{G}_\mathsf{T}$
$\mathsf{bit} \leftarrow_\$ \{0,1\}$
if $\mathsf{bit} = 0$ return $(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_1\mathsf{z}_3}, \mathsf{g}^{\mathsf{z}_2\mathsf{z}_4}, \mathsf{g}^{\mathsf{z}_3+\mathsf{z}_4})$
else return $(\Gamma, \mathsf{g}^{\mathsf{z}_1}, \mathsf{g}^{\mathsf{z}_2}, \mathsf{g}^{\mathsf{z}_1\mathsf{z}_3}, \mathsf{g}^{\mathsf{z}_2\mathsf{z}_4}, \mathsf{Z})$

procedure Finalize(bit'):
if $\mathsf{bit} = \mathsf{bit}'$ return True
else return False

Fig. 2. Game DLIN.

Definition 3. *Let $\Gamma = (\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ be the description output by $\mathcal{G}_{\mathcal{P}}(\lambda)$. We say the DLIN assumption holds for description Γ if, for every PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ.*

$$\mathbf{Adv}^{\mathsf{DLIN}}_{\Gamma,\mathcal{A}} := 2 \cdot \Pr[\mathsf{DLIN} \Rightarrow \mathsf{True}] - 1,$$

where game DLIN *is described in Fig. 2.*

Bilinear groups of composite order

Definition 4. *A composite-order bilinear group generator is an algorithm $\mathcal{G}_{\mathcal{C}}$ that takes as input a security parameter λ and outputs a description $\Gamma = (\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ where:*

- *$\mathbb{G}$ and $\mathbb{G}_\mathsf{T}$ are groups of order $\mathsf{n} = \mathsf{pq}$, where p and q are independent λ-bit primes, with efficiently computable group laws.*
- *g is a generator of $\mathbb{G}$.*
- *$\mathbf{e}$ is an efficiently-computable bilinear pairing $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathsf{T}$, i.e., a map satisfying the following properties:*
  - *Bilinearity: $\forall \mathsf{a}, \mathsf{b} \in \mathbb{Z}_\mathsf{n}, \mathbf{e}(\mathsf{g}^\mathsf{a}, \mathsf{g}^\mathsf{b}) = \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{ab}}$;*
  - *Non-degeneracy: $\mathbf{e}(\mathsf{g},\mathsf{g}) \neq 1$.*

Subgroups $\mathbb{G}_\mathsf{p} \subset \mathbb{G}$ and $\mathbb{G}_\mathsf{q} \subset \mathbb{G}$ of order p and order q can be generated respectively by $\mathsf{g}_\mathsf{p} = \mathsf{g}^\mathsf{q}$ and $\mathsf{g}_\mathsf{q} = \mathsf{g}^\mathsf{p}$. We recall some important facts regarding these groups:

- $\mathbb{G} = \mathbb{G}_\mathsf{p} \times \mathbb{G}_\mathsf{q}$
- $\mathbf{e}(\mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q}) = \mathbf{e}(\mathsf{g}^\mathsf{q}, \mathsf{g}^\mathsf{p}) = \mathbf{e}(\mathsf{g},\mathsf{g})^\mathsf{n} = 1$
- $\mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{p})^\mathsf{a} \cdot (\mathsf{g}_\mathsf{q})^\mathsf{b}) = \mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{p})^\mathsf{a}) \cdot \mathbf{e}(\mathsf{g}_\mathsf{p}, (\mathsf{g}_\mathsf{q})^\mathsf{b}) = \mathbf{e}(\mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{p})^\mathsf{a}$

Definition 5. *Let $\Gamma = (\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ be the description output by $\mathcal{G}_{\mathcal{C}}(\lambda)$ and $\Gamma' = (\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$, where $\mathsf{n} \leftarrow \mathsf{pq}$. We say the CDDH assumption holds for description Γ' if, for every PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ.*

$$\mathbf{Adv}^{\mathsf{C3DH}}_{\Gamma',\mathcal{A}} := 2 \cdot \Pr[\mathsf{C3DH} \Rightarrow \mathsf{True}] - 1,$$

where game C3DH *is described in Fig. 3.*

Definition 6. *Let $\Gamma = (\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$ be the description output by $\mathcal{G}_{\mathcal{C}}(\lambda)$ and $\Gamma' = (\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g})$, where $\mathsf{n} \leftarrow \mathsf{pq}$. We say the CDDH assumption holds for description Γ' if, for every PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ.*

$$\mathbf{Adv}^{\mathsf{CDDH}}_{\Gamma',\mathcal{A}} := 2 \cdot \Pr[\mathsf{CDDH} \Rightarrow \mathsf{True}] - 1,$$

where game CDDH *is described in Fig. 4.*

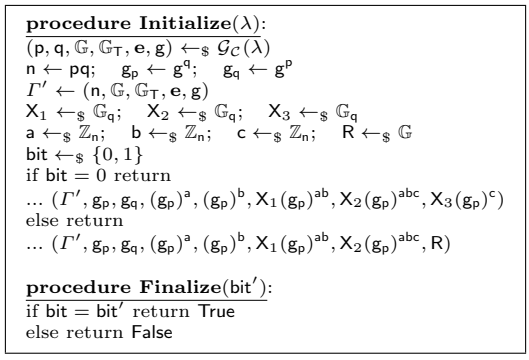

Fig. 3. Game C3DH.

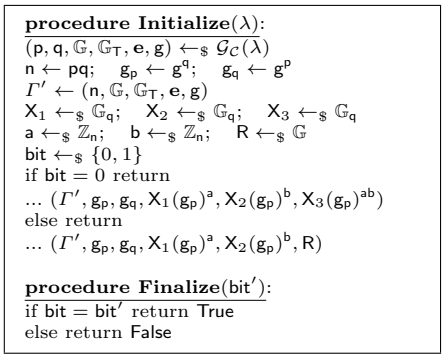

Fig. 4. Game CDDH.

In game C3DH, adversary is given a tuple $(\Gamma', \mathsf{g_p}, \mathsf{g_q}, (\mathsf{g_p})^{\mathsf{a}}, (\mathsf{g_p})^{\mathsf{b}}, \mathsf{X}_1(\mathsf{g_p})^{\mathsf{ab}}, \mathsf{X}_2(\mathsf{g_p})^{\mathsf{abc}}, \mathsf{Z})$ and has to decide whether $\mathsf{Z} = \mathsf{X}_3(\mathsf{g_p})^{\mathsf{c}}$, for some $\mathsf{X}_3 \in \mathbb{G}_{\mathsf{q}}$. For convenience, we rewrite this as $(\Gamma', \mathsf{g_p}, \mathsf{g_q}, (\mathsf{g_p})^{\mathsf{a}}, (\mathsf{g_p})^{\mathsf{b}}, \mathsf{X}_1(\mathsf{g_p})^{\mathsf{ab}}, \mathsf{Y}, \mathsf{X}_3(\mathsf{g_p})^{\mathsf{c}})$, where Y is either $\mathsf{X}_2(\mathsf{g_p})^{\mathsf{abc}}$ or random in $\mathbb{G}$. Now, notice that $(\Gamma', \mathsf{g_p}, \mathsf{g_q}, \mathsf{X}_1(\mathsf{g_p})^{\mathsf{ab}}, \mathsf{X}_3(\mathsf{g_p})^{\mathsf{c}}, \mathsf{Y})$ is a CDDH tuple. Therefore, CDDH is a weaker assumption than C3DH.

2.2 Anonymous Identity-Based Encryption

Identity-Based Encryption (IBE) was first introduced by Boneh and Franklin [11]. For convenience, and following the same approach as in [8], we provide *real-or-random* definitions for *semantic security* and *anonymity*, instead of the more usual (but asymptotically equivalent) *left-or-right* indistinguishability games[2].

An IBE scheme $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ is specified by four polynomial-time algorithms associated with a message space $\mathcal{M}$ and an identity space $\mathcal{I}$.

- $\mathsf{Setup}(\lambda)$: On input the security parameter λ, this algorithm returns a master secret key Msk and public parameters params.
- $\mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id})$: On input public parameters params, a master secret key Msk and an identity $\mathsf{id} \in \mathcal{I}$, this algorithm outputs a secret key sk.
- $\mathsf{Enc}(\mathsf{params}, \mathsf{m}, \mathsf{id})$: On input public parameters params, a message $\mathsf{m} \in \mathcal{M}$ and an identity $\mathsf{id} \in \mathcal{I}$, this algorithm outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{params}, \mathsf{c}, \mathsf{sk})$: On input public parameters params, a ciphertext c and a secret key sk, this algorithm outputs either a message m or a failure symbol $\perp$.

The correctness of an IBE scheme requires that decryption reverses encryption, i.e., for any $\lambda \in \mathbb{N}$, any $(\mathsf{Msk}, \mathsf{params}) \leftarrow_{\$} \mathsf{Setup}(\lambda)$, any $\mathsf{id} \in \mathcal{I}$, any $\mathsf{m} \in \mathcal{M}$, we have that $\mathsf{Dec}(\mathsf{params}, \mathsf{Enc}(\mathsf{params}, \mathsf{m}, \mathsf{id}), \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id})) = \mathsf{m}$.

Definition 7. *An IBE scheme Π is semantically secure if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{IBE\text{-}IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{IBE\text{-}IND\text{-}CPA} \Rightarrow \mathsf{True}] - 1,$$

where game IBE-IND-CPA *is described in Fig. 5.*

[2] There is a simple reduction that loses a factor of 2 in tightness from *real-or-random* to *left-or-right* definitions.

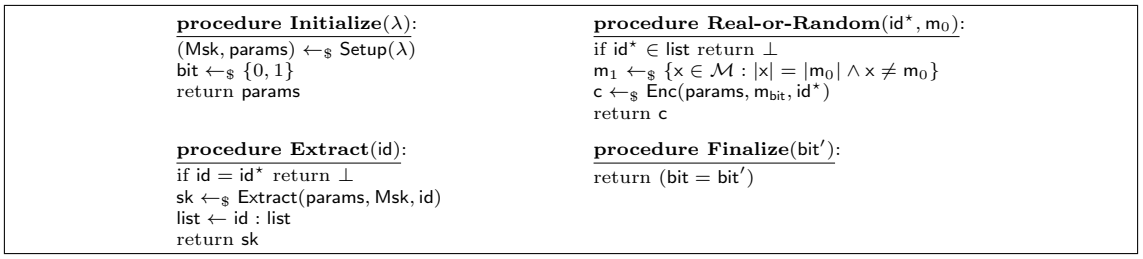

Fig. 5. Game IBE-IND-CPA. Adversary is legitimate if it only calls Real-or-Random once.

Definition 8. *An IBE scheme Π is anonymous if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{IBE\text{-}ANO}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{IBE\text{-}ANO} \Rightarrow \mathsf{True}] - 1,$$

where game IBE-ANO *is described in Fig. 6.*

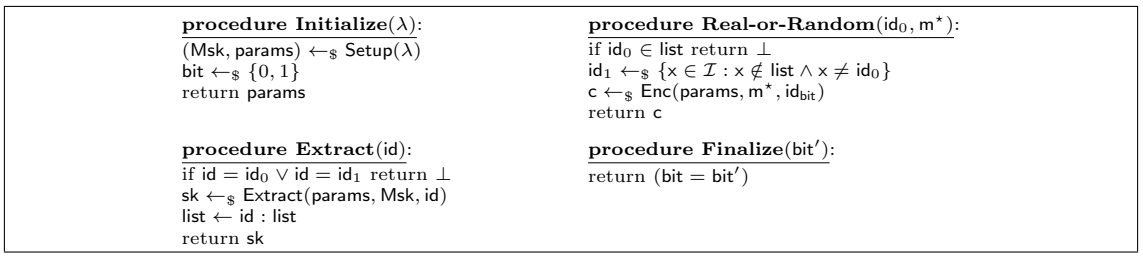

Fig. 6. Game IBE-ANO. Adversary is legitimate if it only calls Real-or-Random once.

