# Privacy-preserving Policy-based Information Transfer

Emiliano De Cristofaro[1], Stanislaw Jarecki[1], Jihye Kim[2], Gene Tsudik[1]

[1] Computer Science Department, University of California, Irvine
[2] Department of Mathematical Sciences, Seoul National University

**Abstract.** As the global society becomes more interconnected and more privacy-conscious, communication protocols must balance access control with protecting participants' privacy. A common current scenario involves an authorized party (client) who needs to retrieve sensitive information held by another party (server) such that: (1) the former only gets the information for which it is duly authorized, (2) the latter does not learn what information information is retrieved. To address this scenario, in this paper, we introduce and explore the concept of Privacy-preserving Policy-based Information Transfer (PPIT). We construct three PPIT schemes based, respectively, on: RSA, Schnorr and IBE techniques. We then investigate various performance improvements and demonstrate the practicality of proposed PPIT schemes.

## 1 Introduction

There are many scenarios where sensitive information is requested by some authority due to some legitimate need. The challenge for the information owner (server) is to allow access to only duly authorized information, whereas, the challenge for the information requester (client) is to obtain needed information without divulging what is being requested. We refer to this concept as *Privacy-preserving Policy-based Information Transfer* or PPIT. To motivate it, we begin with two envisaged scenarios:

**Scenario 1.** University of Lower Vermont (ULoVe) is confronted with an FBI investigation focused on one of its faculty members (Alice). The university is understandably reluctant to allow FBI unlimited access to its employee records. For its part, FBI is unwilling to disclose that Alice is the target of investigation. There might be several reasons for FBI's stance: (1) Concern about unwarranted rumors and tarnishing Alice's reputation, e.g. leaked information might cause legal action and result in bad PR for the FBI; (2) The need to keep the investigation secret, i.e., preventing malicious insiders (ULoVe employees) from forewarning Alice about the investigation.

Ultimately, ULoVe must comply with FBI's demands, especially, if the latter is armed with appropriate authorization (e.g., a court order) from, say, the US Attorney General's office. However, the authorization presumably applies only to Alice. Assuming all communication between ULoVe and FBI is electronic, there seems to be an impasse.

An additional nuance is that, even if ULoVe is willing to provide FBI unrestricted access to all its employee records, FBI may not want the associated liability. This is because mere possession of ULoVe sensitive employee information would require FBI to demonstrate that the information is/was treated appropriately and disposed of when no longer needed. Considering a number of recent incidents of massive losses of sensitive government and commercial employees' records, FBI might be unwilling to assume additional risk.

An ideal solution would be as follows: ULoVe learns that FBI is most likely investigating someone who might be an employee of ULoVe. No one outside FBI learns who is being investigated. This holds even if someone in ULoVe tries to manipulate the process attempting to learn more information. For its part, FBI learns nothing about any ULoVe employee who does not meet the exact criteria specified in its court order.

**Scenario 2.** An international airline (VultureAir) has daily flights transiting the United States. US Department of Homeland Security (DHS) maintains a **secret** terrorist watch-list and needs to determine whether any names on the watch-list match those on the passenger manifest of each VultureAir flight. Bound by some international privacy treaty (or its own policy), VultureAir is unwilling to disclose its passenger list to DHS. However, as ULoVe in Scenario 1, VultureAir is ready to comply with DHS's request as long as each entry on the DHS's watch-list is individually and duly authorized by the independent Judicial Branch.

Ideally, VultureAir transfers information to DHS only about those passengers for which DHS has valid authorizations. In the process, VultureAir does not learn whether DHS has an authorization on any of its passengers. In particular, VultureAir can learn nothing about the DHS watch-list by manipulating its own passenger lists. More generally, no party learns any material it should not have, either by law or because of liability. Nonetheless, DHS retrieves all information to which it is entitled.

**What is PPIT?** Privacy-preserving Policy-based Information Transfer (PPIT) is applicable to any scenario with a need to transfer information – and, more generally, perform some data-centric task – between parties who:

1. Are willing and/or obligated to transfer information in an accountable and policy-guided (*authorized*) manner.
2. Need to ensure privacy of server's data by preventing unauthorized access.
3. Need to ensure privacy of client's *authorization(s)* which grant it access to server's data.