2.3 Public Key Encryption with Keyword Search

Public Key Encryption with Keyword Search (PEKS) [1] is a form of Asymmetric Searchable Encryption with *exact-match* searches. The standard notion of security for keyword privacy in ciphertexts is semantic security, which ensures that any partial information on the keyword cannot be feasibly determined from the ciphertext, unless a trapdoor for that keyword is available. The notion of computational consistency is taken from [6]. It guarantees that the primitive fulfills its function. We use the alternative (but asymptotically equivalent) *real-or-random* definitions.

A PEKS scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$ is specified by four polynomial-time algorithms associated with a keyword space $\mathcal{W}$.

- $\mathsf{KeyGen}(\lambda)$: On input the security parameter λ, this algorithm returns a private/public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{PEKS}(\mathsf{pk}, \mathsf{w})$: On input a public key pk and a keyword $\mathsf{w} \in \mathcal{W}$, this algorithm produces a searchable ciphertext c.
- $\mathsf{Trapdoor}(\mathsf{pk}, \mathsf{sk}, \mathsf{w})$: On input a public key pk, a secret key sk and a keyword $\mathsf{w} \in \mathcal{W}$, this algorithm outputs a trapdoor tp for w.
- $\mathsf{Test}(\mathsf{pk}, \mathsf{c}, \mathsf{tp})$: On input a public key pk, a searchable ciphertext c and a trapdoor tp, this algorithm outputs either True or False.

Definition 9. *A PEKS scheme $\mathcal{E}$ is computationally consistent if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{PEKS\text{-}CONSIST}}_{\mathcal{E},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{PEKS\text{-}CONSIST} \Rightarrow \mathsf{True}] - 1,$$

where game PEKS-CONSIST *is described in Fig. 7.*

Definition 10. *A PEKS scheme $\mathcal{E}$ is semantically secure if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{PEKS\text{-}IND\text{-}CPA}}_{\mathcal{E},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{PEKS\text{-}IND\text{-}CPA} \Rightarrow \mathsf{True}] - 1,$$

where game PEKS-IND-CPA *is described in Fig. 8.*

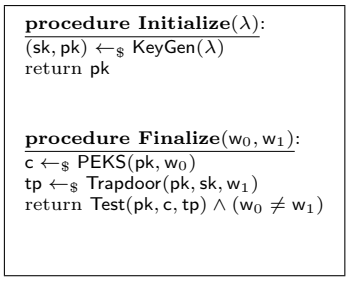

Fig. 7. Game PEKS-CONSIST.

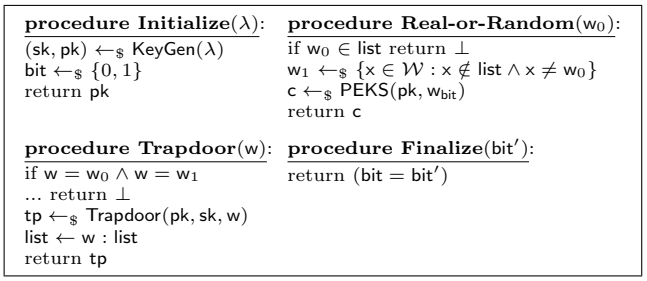

Fig. 8. Game PEKS-IND-CPA. Adversary is legitimate if it only calls Real-or-Random once.

2.4 From anonymous IBE to PEKS: a black-box transformation

A *semantically secure* and *computationally consistent* PEKS scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$ can be constructed from an *anonymous* and *semantically secure* IBE scheme $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$, by applying the black-box transformation given by Abdalla et al. in [6]. For completeness, we describe the transformation bellow.

- $\mathsf{KeyGen}(\lambda)$: $(\mathsf{Msk}, \mathsf{params}) \leftarrow_\$ \mathsf{Setup}(\lambda)$; $(\mathsf{sk}, \mathsf{pk}) \leftarrow (\mathsf{Msk}, \mathsf{params})$; return $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{PEKS}(\mathsf{pk}, \mathsf{w})$: $\mathsf{id} \leftarrow \mathsf{w}$; $\mathsf{m} \leftarrow_\$ \mathcal{M}$; $\hat{\mathsf{c}} \leftarrow_\$ \mathsf{Enc}(\mathsf{pk}, \mathsf{m}, \mathsf{id})$; $\mathsf{c} \leftarrow (\mathsf{m}, \hat{\mathsf{c}})$; return c.
- $\mathsf{Trapdoor}(\mathsf{pk}, \mathsf{sk}, \mathsf{w})$: $\mathsf{id} \leftarrow \mathsf{w}$; $\mathsf{sk}_{\mathsf{id}} \leftarrow_\$ \mathsf{Extract}(\mathsf{pk}, \mathsf{sk}, \mathsf{id})$; $\mathsf{tp}_{\mathsf{w}} \leftarrow \mathsf{sk}_{\mathsf{id}}$; return tp_{w}.
- $\mathsf{Test}(\mathsf{pk}, \mathsf{c}, \mathsf{tp}_{\mathsf{w}})$: $(\mathsf{m}, \hat{\mathsf{c}}) \leftarrow \mathsf{c}$; $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{tp}_{\mathsf{w}}$; $\hat{\mathsf{m}} \leftarrow \mathsf{Dec}(\mathsf{pk}, \hat{\mathsf{c}}, \mathsf{sk}_{\mathsf{id}})$; if $\mathsf{m} = \hat{\mathsf{m}}$ then return True else return False.

Note that the keyword space $\mathcal{W}$ of the derived PEKS scheme is the identity space $\mathcal{I}$ of the original IBE scheme.

3 Security Definitions

In this section, we recall Nishioka's *Search Pattern Privacy* model for PEKS [3]. We then strengthen the model to better reflect real attack scenarios, by allowing the adversary to be challenged with multiple trapdoors, instead of just two. We denote this model by *Strong Search Pattern Privacy*. We also formulate the "dual" proprieties for IBE, *Weak Key Unlinkability* and *Strong Key Unlinkability*, which lead to *Search Pattern Privacy* and *Strong Search Pattern Privacy* for PEKS, respectively, after the black-box transformation from IBE to PEKS described in Section 2.

3.1 Search Pattern Privacy for PEKS

In Search Pattern Privacy model from [3], the adversary receives two trapdoors and is asked to determine whether the two trapdoors encode the same or different keywords, which are uniformly sampled from the keyword space to ensure that these are *sufficiently unpredictable* and avoid brute force attacks[3].

Definition 11. *A PEKS scheme $\mathcal{E}$, associated with a non-polynomial size keyword space, has (Weak) Search Pattern Privacy if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{WEAK\text{-}SSP}}_{\mathcal{E},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{WEAK\text{-}SSP}(\lambda) \Rightarrow \mathsf{True}] - 1,$$

where game WEAK-SSP *is described in Fig. 9.*

[3] The size of the keyword space is require to be at least $\omega(\log \lambda)$, where λ is the security parameter of the scheme, to guarantee security in an asymptotic sense.

Given that Search Pattern Privacy model is not enough to guarantee that any search pattern is completely hidden from an attacker, we propose a strictly stronger model denoted *Strong Search Pattern Privacy*. In this new model, the adversary chooses two lists list_0 and list_1, of the same length k, and receives as a challenge a list of k trapdoors. Both list_0 and list_1 contain bitstring-identifiers from $\{0,1\}^\star$. The challenger samples keywords to generate the trapdoors as a random oracle: for each *unique* identifier, a keyword is uniformly sampled from the keyword space. Thereby, the adversary is allowed to choose two search patterns, by defining equality relations between the encoded keywords, and receives trapdoors according to the pattern of either list_0 or list_1. Finally, the adversary is asked to determine if the trapdoors were generated according to pattern in list_0 or list_1.

Remark. If the adversary chooses to be challenged on $\mathsf{list}_0 = [0,0]$ and $\mathsf{list}_1 = [0,1]$ it is actually playing Game WEAK-SSP.

Remark. What Weak Search Pattern Privacy model guarantees is that search patterns are kept private as long as the same keyword is not search for more than twice.

Definition 12. *A PEKS scheme $\mathcal{E}$, associated with a non-polynomial size keyword space, has Strong Search Pattern Privacy if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{STRONG\text{-}SSP}}_{\mathcal{E},\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{STRONG\text{-}SSP}(\lambda) \Rightarrow \mathsf{True}] - 1,$$

where game STRONG-SSP *is described in Fig. 10.*

procedure Initialize(λ):
$(\mathsf{sk},\mathsf{pk}) \leftarrow_\$ \mathsf{KeyGen}(\lambda)$
$\mathsf{bit} \leftarrow_\$ \{0,1\}$
$\mathsf{w}_0 \leftarrow_\$ \mathcal{W}$
$\mathsf{w}_1 \leftarrow_\$ \mathcal{W}$
$\mathsf{tp}_0 \leftarrow_\$ \mathsf{Trapdoor}(\mathsf{pk},\mathsf{sk},\mathsf{w}_0)$
$\mathsf{tp}_1 \leftarrow_\$ \mathsf{Trapdoor}(\mathsf{pk},\mathsf{sk},\mathsf{w}_{\mathsf{bit}})$
return $(\mathsf{pk},\mathsf{tp}_0,\mathsf{tp}_1)$

procedure Trapdoor(w):
$\mathsf{tp} \leftarrow_\$ \mathsf{Trapdoor}(\mathsf{pk},\mathsf{sk},\mathsf{w})$
return tp

procedure Finalize(bit'):
return $(\mathsf{bit} = \mathsf{bit}')$

Fig. 9. Game WEAK-SSP [3].

procedure Initialize(λ):
$(\mathsf{sk},\mathsf{pk}) \leftarrow_\$ \mathsf{KeyGen}(\lambda)$
$\mathsf{bit} \leftarrow_\$ \{0,1\}$
$\mathsf{list}_\mathsf{w} \leftarrow []$
$\mathsf{list}_\mathsf{tp} \leftarrow []$
return pk

procedure Trapdoor(w):
$\mathsf{tp} \leftarrow_\$ \mathsf{Trapdoor}(\mathsf{sk},\mathsf{w})$
return tp

procedure Finalize(bit'):
return $(\mathsf{bit} = \mathsf{bit}')$

procedure Challenge$(\mathsf{list}_0,\mathsf{list}_1)$:
$\mathsf{k} \leftarrow |\mathsf{list}_0|$
for i in {0..k-1}
... get w for $\mathsf{list}_\mathsf{bit}$[i] from list_w
... if $\mathsf{w} = \perp$
... ... $\mathsf{w} \leftarrow_\$ \mathcal{W}$
... ... $\mathsf{list}_\mathsf{w} \leftarrow (\mathsf{list}_\mathsf{bit}[\mathsf{i}],\mathsf{w}) : \mathsf{list}_\mathsf{w}$
... $\mathsf{list}_\mathsf{tp}[\mathsf{i}] \leftarrow_\$ \mathsf{Trapdoor}(\mathsf{pk},\mathsf{sk},\mathsf{w})$
return $\mathsf{list}_\mathsf{tp}$

Fig. 10. Game STRONG-SSP. Adversary is legitimate if it only calls Challenge once with $|\mathsf{list}_0| = |\mathsf{list}_1|$.

Remark. Nishioka [3] also proposed the notion of IND-PKP security, where two trapdoors for either the same or different keywords, but generated under different secret keys, are given to the adversary as a challenge. This notion may be relevant in specific multi-user scenarios, however is out of the scope of our work.

3.2 Key Unlinkability for IBE

Similar to the requirements for keyword space in PEKS *Search Pattern Privacy* models, *Key Unlinkability* models for IBE require that the size of the identity space is at least $\omega(\log \lambda)$, where λ is the security parameter of the scheme.

Definition 13. *An IBE scheme Π, associated with a non-polynomial size identity space, has Weak Key Unlinkability if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{WEAK\text{-}KEY\text{-}UNLINK}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{WEAK\text{-}KEY\text{-}UNLINK}(\lambda) \Rightarrow \mathsf{True}] - 1,$$

where game WEAK-KEY-UNLINK *is described in Fig. 11.*

Definition 14. *An IBE scheme Π, associated with a non-polynomial size identity space, has Strong Key Unlinkability if, for every legitimate PPT adversary $\mathcal{A}$, the following definition of advantage is negligible in λ*

$$\mathbf{Adv}^{\mathsf{STRONG\text{-}KEY\text{-}UNLINK}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr[\mathsf{STRONG\text{-}KEY\text{-}UNLINK}(\lambda) \Rightarrow \mathsf{True}] - 1,$$

where game STRONG-KEY-UNLINK *is described in Fig. 12.*

procedure Initialize(λ):
(Msk, params) $\leftarrow_\$$ Setup(λ)
bit $\leftarrow_\$ \{0,1\}$
$\mathsf{id}_0 \leftarrow_\$ \mathcal{I}$
$\mathsf{id}_1 \leftarrow_\$ \mathcal{I}$
$\mathsf{sk}_0 \leftarrow_\$$ Extract(params, Msk, id_0)
$\mathsf{sk}_1 \leftarrow_\$$ Extract(params, Msk, $\mathsf{id}_{\mathsf{bit}}$)
return (params, sk_0, sk_1)

procedure Extract(id):
$\mathsf{sk}_{\mathsf{id}} \leftarrow_\$$ Extract(params, Msk, id)
return $\mathsf{sk}_{\mathsf{id}}$

procedure Finalize(bit′):
return (bit = bit′)

Fig. 11. Game WEAK-KEY-UNLINK.

procedure Initialize(λ):
(Msk, params) $\leftarrow_\$$ Setup(λ)
bit $\leftarrow_\$ \{0,1\}$
$\mathsf{list}_{\mathsf{id}} \leftarrow []$
$\mathsf{list}_{\mathsf{sk}} \leftarrow []$
return pk

procedure Extract(id):
sk $\leftarrow_\$$ Extract(params, Msk, id)
return tp

procedure Finalize(bit′):
return (bit = bit′)

procedure Challenge($\mathsf{list}_0, \mathsf{list}_1$):
k $\leftarrow |\mathsf{list}_0|$
for i in {0..k-1}
... get id for $\mathsf{list}_{\mathsf{bit}}[\mathsf{i}]$ from $\mathsf{list}_{\mathsf{id}}$
... if id $=\perp$
... ... id $\leftarrow_\$ \mathcal{I}$
... ... $\mathsf{list}_{\mathsf{id}} \leftarrow (\mathsf{list}_{\mathsf{bit}}[\mathsf{i}], \mathsf{id}) : \mathsf{list}_{\mathsf{id}}$
... $\mathsf{list}_{\mathsf{sk}}[\mathsf{i}] \leftarrow_\$$ Extract(params, Msk, id)
return $\mathsf{list}_{\mathsf{sk}}$

Fig. 12. Game STRONG-KEY-UNLINK. Adversary is legitimate if it only calls Challenge once with $|\mathsf{list}_0| = |\mathsf{list}_1|$.

Weak Key Unlinkability and *Strong Key Unlinkability* lead to *Search Pattern Privacy* and *Strong Search Pattern Privacy* for PEKS, respectively, after the black-box transformation from IBE to PEKS described in Section 2. Proofs follow directly from the transformation and are omitted here.

Lemma 1. *An IBE scheme Π with Weak Key Unlinkability leads to a PEKS scheme $\mathcal{E}$ with (Weak) Search Pattern Privacy, after the black-box transformation from IBE to PEKS proposed in [6].*

Lemma 2. *An IBE scheme Π with Strong Key Unlinkability leads to a PEKS scheme $\mathcal{E}$ with Strong Search Pattern Privacy, after the black-box transformation from IBE to PEKS proposed in [6].*

3.3 Function Privacy for IBE: an independent security notion

Very recently, Boneh, Raghunathan and Segev [4] put forward two security notions, of different strength, for IBE, inspired by the security definition given for deterministic encryption in [7]: *Function Privacy* and *Enhanced Function Privacy*. These notions ask that "decryption keys reveal essentially no information on their corresponding identities, beyond the absolute minimum necessary". In both definitions, the adversary is first given the public parameters and then interacts with a Real-or-Random function privacy oracle, which takes as input an adversarially-chosen joint probability distribution – represented as a circuit – for random variables $\mathsf{X}_1, \mathsf{X}_2, ..., \mathsf{X}_\mathsf{L}$ defined over the identity space $\mathcal{I}$, and outputs L secret keys either for identities sampled from the given joint probability distribution or for independent and uniformly distributed identities over $\mathcal{I}$.
An adversary is legitimate if, for every $\mathsf{i} \in \{1..\mathsf{L}\}$ and every $\mathsf{x}_1, ..., \mathsf{x}_\mathsf{i} \in \mathcal{I}$, it holds that: $\mathbf{H}_\infty(\mathsf{X}_\mathsf{i}|_{\mathsf{X}_1=\mathsf{x}_1, ..., \mathsf{X}_{\mathsf{i}-1}=\mathsf{x}_{\mathsf{i}-1}}) = -\log(\max \Pr[\mathsf{X}_\mathsf{i} = \mathsf{x}_\mathsf{i}|_{\mathsf{X}_1=\mathsf{x}_1,...,\mathsf{X}_{\mathsf{i}-1}=\mathsf{x}_{\mathsf{i}-1}}]) \geq \omega(\log \lambda)$. Put differently, the chosen joint probability distribution for $(\mathsf{X}_1, ..., \mathsf{X}_\mathsf{L})$ has to be such that every random variable X_i is *sufficiently unpredictable*, even if every random variable $\mathsf{X}_{\mathsf{j}<\mathsf{i}}$ has been fixed. To discard exhaustive searches, a *conditional min-entropy* $\mathbf{H}_\infty(\mathsf{X}_\mathsf{i}|_{\mathsf{X}_1=\mathsf{x}_1,...,\mathsf{X}_{\mathsf{i}-1}})$ of at least $\omega(\log \lambda)$ bits is required[4]. The *Enhanced* model provides the adversary with an extra function-privacy encryption oracle capable of encrypting adversarially-chosen messages under the identities sampled by Real-or-Random oracle.

[4] The *minimal* unpredictability requirement of $\omega(\log \lambda)$ bits has only been achieve later in [5]. Schemes in [4] have only been proven secure for highly unpredictable identities with min-entropy of $\lambda + \omega(\log \lambda)$.

We first remark that Key Unlinkability and Function Privacy security models are essentially different in the way the challenger samples ids: in the former ids are sampled uniformly from the id space, whereas in the latter model ids may be sampled from an adversarial-chosen joint probability distribution, with (possibly) non-uniform random variables, but also high min-entropy requirements. In the following subsections we provide counterexamples to show that Function Privacy (both *Non-enhanced* and *Enhanced*) and Key Unlinkability (both *Weak* and *Strong*) are independent security notions. Meaningful counterexamples follow. For a quick overview, Figure 13 states the relations between Weak Key Unlinkability, Strong Key Unlinkability, Function Privacy and Enhanced Function Privacy security notions.

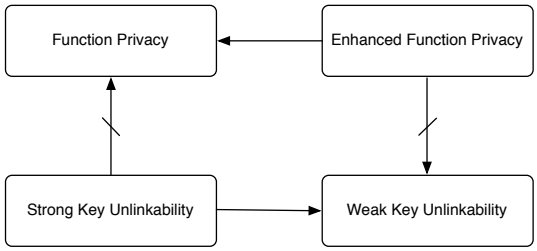

Fig. 13. Relations between Key Unlinkability and Function Privacy security notions.

We stress that even Enhanced Function Privacy fails to capture the security guarantees of Weak Key Unlinkability. In practice, transforming an anonymous IBE with Enhanced Function Privacy to PEKS (according to the transformation described in Section 2) results in no guarantee that the service provider will not be able to find search patterns in the users' trapdoors.

Enhanced Function Privacy $\not\Rightarrow$ Weak Key Unlinkability. More generally, we can even show that an enhanced function-private IBE is not necessarily weak key-unlinkable. Consider $F : \{0,1\}^\lambda \times \mathcal{I} \to \{0,1\}^\lambda$ to be a secure PRF. We denote by $f \leftarrow_\$ F$ the operation: $\mathsf{k} \leftarrow_\$ \{0,1\}^\lambda; f \leftarrow F(\mathsf{k}, \cdot)$. Let $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be an enhanced function-private IBE. From Π we can construct Π', where Π' is still enhanced function-private but definitely not weak key-unlinkable. We do so by simply modifying the extraction algorithm and appending to each secret key the result of a PRF on id. More precisely, $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Enc}', \mathsf{Dec}')$ is constructed as follows:

- $\mathsf{Setup}'(\lambda) : (\mathsf{Msk}, \mathsf{params}) \leftarrow_\$ \mathsf{Setup}(\lambda)$; $f \leftarrow_\$ F$; return $((\mathsf{Msk}, f), \mathsf{params})$.
- $\mathsf{Extract}'(\mathsf{Msk}, \mathsf{id}) : \mathsf{sk} \leftarrow_\$ \mathsf{Extract}(\mathsf{Msk}, \mathsf{id}); \mathsf{sk}' \leftarrow (\mathsf{sk}, f(\mathsf{id}))$; return sk'.
- $\mathsf{Enc}'(\mathsf{params}, \mathsf{m}, \mathsf{id}) : \mathsf{c} \leftarrow_\$ \mathsf{Enc}(\mathsf{params}, \mathsf{m}, \mathsf{id})$; return c.
- $\mathsf{Dec}'(\mathsf{params}, \mathsf{c}, \mathsf{id}, \mathsf{sk}') : (\mathsf{sk}, \mathsf{y}) \leftarrow \mathsf{sk}'$; $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{params}, \mathsf{c}, \mathsf{id}, \mathsf{sk})$; return m.

Informally, since f is unknown to the adversary, the adversary cannot choose distributions depending on f. Furthermore, F is a secure PRF, so no information on id can be acquired. Therefore, Π' is still an enhanced function-private IBE. But, because f is deterministic, it is trivial to identify with overwhelming probability if two keys have been extracted from the same identity.

Strong Key Unlinkability $\not\Rightarrow$ Function Privacy. Again, we show this by counterexample. Let $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be a strong key-unlinkable IBE associated with id space $\mathcal{I} = \{0,1\}^{2\lambda}$. We build $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Enc}', \mathsf{Dec}')$ based on Π as follows:

- $\mathsf{Setup}'(\lambda) : (\mathsf{Msk}, \mathsf{params}) \leftarrow_\$ \mathsf{Setup}(\lambda)$; return $(\mathsf{Msk}, \mathsf{params})$.
- $\mathsf{Extract}'(\mathsf{Msk}, \mathsf{id}) : \mathsf{sk} \leftarrow_\$ \mathsf{Extract}(\mathsf{Msk}, \mathsf{id})$; if $\mathsf{id} \in \{0,1\}^\lambda 0^\lambda$ then $\mathsf{sk}' \leftarrow (\mathsf{sk}||0)$ else $\mathsf{sk}' \leftarrow (\mathsf{sk}||1)$; return sk'.
- $\mathsf{Enc}'(\mathsf{params}, \mathsf{m}, \mathsf{id}) : \mathsf{c} \leftarrow_\$ \mathsf{Enc}(\mathsf{params}, \mathsf{m}, \mathsf{id})$ return c.
- $\mathsf{Dec}'(\mathsf{params}, \mathsf{c}, \mathsf{id}, \mathsf{sk}') : (\mathsf{sk}||\mathsf{b}) \leftarrow \mathsf{sk}'; \mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{params}, \mathsf{c}, \mathsf{id}, \mathsf{sk})$; return m.

In our counterexample scheme Π', we put a mark in keys for identities whose last λ bits are 0, by appending a 0 to the key (otherwise, 1 is appended). Since the subset containing these identities – let us call it $\mathcal{U}$ – is much smaller than the identity space $\mathcal{I}$, identities uniformly sampled from

$\mathcal{I}$ are very unlikely to be in $\mathcal{U}$, and thus to possess the mark. In fact, this only happens with probability $\Pr = \frac{2^\lambda}{2^{2\lambda}} = \frac{1}{2^\lambda}$, which is a negligible function in the security parameter λ. Strong Key Unlinkability is therefore preserved in Π'. However, $\mathcal{U}$ is big enough so that the unpredictability of an id uniformly sampled from $\mathcal{U}$ is high. By choosing to be challenged on a random variable X that selects any element in $\mathcal{U}$ with probability $\frac{1}{2^\lambda}$ and any element in $\{\mathsf{x} \in \mathcal{I} : \mathsf{x} \notin \mathcal{U}\}$ with zero probability, an adversary could trivially win the function-privacy game, with overwhelming probability, just by looking into the key's mark. Also notice that the condition $\mathbf{H}_\infty(\mathsf{X}) > \omega(\log \lambda)$ is satisfied. Generically, we can conclude that a strong key-unlinkable scheme is not necessarily function-private secure.

4 From Weak to Strong Key Unlinkability

A scheme with Weak Key Unlinkability. We construct a new anonymous IBE scheme with Weak Key Unlinkability, based on the anonymous IBE scheme of Boyen and Waters [8]. Our scheme relies on a bilinear group description Γ of prime order. To eliminate the selective-ID constraint, we adopt the random oracle approach introduced in the full paper of [14], which replaces identities with their hash values. Furthermore, we simplify the resulted scheme by removing two group elements from the public parameters and from private keys, and obtain the final scheme in Fig. 14. Compared with the original scheme, our scheme also saves two exponentiations in the key-extraction and encryption algorithms, and saves two pairing computations in the decryption algorithm. Our scheme preserves *anonymity* and *semantic security* proprieties, provided that the hash function $\mathsf{H} : \mathcal{I} \to \mathbb{G}$ is modeled as a random oracle. Added to this, the scheme also has the Weak Key Unlinkability property.

$\underline{\mathsf{Setup}(\lambda)}$:	$\underline{\mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id})}$:	$\underline{\mathsf{Enc}(\mathsf{params}, \mathsf{m}, \mathsf{id})}$:	$\underline{\mathsf{Dec}(\mathsf{params}, \mathsf{c}, \mathsf{id}, \mathsf{sk}_{\mathsf{id}})}$:
$\Gamma \leftarrow_\$ \mathcal{G}_{\mathcal{P}}(\lambda)$	$\mathsf{r} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$	$\mathsf{s}, \mathsf{s}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}^2$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$
$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$(\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2) \leftarrow \mathsf{Msk}$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$
$\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}^3$	$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2) \leftarrow \mathsf{sk}_{\mathsf{id}}$
$\Omega \leftarrow \mathbf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}}$	$(\mathsf{p}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow \Gamma$	$\mathsf{h} \leftarrow \mathsf{H}(\mathsf{id})$	$(\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2) \leftarrow \mathsf{c}$
$\mathsf{v}_1 \leftarrow \mathsf{g}^{\mathsf{t}_1}$	$\mathsf{h} \leftarrow \mathsf{H}(\mathsf{id})$	$\hat{\mathsf{c}} \leftarrow \Omega^\mathsf{s}\mathsf{m}$	$\mathsf{e}_0 \leftarrow \mathbf{e}(\mathsf{c}_0, \mathsf{d}_0)$
$\mathsf{v}_2 \leftarrow \mathsf{g}^{\mathsf{t}_2}$	$\mathsf{d}_0 \leftarrow \mathsf{g}^{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}$	$\mathsf{c}_0 \leftarrow \mathsf{h}^\mathsf{s}$	$\mathsf{e}_1 \leftarrow \mathbf{e}(\mathsf{c}_1, \mathsf{d}_1)$
$\mathsf{params} \leftarrow (\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2)$	$\mathsf{d}_1 \leftarrow \mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_2}$	$\mathsf{c}_1 \leftarrow \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1}$	$\mathsf{e}_2 \leftarrow \mathbf{e}(\mathsf{c}_2, \mathsf{d}_2)$
$\mathsf{Msk} \leftarrow (\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2)$	$\mathsf{d}_2 \leftarrow \mathsf{g}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_1}$	$\mathsf{c}_2 \leftarrow \mathsf{v}_2^{\mathsf{s}_1}$	$\mathsf{m} \leftarrow \hat{\mathsf{c}} \cdot \mathsf{e}_0 \cdot \mathsf{e}_1 \cdot \mathsf{e}_2$
return $(\mathsf{Msk}, \mathsf{params})$	$\mathsf{sk}_{\mathsf{id}} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$	$\mathsf{c} \leftarrow (\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2)$	return m
	return $\mathsf{sk}_{\mathsf{id}}$	return c	

Fig. 14. Anonymous IBE scheme Π with Weak Key Unlinkability.

Theorem 1 (Appendix A). *IBE scheme Π [Fig. 14] is semantically secure [Definition 7], in the random oracle model, assuming DBDH is intractable [Definition 2].*

Theorem 2 (Appendix B). *IBE scheme Π [Fig. 14] is anonymous [Definition 8], in the random oracle model, assuming DBDH and DLIN are intractable [Definitions 2 and 3].*

Theorem 3 (Appendix C). *IBE scheme Π [Fig. 14] has the Weak Key Unlinkability property [Definition 13], in the random oracle model, assuming DLIN is intractable [Definition 3].*

Weak Key Unlinkability $\not\Rightarrow$ Strong Key Unlinkability. Standard real-or-random definitions for public-key encryption model the encryption of a single plaintext. These definitions are equivalent (with some loss in tightness) to those allowing an adversary to acquire multiple encryptions, which can be shown by applying the hybrid argument from [15]. One might be tempted to think that the same hybrid argument also applies to Weak Key Unlinkability model. However, this argument does *not* apply, since we can show that an adversary can still easily distinguish patterns when more than two keys are issued with scheme Π [Fig. 14].

Suppose that an adversary is asked to distinguished between tuples of the form $(\mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_0))$, where the three secret keys are extracted from the same id, from those of the form $(\mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_1))$, where the third key is extracted from an independent id, for uniformly sampled id_0 and $\mathsf{id}_1 \in \mathcal{I}$. Let $(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2)$ be the tuple the adversary

receives, and for which it has to decide its form. We further expand sk_i to $(\mathsf{d}_{i0}, \mathsf{d}_{i1}, \mathsf{d}_{i2})$ according to our scheme. If the keys were generated honestly, i.e. by following the algorithm Extract as described in Fig. 14, the adversary simply has to check if

$$\mathbf{e}(\frac{\mathsf{d}_{10}}{\mathsf{d}_{00}}, \frac{\mathsf{d}_{21}}{\mathsf{d}_{01}}) \stackrel{?}{=} \mathbf{e}(\frac{\mathsf{d}_{00}}{\mathsf{d}_{20}}, \frac{\mathsf{d}_{01}}{\mathsf{d}_{11}})$$

to determine the form of the tuple with overwhelming probability. If the result from the equality is true, then the three secret keys are very likely to have been extracted for the same id[5]. If the result is false, then the tuple is definitely of the form $(\mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_0), \mathsf{Extract}(\mathsf{id}_1))$. For completeness, we show this by expanding and simplifying the above expression.

$$\begin{aligned}
\mathbf{e}(\frac{\mathsf{d}_{10}}{\mathsf{d}_{00}}, \frac{\mathsf{d}_{21}}{\mathsf{d}_{01}}) &= \mathbf{e}(\frac{\mathsf{d}_{00}}{\mathsf{d}_{20}}, \frac{\mathsf{d}_{01}}{\mathsf{d}_{11}}) \Leftrightarrow \\
\mathbf{e}(\frac{\mathsf{g}^{\mathsf{r}_1\mathsf{t}_1\mathsf{t}_2}}{\mathsf{g}^{\mathsf{r}_0\mathsf{t}_1\mathsf{t}_2}}, \frac{\mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}_2^{-\mathsf{r}_2\mathsf{t}_2}}{\mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}_0^{-\mathsf{r}_0\mathsf{t}_2}}) &= \mathbf{e}(\frac{\mathsf{g}^{\mathsf{r}_0\mathsf{t}_1\mathsf{t}_2}}{\mathsf{g}^{\mathsf{r}_2\mathsf{t}_1\mathsf{t}_2}}, \frac{\mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}_0^{-\mathsf{r}_0\mathsf{t}_2}}{\mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}_1^{-\mathsf{r}_1\mathsf{t}_2}}) \Leftrightarrow \\
\mathbf{e}(\frac{\mathsf{g}^{\mathsf{r}_1\mathsf{t}_1\mathsf{t}_2}}{\mathsf{g}^{\mathsf{r}_0\mathsf{t}_1\mathsf{t}_2}}, \frac{\mathsf{h}_2^{-\mathsf{r}_2\mathsf{t}_2}}{\mathsf{h}_0^{-\mathsf{r}_0\mathsf{t}_2}}) &= \mathbf{e}(\frac{\mathsf{g}^{\mathsf{r}_0\mathsf{t}_1\mathsf{t}_2}}{\mathsf{g}^{\mathsf{r}_2\mathsf{t}_1\mathsf{t}_2}}, \frac{\mathsf{h}_0^{-\mathsf{r}_0\mathsf{t}_2}}{\mathsf{h}_0^{-\mathsf{r}_1\mathsf{t}_2}}) \Leftrightarrow \\
\mathbf{e}(\mathsf{g}^{(\mathsf{r}_1-\mathsf{r}_0)}, \mathsf{h}_0^{\mathsf{r}_0} \cdot \mathsf{h}_2^{-\mathsf{r}_2})^{\mathsf{t}_1(\mathsf{t}_2)^2} &= \mathbf{e}(\mathsf{g}^{(\mathsf{r}_0-\mathsf{r}_2)}, \mathsf{h}_0^{(\mathsf{r}_1-\mathsf{r}_0)})^{\mathsf{t}_1(\mathsf{t}_2)^2} \Leftrightarrow \\
\mathbf{e}(\mathsf{g}, \mathsf{h}_0^{\mathsf{r}_0} \cdot \mathsf{h}_2^{-\mathsf{r}_2}) &= \mathbf{e}(\mathsf{g}, \mathsf{h}_0^{(\mathsf{r}_0-\mathsf{r}_2)}) \Leftrightarrow \\
\mathsf{h}_2 &= \mathsf{h}_0
\end{aligned}$$

It is now clear that IBE scheme Π [Fig. 14] fails to achieve the Strong Key Unlinkability property.

A scheme with Strong Key Unlinkability. We extend Π to groups of composite order and obtain Π' [Fig. 15]. The extension is very simple: let all the parameters in the original scheme be from the subgroup $\mathbb{G}_\mathsf{p}$ (generated by g_p) and randomize each element of the extracted secret key by a random element from the subgroup $\mathbb{G}_\mathsf{q}$ (generated by g_q). Note that the message space is $\mathbb{G}_\mathsf{T}$.

Setup(1^λ):
$(\mathsf{p}, \mathsf{q}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}) \leftarrow_\$ \mathcal{G}_\mathcal{C}(\lambda)$
$\mathsf{n} \leftarrow \mathsf{pq}$; $\mathsf{g}_\mathsf{p} \leftarrow \mathsf{g}^\mathsf{q}$; $\mathsf{g}_\mathsf{q} \leftarrow \mathsf{g}^\mathsf{p}$
$\Gamma \leftarrow (\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q})$
$\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{n}$
$\Omega \leftarrow \mathbf{e}(\mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{p})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}}$
$\mathsf{v}_1 \leftarrow \mathsf{g}_\mathsf{p}^{\mathsf{t}_1}$
$\mathsf{v}_2 \leftarrow \mathsf{g}_\mathsf{p}^{\mathsf{t}_2}$
$\mathsf{params} \leftarrow (\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2)$
$\mathsf{Msk} \leftarrow (\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2)$
return $(\mathsf{Msk}, \mathsf{params})$

Extract(params, Msk, id):
$(\mathsf{w}, \mathsf{t}_1, \mathsf{t}_2) \leftarrow \mathsf{Msk}$
$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$
$(\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q}) \leftarrow \Gamma$
$\mathsf{r} \leftarrow_\$ \mathbb{Z}_\mathsf{n}$
$\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2 \leftarrow_\$ \mathbb{G}_\mathsf{q}$
$\mathsf{h} \leftarrow \mathsf{H}(\mathsf{id})$
$\mathsf{d}_0 \leftarrow \mathsf{x}_0 \cdot \mathsf{g}_\mathsf{p}^{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}$
$\mathsf{d}_1 \leftarrow \mathsf{x}_1 \cdot \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_2}$
$\mathsf{d}_2 \leftarrow \mathsf{x}_2 \cdot \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_1}$
$\mathsf{sk} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
return sk

Enc(params, m, id):
$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$
$(\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q}) \leftarrow \Gamma$
$\mathsf{s}, \mathsf{s}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{n}$
$\mathsf{h} \leftarrow \mathsf{H}(\mathsf{id})$
$\hat{\mathsf{c}} \leftarrow \Omega^\mathsf{s}\mathsf{m}$
$\mathsf{c}_0 \leftarrow \mathsf{h}^\mathsf{s}$
$\mathsf{c}_1 \leftarrow \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1}$
$\mathsf{c}_2 \leftarrow \mathsf{v}_2^{\mathsf{s}_1}$
$\mathsf{c} \leftarrow (\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2)$
return c

Dec(params, c, id, sk_id):
$(\Gamma, \Omega, \mathsf{v}_1, \mathsf{v}_2) \leftarrow \mathsf{params}$
$(\mathsf{n}, \mathbb{G}, \mathbb{G}_\mathsf{T}, \mathbf{e}, \mathsf{g}, \mathsf{g}_\mathsf{p}, \mathsf{g}_\mathsf{q}) \leftarrow \Gamma$
$(\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2) \leftarrow \mathsf{sk}_\mathsf{id}$
$(\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2) \leftarrow \mathsf{c}$
$\mathsf{e}_0 \leftarrow \mathbf{e}(\mathsf{c}_0, \mathsf{d}_0)$
$\mathsf{e}_1 \leftarrow \mathbf{e}(\mathsf{c}_1, \mathsf{d}_1)$
$\mathsf{e}_2 \leftarrow \mathbf{e}(\mathsf{c}_2, \mathsf{d}_2)$
$\mathsf{m} \leftarrow \hat{\mathsf{c}} \cdot \mathsf{e}_0 \cdot \mathsf{e}_1 \cdot \mathsf{e}_2$
return m

Fig. 15. Anonymous IBE scheme Π' with Strong Key Unlinkability.

The decryption algorithm remains correct, since

$$\begin{aligned}
&\mathbf{e}(\mathsf{h}^\mathsf{s}, \mathsf{x}_0 \cdot \mathsf{g}_\mathsf{p}^{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}) = \mathbf{e}(\mathsf{h}^\mathsf{s}, \mathsf{g}_\mathsf{p}^{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}) \\
&\mathbf{e}(\mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1}, \mathsf{x}_1 \cdot \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_2}) = \mathbf{e}(\mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1}, \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_2}) \\
&\mathbf{e}(\mathsf{v}_2^{\mathsf{s}_1}, \mathsf{x}_2 \cdot \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_1}) = \mathbf{e}(\mathsf{v}_2^{\mathsf{s}_1}, \mathsf{g}_\mathsf{p}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}\mathsf{t}_1})
\end{aligned}$$

Also, *semantic security* and *anonymity* properties are not affected, assuming DBDH and DLIN hold in $\mathbb{G}_\mathsf{p}$. We only need to prove that Π' possesses the *Strong Key Unlinkability* property.

Theorem 4 (Appendix D). *IBE scheme Π' [Fig. 15] has Strong Key Unlinkability [Definition 14], assuming CDDH is intractable [Definition 5].*

[5] Collisions in the hash function H may lead to misleading results but only occur with negligible probability.

5 Conclusions and Future Directions

Our work shows that two distinct scenarios have to be considered to model trapdoor privacy: one in the presence of ciphertexts that match trapdoors, and the other in the absence of such ciphertexts. The notion of Strong Search Pattern Privacy we introduced here addresses privacy concerns up to the point there is ciphertexts matching the issued trapdoors, after which, search patterns can no longer be hidden from an attacker. Previous models provide limited privacy guarantees against search patterns.

The full version of [4] proposes a generic method for transforming any IBE scheme into an IBE scheme which achieves a *weaker* form of Enhanced Function Privacy, where the adversary is not allowed to chose distributions (from which identities are sampled for the challenge) that depend on the public parameters of the scheme. This relaxation results in a definition denoted *Non-Adaptive Enhanced Function Privacy*. In practice, for PEKS, it seems reasonable to assume that keywords will not depend on the public parameters of the scheme, and, in particular, on the values output by the random oracle. So, from any anonymous IBE, using the black-box transformation described in Section 2, it is easy to obtain a PEKS scheme with *reasonable* trapdoor privacy guarantees for the scenario where the adversary possesses ciphertexts that positively match the trapdoors. A *non-adaptive* version of *non-enhanced* Function Privacy could be defined as well. We remark that in the random oracle model, our notion of Strong Key Unlinkability implies the notion of *Non-adaptive Function Privacy*, if keywords are hashed before used in the scheme.

In our Strong Search Pattern Privacy model, keywords are uniformly sampled from the keyword space. Instead, we could have adopted the strategy of [4] and allowed the adversary to specify a joint probability distribution from which identities are sampled, forcing only min-entropy requirements on each random variable individually. It becomes clear that there are two ways to strengthen the Function Privacy model in [4]: either by providing to the adversary a function-privacy encryption oracle as in model Enhanced Function Privacy [4], or by relaxing the unpredictability requirements on the joint probability distribution from *conditional min-entropy* to *min-entropy*, on each random variable. The minimal requirement is that $\mathbf{H}_{\infty}(\mathsf{X}_{\mathsf{i}}) \geq \omega(\log \lambda)$, for each random variable X_{i} in the adversarial-chosen joint probability distribution. It remains an open problem to prove if our scheme Π' [Fig. 15] (or any other) can achieve security according to this generalized definition of Strong Key Unlinkability.

References

1. Boneh, D., Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology – EUROCRYPT '04. Volume 3027 of LNCS., Springer-Verlag (2004) 506–522
2. Tang, Q.: Search in Encrypted Data: Theoretical Models and Practical Applications. In: Theory and Practice of Cryptography Solutions for Secure Information Systems. IGI (2013) 84–108
3. Nishioka, M.: Perfect keyword privacy in PEKS systems. In: Provable Security – ProvSec '12. Volume 7496 of LNCS., Springer-Verlag (2012) 175–192
4. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: Hiding the function in functional encryption. In: Advances in Cryptology – CRYPTO '13. Volume 8043 of LNCS., Springer-Verlag (2013) 461–478
5. Boneh, D., Raghunathan, A., Segev, G.: Function-private subspace-membership encryption and its applications. To appear in *Advances in Cryptology – ASIACRYPT'13*. Available as Cryptology ePrint Archive, Report 2013/403 (2013)
6. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. Journal of Cryptology **21**(3) (2008) 350–391
7. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Advances in Cryptology – CRYPTO '07. Volume 4622 of LNCS., Springer-Verlag (2007) 535–552
8. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Advances in Cryptology – CRYPTO '06. Volume 4117 of LNCS., Springer-Verlag (2006) 290–307
9. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Theory of Cryptography Conference – TCC '07. Volume 4392 of LNCS., Springer-Verlag (2007) 535–554
10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Advances in Cryptology – EUROCRYPT '06. Volume 4004 of LNCS., Springer-Verlag (2006) 409–426
11. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in Cryptology – CRYPTO '01. Volume 2139 of LNCS., Springer-Verlag (2001) 213–229
12. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology – CRYPTO '04. Volume 3152 of LNCS., Springer-Verlag (2004) 41–55
13. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Theory of Cryptography Conference – TCC '05. Volume 3378 of LNCS., Springer-Verlag (2005) 325–341
14. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Advances in Cryptology – EUROCRYPT '04. Volume 3027 of LNCS., Springer-Verlag (2004) 223–238
15. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Advances in Cryptology – EUROCRYPT '00. Volume 1807 of LNCS., Springer-Verlag (2000) 259–274

A Proof of Theorem 1

Game_0 is game $\mathsf{IBE\text{-}IND\text{-}CPA}_{\Pi,\mathcal{A}}$ [Fig. 5], where $\mathcal{A}$ is any legitimate PPT adversary. In Game_1, we set $\hat{c} \leftarrow_{\$} \mathbb{G}_{\mathsf{T}}$ instead of computing its value. We now show that the existence of $\mathcal{A}$ that can successfully distinguish between Game_0 and Game_1 contradicts the DBDH assumption. More precisely, we construct a simulator $\mathcal{S}_0$ [Fig. 16] that interpolates between Game_0 and Game_1 by playing game $\mathsf{DBDH}_{\Gamma,\mathcal{S}_0}$ [Fig. 1]. The hash function H is modelled as a random oracle and we assume, without loss of generality, that $\mathcal{A}$ always asks for the hash value of id before querying id to oracles Extract and Real-or-Random. Furthermore, for simplicity of exposition, we assume that $\mathcal{A}$ places *exactly* q queries, where q is bounded by some polynomial in the security parameter λ, since $\mathcal{A}$ is required to run in polynomial-time. Simulator $\mathcal{S}_0$ randomly tries to guess which query $\mathsf{i} \in \{0, \mathsf{q}\text{-}1\}$ contains the id on which adversary $\mathcal{A}$ asks to be challenged. When i is *not* sucessfully guessed, $\mathcal{S}_0$ simply aborts. But when it is, which happens with probability $\frac{1}{\mathsf{q}}$, $\mathcal{S}_0$ perfectly simulates Game_0 if Z is of the form $\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3}$, and perfectly simulates Game_1 if Z is just a random element in $\mathbb{G}_{\mathsf{T}}$. Notice that this implies $\mathsf{w} = \mathsf{z}_1\mathsf{z}_2$ and $\mathsf{s} = \mathsf{z}_3$.

Let $\mathsf{id}^\star$ be the id chosen by $\mathcal{A}$ for the challenge. Random function H is consistently computed: if queried twice on the same id, the same result is returned. When $\mathcal{S}_0$ successfully completes its simulation, the function is set to $(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}$ for every id but $\mathsf{id}^\star$, and to g^{x} in this particular case, where x is a random value sampled from $\mathbb{Z}_{\mathsf{p}}$.

Challenge is well formed, as well as secret keys for random exponents $\mathsf{r}' = \frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}$, where x here is the value used to compute the hash of the corresponding id and r is sampled from $\mathbb{Z}_\mathsf{p}$. For completeness, we present the equalities between the original expressions and those computed by the simulator[6].

$$\mathsf{d}_0 = \mathsf{g}^{\mathsf{r}'\mathsf{t}_1\mathsf{t}_2} = \mathsf{g}^{\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_1\mathsf{t}_2} = \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{g}^{\frac{-\mathsf{z}_2\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} = \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{Z}_2^{\frac{-\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}}$$
$$\mathsf{d}_1 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_2} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_2} = \mathsf{g}^{-\mathsf{z}_1\mathsf{z}_2\mathsf{t}_2} \cdot \left[(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}\right]^{-\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_2} = \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_2}$$
$$\mathsf{d}_2 = \mathsf{g}^{-\mathsf{w}\mathsf{t}_1} \cdot \mathsf{h}^{-\mathsf{r}'\mathsf{t}_1} = \mathsf{g}^{-\mathsf{z}_1\mathsf{z}_2\mathsf{t}_1} \cdot \left[(\mathsf{g}^{\mathsf{z}_1})^{\mathsf{x}}\right]^{-\frac{\mathsf{r}-\mathsf{z}_2}{\mathsf{x}}\mathsf{t}_1} = \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_1}$$

$$\hat{\mathsf{c}} = \Omega^{\mathsf{s}} \cdot \mathsf{m} = \left[\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{w}}\right]^{\mathsf{s}} \cdot \mathsf{m} = \left[\mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{t}_1\mathsf{t}_2\mathsf{z}_1\mathsf{z}_2}\right]^{\mathsf{z}_3} \cdot \mathsf{m} = \mathsf{Z}^{\mathsf{t}_1\mathsf{t}_2} \cdot \mathsf{m}$$
$$\mathsf{c}_0 = \mathsf{h}^{\mathsf{s}} = (\mathsf{g}^{\mathsf{x}})^{\mathsf{z}_3} = \mathsf{Z}_3^{\mathsf{x}}$$
$$\mathsf{c}_1 = \mathsf{v}_1^{\mathsf{s}-\mathsf{s}_1} = (\mathsf{g}_1^{\mathsf{t}})^{\mathsf{z}_3-\mathsf{s}_1} = (\mathsf{Z}_3 \cdot g^{-\mathsf{s}_1})^{\mathsf{t}_1}$$
$$\mathsf{c}_2 = \mathsf{v}_2^{\mathsf{s}_1}$$

In Game_1, the challenge ciphertext does not depend on $\mathsf{m}_{\mathsf{bit}}$. Therefore, we conclude that $\mathbf{Adv}^{\mathsf{IBE\text{-}IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda) = \frac{1}{\mathsf{q}} \cdot \mathbf{Adv}^{\mathsf{DBDH}}_{\Gamma,\mathcal{S}_0}$. □

procedure Initialize(λ):
$(\mathsf{Z}_1, \mathsf{Z}_2, \mathsf{Z}_3, \mathsf{Z}) \leftarrow \mathsf{DBDH.Initialize}$
$\mathsf{t}_1, \mathsf{t}_2 \leftarrow_\$ \mathbb{Z}_\mathsf{p}^2$
$\Omega \leftarrow \mathbf{e}(\mathsf{Z}_1, \mathsf{Z}_2)^{\mathsf{t}_1\mathsf{t}_2}$
$\mathsf{v}_1 \leftarrow \mathsf{g}^{\mathsf{t}_1}$
$\mathsf{v}_2 \leftarrow \mathsf{g}^{\mathsf{t}_2}$
$\mathsf{params} \leftarrow (\Omega, \mathsf{v}_1, \mathsf{v}_2)$
$\mathsf{list}_{\mathsf{ID}} \leftarrow [\,]$, $\mathsf{list}_{\mathsf{H}} \leftarrow [\,]$
$\mathsf{id}^\star \leftarrow \mathsf{null}$, $\mathsf{i} \leftarrow_\$ \{0, \mathsf{q}\text{-}1\}$
return params

procedure Extract(id):
if $\mathsf{id} == \mathsf{id}^\star$ return $\perp$
if id is in $\mathsf{list}_{\mathsf{H}}$[i] abort
get x for id from $\mathsf{list}_{\mathsf{H}}$
$\mathsf{r} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$

$\mathsf{d}_0 \leftarrow \mathsf{g}^{\frac{\mathsf{r}\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}} \cdot \mathsf{Z}_2^{\frac{-\mathsf{t}_1\mathsf{t}_2}{\mathsf{x}}}$
$\mathsf{d}_1 \leftarrow \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_2}$
$\mathsf{d}_2 \leftarrow \mathsf{Z}_1^{-\mathsf{r}\mathsf{t}_1}$

$\mathsf{sk}_{\mathsf{id}} \leftarrow (\mathsf{d}_0, \mathsf{d}_1, \mathsf{d}_2)$
$\mathsf{list}_{\mathsf{ID}} \leftarrow \mathsf{id} : \mathsf{list}_{\mathsf{ID}}$
return $\mathsf{sk}_{\mathsf{id}}$

procedure Finalize(bit):
return

procedure H(id):
get x for id from $\mathsf{list}_{\mathsf{H}}$
if $\mathsf{x} == \perp$
..... $\mathsf{x} \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
..... $\mathsf{list}_{\mathsf{H}} \leftarrow (\mathsf{id}, \mathsf{x}) : \mathsf{list}_{\mathsf{H}}$
if id is in $\mathsf{list}_{\mathsf{H}}$[i] then $\mathsf{h} \leftarrow \mathsf{g}^{\mathsf{x}}$ else $\mathsf{h} \leftarrow \mathsf{Z}_1^{\mathsf{x}}$
return h

procedure Real-or-Random($\mathsf{id}^\star$, m):
if $\mathsf{id}^\star \in \mathsf{list}_{\mathsf{ID}}$ return $\perp$
if $\mathsf{id}^\star$ is not in $\mathsf{list}_{\mathsf{H}}$[i] abort

$\mathsf{s}_1 \leftarrow_\$ \mathbb{Z}_\mathsf{p}$
get x for $\mathsf{id}^\star$ from $\mathsf{list}_{\mathsf{H}}$

$\hat{\mathsf{c}} \leftarrow \mathsf{Z}^{\mathsf{t}_1\mathsf{t}_2} \cdot \mathsf{m}$
$\mathsf{c}_0 \leftarrow \mathsf{Z}_3^{\mathsf{x}}$
$\mathsf{c}_1 \leftarrow \mathsf{Z}_3^{\mathsf{t}_1} \cdot \mathsf{g}^{-\mathsf{t}_1\mathsf{s}_1}$
$\mathsf{c}_2 \leftarrow \mathsf{v}_2^{\mathsf{s}_1}$

$\mathsf{c} \leftarrow (\hat{\mathsf{c}}, \mathsf{c}_0, \mathsf{c}_1, \mathsf{c}_2)$
return c

Fig. 16. Simulator $\mathcal{S}_0$ interpolates between Game_0 and Game_1 by playing game $\mathsf{DBDH}_{\Gamma,\mathcal{S}_0}$.

B Proof of Theorem 2

By applying the reduction of Theorem 1, we start with Game_1, which sets $\hat{c} \leftarrow_\$ \mathbb{G}_\mathsf{T}$. In Game_2, instead of computing its value, we set $\mathsf{c}_1 \leftarrow_\$ \mathbb{G}$. For $\mathcal{A}$ to successfully distinguish between Game_1 and Game_2, the DLIN assumption would have to be tractable. Formally, we show this by constructing a simulator $\mathcal{S}_1$ [Fig. 17] that interpolates between Game_1 and Game_2 by playing game $\mathsf{DLIN}_{\Gamma,\mathcal{S}_1}$ [Fig. 2]. As in Theorem 1, the hash function H is modelled as a random oracle. Without loss of generality, we assume that $\mathcal{A}$ places *exactly* q queries and always asks for the hash value of id before querying id to Extract and Real-or-Random. Employing the same strategy as before, simulator $\mathcal{S}_1$ randomly tries to guess which query $\mathsf{i} \in \{0, \mathsf{q}\text{-}1\}$ contains the id on which $\mathcal{A}$ asks to be challenged. When i is sucessfully guessed, which happens with probability $\frac{1}{\mathsf{q}}$, $\mathcal{S}_1$ perfectly simulates Game_1 if Z is of the form $\mathsf{g}^{\mathsf{z}_3+\mathsf{z}_4}$, and perfectly simulates Game_2 otherwise. This implies that $\mathsf{t}_1 = \mathsf{z}_1$ and $\mathsf{t}_2 = \mathsf{z}_2$. Random function H is set to $(\mathsf{g}^{\mathsf{z}_2})^{\mathsf{x}}$ for every id but the id chosen for the challenge, which is set to g^{x}, where x is a random value sampled from $\mathbb{Z}_\mathsf{p}$. Challenge is well formed, as well as secret keys for random exponents $\mathsf{r}' = \frac{\mathsf{r}}{\mathsf{z}_2}$, where r is sampled from $\mathbb{Z}_\mathsf{p}$. Finally, notice that $\mathsf{s} = \mathsf{z}_3 + \mathsf{z}_4$

[6] For $\hat{c}$ we used the case where $\mathsf{Z} = \mathbf{e}(\mathsf{g},\mathsf{g})^{\mathsf{z}_1\mathsf{z}_2\mathsf{z}_3}$, which corresponds to the simulation of Game_0.

and $s_1 = z_4$, and that t_1, t_2, r', s and s_1 are uniformly distributed over $\mathbb{Z}_p$, as they should be. For completeness, we present the equalities between the original expressions and those computed by the simulator[7].

$$d_0 = g^{r' t_1 t_2} = g^{\frac{r}{z_2} z_1 z_2} = (g^{z_1})^r = Z_1^r$$
$$d_1 = g^{-w t_2} \cdot h^{-r' t_2} = g^{-w z_2} \cdot \left[(g^{z_2})^x\right]^{-\frac{r}{z_2} z_2} = Z_2^{-w} \cdot Z_2^{-xr} = Z_2^{-w-xr}$$
$$d_2 = g^{-w t_1} \cdot h^{-r' t_1} = g^{-w z_1} \cdot \left[(g^{z_2})^x\right]^{-\frac{r}{z_2} z_1} = Z_1^{-w} \cdot Z_1^{-xr} = Z_1^{-w-xr}$$

$$c_0 = h^s = (g^x)^{z_3+z_4} = Z^x$$
$$c_1 = v_1^{s-s_1} = (g^{z_1})^{(z_3+z_4)-z_4} = Z_{13}$$
$$c_2 = v_2^{s_1} = (g^{z_2})^{z_4} = Z_{24}$$

In Game_2, the challenge ciphertext does not depend on the receiver's identity. Therefore, we have that $\mathbf{Adv}^{\mathsf{IBE\text{-}ANO}}_{\Pi,\mathcal{A}}(\lambda) = \frac{1}{q} \cdot \mathbf{Adv}^{\mathsf{DBDH}}_{\Gamma,\mathcal{S}_0} + \frac{1}{q} \cdot \mathbf{Adv}^{\mathsf{DLIN}}_{\Gamma,\mathcal{S}_1}$, which concludes our proof. □

procedure Initialize(λ):
$(Z_1, Z_2, Z_{13}, Z_{24}, Z) \leftarrow$ DLIN.Initialize
$w \leftarrow_\$ \mathbb{Z}_p$
$\Omega \leftarrow \mathbf{e}(Z_1, Z_2)^w$
$v_1 \leftarrow Z_1$
$v_2 \leftarrow Z_2$
params $\leftarrow (\Omega, v_1, v_2)$
$\mathsf{list}_{\mathsf{ID}} \leftarrow [\,]$, $\mathsf{list}_{\mathsf{H}} \leftarrow [\,]$
$\mathsf{id}^\star \leftarrow$ null, $i \leftarrow_\$ \{0, q\text{-}1\}$
return params

procedure Extract(id):
if id == $\mathsf{id}^\star$ return $\perp$
if id is in $\mathsf{list}_{\mathsf{H}}[i]$ abort
get x for id from $\mathsf{list}_{\mathsf{H}}$
$r \leftarrow_\$ \mathbb{Z}_p$

$d_0 \leftarrow Z_1^r$
$d_1 \leftarrow Z_2^{-w-xr}$
$d_2 \leftarrow Z_1^{-w-xr}$

$\mathsf{sk}_{\mathsf{id}} \leftarrow (d_0, d_1, d_2)$
$\mathsf{list}_{\mathsf{ID}} \leftarrow \mathsf{id} : \mathsf{list}_{\mathsf{ID}}$
return $\mathsf{sk}_{\mathsf{id}}$

procedure Finalize(bit):
return

procedure H(id):
get x for id from $\mathsf{list}_{\mathsf{H}}$
if x ==$\perp$
..... $x \leftarrow_\$ \mathbb{Z}_p$
..... $\mathsf{list}_{\mathsf{H}} \leftarrow (\mathsf{id}, x) : \mathsf{list}_{\mathsf{H}}$
if id is in $\mathsf{list}_{\mathsf{H}}[i]$ then $h \leftarrow g^x$ else $h \leftarrow Z_2^x$
return h

procedure Real-or-Random($\mathsf{id}^\star$, m):
if $\mathsf{id}^\star \in \mathsf{list}_{\mathsf{ID}}$ return $\perp$
if $\mathsf{id}^\star$ is not in $\mathsf{list}_{\mathsf{H}}[i]$ abort

get x for $\mathsf{id}^\star$ from $\mathsf{list}_{\mathsf{H}}$

$\hat{c} \leftarrow_\$ \mathbb{G}_T$
$c_0 \leftarrow Z^x$
$c_1 \leftarrow Z_{13}$
$c_2 \leftarrow Z_{24}$

$c \leftarrow (\hat{c}, c_0, c_1, c_2)$
return c

Fig. 17. Simulator $\mathcal{S}_1$ interpolates between Game_1 and Game_2 by playing game $\mathsf{DLIN}_{\Gamma,\mathcal{S}_1}$.

C Proof of Theorem 3

Let $\mathcal{A}$ be any legitimate PPT adversary in game $\mathsf{WEAK\text{-}KEY\text{-}UNLINK}_{\Pi,\mathcal{A}}$ [Fig. 11]. By building a simulator $\mathcal{S}_2$ [Fig. 18] that plays game $\mathsf{DLIN}_{\Gamma,\mathcal{S}_2}$ [Fig. 2] and simulates game $\mathsf{WEAK\text{-}KEY\text{-}UNLINK}_{\Pi,\mathcal{A}}$ in such a way that $\mathcal{A}$'s guess can be forward to game $\mathsf{DLIN}_{\Gamma,\mathcal{S}_2}$, we upper-bound the adversary's advantage to the hardness of deciding on an instance of this problem.
The master secret key is set as following: $t_1 = z_1$, $t_2 = z_1 \cdot a$ for random $a \in \mathbb{Z}_p$, and $w = \frac{z_3 \cdot b}{z_1}$ for random $b \in \mathbb{Z}_p$. Although the values of t_1, t_2 and w are unknown to $\mathcal{S}_2$, the corresponding public parameters can still be consistently computed:

$$\Omega = \mathbf{e}(g,g)^{t_1 t_2 w} = \mathbf{e}(g,g)^{z_1 z_1 a \frac{z_3 \cdot b}{z_1}} = \mathbf{e}(Z_{13}, g)^{ab}$$
$$v_1 = g^{t_1} = Z_1$$
$$v_2 = g^{t_2} = (Z_1)^a$$

The hash function H is modelled as a random oracle and set to $(g^{z_1})^x \cdot g^{-\frac{1}{y}}$, for random $x, y \in \mathbb{Z}_p^2$. We assume, without loss of generality, that $\mathcal{A}$ always asks for the hash value of id before querying id to oracle Extract. Whenever asked to extract a private key on some id, we set $r = w \cdot y$, where y is the value used to compute the hash of that particular id. Note that this still makes r uniformly

[7] For c_0 we used the case where $Z = g^{z_3+z_4}$, which corresponds to the simulation of Game_1.

distributed over $\mathbb{Z}_p$ and independent of h and w. Given this, private keys can be extracted as follows:

$$d_0 = g^{rt_1t_2} = g^{wyt_1t_2} = g^{\frac{z_3 \cdot b}{z_1} yz_1z_1a} = (Z_{13})^{aby}$$
$$d_1 = g^{-wt_2} \cdot h^{-rt_2} = g^{-wt_2} \cdot \left[(g^{z_1})^x \cdot g^{-\frac{1}{y}}\right]^{-wyt_2} = g^{-z_1 x wyt_2} = g^{-z_1 x \frac{z_3 \cdot b}{z_1} yz_1a} = (Z_{13})^{-abxy}$$
$$d_2 = g^{-wt_1} \cdot h^{-rt_1} = g^{-wt_1} \cdot \left[(g^{z_1})^x \cdot g^{-\frac{1}{y}}\right]^{-wyt_1} = g^{-z_1 x wyt_1} = g^{-z_1 x \frac{z_3 \cdot b}{z_1} yz_1} = (Z_{13})^{-bxy}$$

Finally, to complete the simulation, we extract two private keys to challenge $\mathcal{A}$, such that these private keys are for the same id if $\mathcal{S}_2$ received a valid DLIN tuple, and for different ids otherwise. Let $sk^\star = (d_0^\star, d_1^\star, d_2^\star)$ and $sk^\circ = (d_0^\circ, d_1^\circ, d_2^\circ)$ be the challenge keys. We set $h = g^{z_1 z_4}$, $r^\star = \frac{b}{(z_1)^2}$ and $r^\circ = \frac{z_2 + b}{(z_1)^2}$. Note that h is uniformly distributed over $\mathbb{G}$, and $r^\star$ and r° are uniformly distributed over $\mathbb{Z}_p$, independent of each other and of w. For completeness, we present the equalities between the original expressions and those computed by the simulator.

$$d_0^\star = g^{r^\star t_1 t_2} = g^{\frac{b}{(z_1)^2} z_1 z_1 a} = g^{ab}$$
$$d_1^\star = g^{-wt_2} \cdot h^{-r^\star t_2} = g^{-\frac{z_3 b}{z_1} z_1 a} \cdot \left(g^{z_1 z_4}\right)^{-\frac{b}{(z_1)^2} z_1 a} = \left(g^{-ab}\right)^{z_3} \cdot \left(g^{-ab}\right)^{z_4} = Z^{-ab}$$
$$d_1^\star = g^{-wt_1} \cdot h^{-r^\star t_1} = g^{-\frac{z_3 b}{z_1} z_1} \cdot \left(g^{z_1 z_4}\right)^{-\frac{b}{(z_1)^2} z_1} = \left(g^{-b}\right)^{z_3} \cdot \left(g^{-b}\right)^{z_4} = Z^{-b}$$

$$d_0^\circ = g^{r^\circ t_1 t_2} = g^{\frac{z_2 + b}{(z_1)^2} z_1 z_1 \cdot a} = g^{z_2 \cdot a + ab} = (Z_2)^a \cdot g^{ab}$$
$$d_1^\circ = g^{-wt_2} \cdot h^{-r^\circ t_2} = g^{-\frac{z_3 b}{z_1} z_1 a} \cdot \left(g^{z_1 z_4}\right)^{-\frac{z_2 + b}{(z_1)^2} z_1 a} = \left(g^{-ab}\right)^{(z_3 + z_4)} \cdot \left(g^{z_2 z_4}\right)^{-a} = Z^{-ab} \cdot (Z_{24})^{-a}$$
$$d_2^\circ = g^{-wt_1} \cdot h^{-r^\circ t_1} = g^{-\frac{z_3 b}{z_1} z_1} \cdot \left(g^{z_1 z_4}\right)^{-\frac{z_2 + b}{(z_1)^2} z_1} = \left(g^{-b}\right)^{(z_3 + z_4)} \cdot \left(g^{z_2 z_4}\right)^{-1} = Z^{-b} \cdot (Z_{24})^{-1}$$

Therefore, we have that $\mathbf{Adv}^{\text{WEAK-KEY-UNLINK}}_{\Pi,\mathcal{A}}(\lambda) = \mathbf{Adv}^{\text{DLIN}}_{\Gamma,\mathcal{S}_2}$, which concludes our proof. □

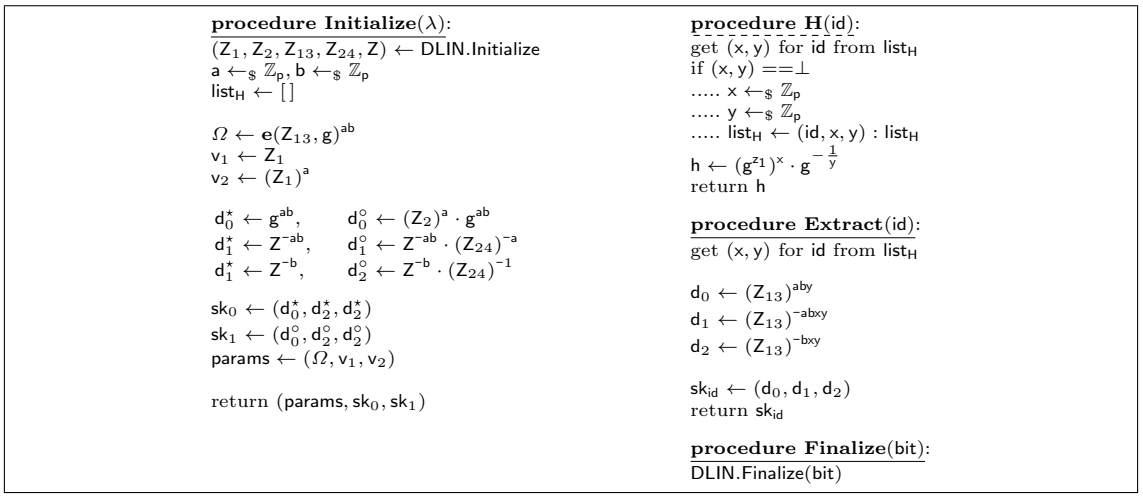

Fig. 18. Simulator $\mathcal{S}_2$ forwards $\mathcal{A}$'s guess from game $\text{WEAK-KEY-UNLINK}_{\Pi,\mathcal{A}}$ to game $\text{DLIN}_{\Gamma,\mathcal{S}_2}$.

D Proof of Theorem 4

First, let us show an important re-randomization propriety that scheme Π' possess and that is relevant for the completion of this proof. From two keys honestly extracted from the same identity, say $sk_0 = (d_{00}, d_{01}, d_{02})$ and $sk_1 = (d_{10}, d_{11}, d_{12})$, one can generate new valid keys for that identity with fresh random coins, without the knowledge of any secret parameter. Concretely, $sk_2 = (d_{20}, d_{21}, d_{22})$ can be generated as follows, with a random $y \in \mathbb{Z}_n$ and random $R_0, R_1, R_2 \in \mathbb{G}_q$:

$$d_{20} = R_0 \cdot \left(\frac{d_{10}}{d_{00}}\right)^y \cdot d_{00} = \left[R_0 \cdot \frac{(x_{10})^y}{(x_{00})^{(y-1)}}\right] \cdot g^{[yr_1 - (y-1)r_0]t_1t_2}$$
$$d_{21} = R_1 \cdot \left(\frac{d_{11}}{d_{01}}\right)^y \cdot d_{01} = \left[R_1 \cdot \frac{(x_{11})^y}{(x_{01})^{(y-1)}}\right] \cdot g^{-wt_2} \cdot h^{-[yr_1 - (y-1)r_0]t_2}$$

$\mathsf{d}_{22} = \mathsf{R}_2 \cdot (\frac{\mathsf{d}_{12}}{\mathsf{d}_{02}})^{\mathsf{y}} \cdot \mathsf{d}_{02} = [\mathsf{R}_2 \cdot \frac{(\mathsf{x}_{12})^{\mathsf{y}}}{(\mathsf{x}_{02})^{(\mathsf{y}-1)}}] \cdot \mathsf{g}^{-\mathsf{wt}_1} \cdot \mathsf{h}^{-[\mathsf{yr}_1-(\mathsf{y}-1)\mathsf{r}_0]\mathsf{t}_1}$

Let $\mathcal{A}$ be any PPT adversary against $\mathsf{STRONG\text{-}KEY\text{-}UNLINK}_{\Pi',\mathcal{A}}$ [Fig. 12]. We now drastically simplify the security model, so that it looks like the one presented in Fig. 19, which we call 5-KEY-UNLINK. Using a hybrid argument and taking advantage of the re-randomization property previously described, we show that the advantage of $\mathcal{A}$ against $\mathsf{STRONG\text{-}KEY\text{-}UNLINK}_{\Pi',\mathcal{A}}$ is polynomially-bounded by the advantage of $\mathcal{A}$ against 5-KEY-UNLINK.

procedure Initialize(λ):
$(\mathsf{Msk}, \mathsf{params}) \leftarrow_{\$} \mathsf{Setup}(1^{\lambda})$
$\mathsf{bit} \leftarrow_{\$} \{0,1\}$
$\mathsf{id}_0 \leftarrow_{\$} \mathcal{I}$
$\mathsf{id}_1 \leftarrow_{\$} \mathcal{I}$
$\mathsf{sk}_0 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_0)$
$\mathsf{sk}_1 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_0)$
$\mathsf{sk}_2 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_{\mathsf{bit}})$
$\mathsf{sk}_3 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_1)$
$\mathsf{sk}_4 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_1)$
return $(\mathsf{params}, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3, \mathsf{sk}_4)$

procedure Extract(id):
$\mathsf{sk}_{\mathsf{id}} \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id})$
return $\mathsf{sk}_{\mathsf{id}}$

procedure Finalize(bit'):
return $(\mathsf{bit} = \mathsf{bit}')$

Fig. 19. 5-KEY-UNLINK$_{\Pi,\mathcal{A}}$ Game.

In $\mathsf{STRONG\text{-}KEY\text{-}UNLINK}_{\Pi',\mathcal{A}}$, $\mathcal{A}$ submits two lists list_0 and list_1 of the same length, say k, for the challenge. For this argument, we construct $\mathsf{k}+1$ lists. The first list is list_0 and the last list is list_1. In between, we have k-1 intermediate lists that transition from list_0 to list_1, one element at the time. The k-1 intermediate lists are constructed such that the first list is list_0, and for every $\mathsf{i} \in \{1..\mathsf{k}\text{-}1\}$, $\mathsf{list}^{\mathsf{i}} = \mathsf{list}^{\mathsf{i}-1}$, except for the element $\mathsf{list}^{\mathsf{i}}[\mathsf{i}]$ which is taken from $\mathsf{list}_1[\mathsf{i}]$. Again, the last list is list_1. The advantage $\mathcal{A}$ has in distinguishing list_0 from list_1 cannot be more than the sum of the advantages of distinguishing $\mathsf{list}^{\mathsf{i}-1}$ from $\mathsf{list}^{\mathsf{i}}$, for every $\mathsf{i} \in \{0..\mathsf{k}\}$. The probability of distinguishing $\mathsf{list}^{\mathsf{i}-1}$ from $\mathsf{list}^{\mathsf{i}}$ cannot be more than that of identifying the form of the tuple in model 5-KEY-UNLINK. More precisely, one can expand the 5-tuple $(\mathsf{sk}_0^{\circ}, \mathsf{sk}_1^{\circ}, \mathsf{sk}_2^{\circ}, \mathsf{sk}_3^{\circ}, \mathsf{sk}_4^{\circ})$ from 5-KEY-UNLINK into a k-tuple of keys that corresponds to the requirements of either $\mathsf{list}^{\mathsf{i}-1}$ or $\mathsf{list}^{\mathsf{i}}$. Since the lists only (possibly) differ in position i, we set sk_{i} of the k-tuple to sk_2°. Every other key is extracted from the extraction oracle of model 5-KEY-UNLINK or generated from $(\mathsf{sk}_0^{\circ}, \mathsf{sk}_1^{\circ})$ or $(\mathsf{sk}_3^{\circ}, \mathsf{sk}_4^{\circ})$ if the key is required to be extracted from the identity in $\mathsf{list}^{\mathsf{i}-1}[\mathsf{i}]$ or $\mathsf{list}^{\mathsf{i}}[\mathsf{i}]$, respectively.

The model can be further simplified to that of Fig. 20, which we call 4-KEY-UNLINK. Again, we make use of the so-called hybrid argument and the re-randomization property introduced in the beginning of this proof that Π' possesses[8], the difficulty of distinguishing a 5-tuple of keys extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_1, \mathsf{id}_1)$ from those extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0)$, where id_0 and id_1 are sampled from $\mathcal{I}$, is equivalente to that of distinguishuing a 4-tuple of keys that were extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_1, \mathsf{id}_1)$ from those extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0)$, since the fifth key the adversary could generate himself. This difficulty of distinguishing the 5-tuple of keys extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_1, \mathsf{id}_1, \mathsf{id}_1)$ from those extracted from $(\mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0, \mathsf{id}_0)$ is also the same as distinguishing the key tuple in 4-KEY-UNLINK model. So, the advantage $\mathcal{A}$ has in distinguishing the tuples in 5-KEY-UNLINK game cannot be more than *twice* the advantage $\mathcal{A}$ has in distinguishing the tuples in 4-KEY-UNLINK.

procedure Initialize(λ):
$(\mathsf{Msk}, \mathsf{params}) \leftarrow_{\$} \mathsf{Setup}(1^{\lambda})$
$\mathsf{bit} \leftarrow_{\$} \{0,1\}$
$\mathsf{id}_0 \leftarrow_{\$} \mathcal{I}$
$\mathsf{id}_1 \leftarrow_{\$} \mathcal{I}$
$\mathsf{sk}_0 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_0)$
$\mathsf{sk}_1 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_0)$
$\mathsf{sk}_2 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_{\mathsf{bit}})$
$\mathsf{sk}_3 \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id}_{\mathsf{bit}})$
return $(\mathsf{params}, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3)$

procedure Extract(id):
$\mathsf{sk}_{\mathsf{id}} \leftarrow_{\$} \mathsf{Extract}(\mathsf{params}, \mathsf{Msk}, \mathsf{id})$
return $\mathsf{sk}_{\mathsf{id}}$

procedure Finalize(bit'):
return $(\mathsf{bit} = \mathsf{bit}')$

Fig. 20. 4-KEY-UNLINK$_{\Pi,\mathcal{A}}$ Game

[8] From two keys honestly extracted for the same identity, we can generate a third one with random coins.

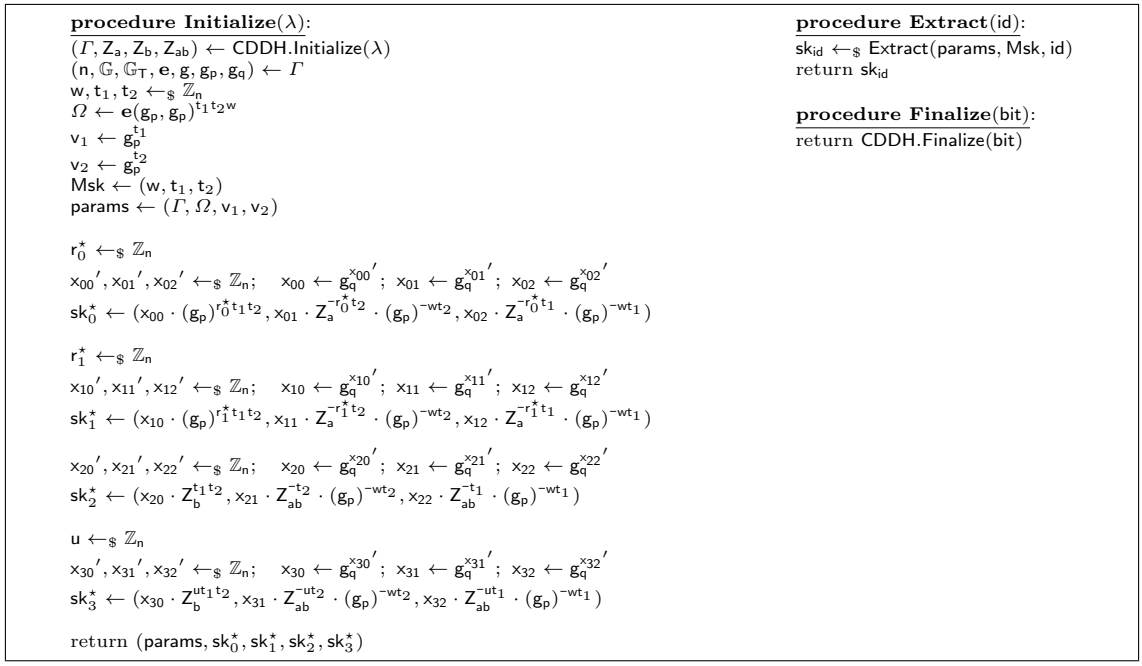

Fig. 21. Simulator $\mathcal{S}_3$ forwards $\mathcal{A}$'s guess from 4-KEY-ANO$_{\Pi',\mathcal{A}}$ to game CDDH.

To complete the proof, we build a simulator $\mathcal{S}_3$ [Fig. 21] that by playing game $\mathsf{CDDH}_{\Gamma',\mathcal{S}_3}$ outputs four keys $(\mathsf{sk}_0^\star, \mathsf{sk}_1^\star, \mathsf{sk}_2^\star, \mathsf{sk}_3^\star)$ such that the adversary's guess in 4-KEY-UNLINK$_{\Pi',\mathcal{A}}$ can be forward to game $\mathsf{CDDH}_{\Gamma',\mathcal{S}_3}$. We refer to key $\mathsf{sk}_\mathsf{i}^\star$ as the tuple $(\mathsf{d}_{\mathsf{i}0}^\star, \mathsf{d}_{\mathsf{i}1}^\star, \mathsf{d}_{\mathsf{i}2}^\star)$, associated with $\mathsf{h}_\mathsf{i}^\star$, the hashed-identity from which $\mathsf{sk}_\mathsf{i}^\star$ was extracted. If the simulator receives a well-formed CDDH tuple, $\mathsf{h}_0^\star = \mathsf{h}_1^\star = \mathsf{h}_2^\star = \mathsf{h}_3^\star$ is set to g^a. Otherwise, $\mathsf{h}_0^\star = \mathsf{h}_1^\star = \mathsf{g}^\mathsf{a}$ and $\mathsf{h}_2^\star = \mathsf{h}_3^\star$ with an independent random value in $\mathbb{G}_\mathsf{p}$. We also set $\mathsf{r}_2^\star = \mathsf{b}$ and $\mathsf{r}_3^\star = \mathsf{b} \cdot \mathsf{u}$, for a random $\mathsf{u} \in \mathbb{Z}_\mathsf{n}$.

Finally, we have that $\mathbf{Adv}^{\mathsf{STRONG\text{-}KEY\text{-}UNLINK}}_{\Pi',\mathcal{A}}(\lambda) \leq 2\mathsf{L} \cdot \mathbf{Adv}^{\mathsf{CDDH}}_{\Gamma',\mathcal{S}_3}$, which concludes our proof. □