**PPIT vs Prior Techniques.** As evident from the remainder of this paper, PPIT's main technical challenge is how to enable the server to efficiently and *obliviously* compute proper authorization decisions. This might sound similar to the goals of certain other concepts, which are overviewed in this section.

Of course, PPIT could be trivially implemented with the aid of on-line trusted third party (TTP) which could take data from both parties and perform necessary operations. However, on-line TTPs are generally unrealistic, for a number of well-known reasons. As any two-party security problem, PPIT could be implemented using generic secure computation techniques [22]. However, such generic techniques are unlikely to yield protocols efficient enough to be used in practice.

PPIT has some features in common with Private Information Retrieval (PIR) [7]. Although PIR aims to ensure privacy of client's query target(s) from the server, a PIR server is willing to unconditionally release any and all of its data to the client. In *symmetric* PIR [11, 16], the server releases to the client exactly one data item per query. However, there is no provision for ensuring that the client is authorized by some trusted authority to retrieve the requested item. Also, a PIR protocol must communicate strictly fewer bits than the the server's database size. Whereas, PPIT involves no such requirements; indeed, PPIT protocols presented in this paper have linear communication complexity.

PPIT can be thought of a variant of secure set intersection [10, 12, 13]. For example, in Scenario 2, a secure set intersection protocol would allow DHS and VultureAir to *privately* compute an intersection of their respective lists (terror watch-list and passenger manifest). However, note that both parties could inject arbitrary data into the protocol. In contrast, in PPIT, the client is forced to request data for which it has valid authorization obtained from appropriate authorities. Thus, PPIT is a strictly stronger *policy-based* version of the secure set intersection problem: its privacy guarantees are the same, but the client's input is controlled by well-defined access policies, e.g., authorization certificates.

PPIT is also related to *Public Encryption with Keyword Search* (PEKS) [2] or searchable encrypted logs [21]. The server could use a PEKS scheme to attach encryptions of keywords to encrypted database entries, which can be tested by the client only using a corresponding trapdoor. Although this can be used to implement PPIT, it is unclear how to make the resulting protocols efficient using existing PEKS schemes in the setting where the client has multiple credentials. However, as shown in this paper, one can indeed construct an efficient PPIT scheme following this approach using Anonymous Identity-Based Encryption: the server encrypts each entry under its keyword, and the client decrypts it using the decryption key corresponding to the same keyword.

Finally, another closely related construction is *Oblivious Signature-Based Envelope* (OSBE) [14]. Like PPIT, OSBE allows the server to release some information to the client conditional upon the latter's possession of a signature (on a message known to both parties, e.g. a keyword) by a trusted authority, while the server learns nothing about the signatures held by the client. This can be implemented using Identity-Based Encryption, or using standard signature schemes, such as RSA, Schnorr, and DSS [14, 17]. However, unlike PPIT, an OSBE scheme does not guarantee privacy of *all* information about the client's authorization. Nevertheless, as shown in this paper, OSBE schemes can be adapted to obtain efficient PPIT instantiations.

**Contributions.** This paper makes several contributions: (1) it defines a new cryptographic notion, PPIT, motivated by certain practical scenarios, (2) it shows that PPIT can be resolved under a variety of standard assumptions, (3) it constructs several efficient PPIT protocols for the case of the client with multiple authorizations, and (4) it demonstrates feasibility of proposed PPIT instantiations with experimental results obtained from prototype implementations.

## 2 Privacy-preserving Policy-based Information Transfer

This section describes out notation as well as the participants and the components of a PPIT scheme.

**Notation.** A function $f(\tau)$ is *negligible* in the security parameter $\tau$ if, for every polynomial $p$, $f(\tau) < 1/|p(t)|$ for large enough $t$. Throughout this paper, we use semantically secure symmetric encryption and we assume the key space to be $\tau_1$-bit strings, where $\tau_1$ is a (polynomial) function of a security parameter $\tau$. We use $Enc_k(\cdot)$ and $Dec_k(\cdot)$ to denote symmetric-key encryption and decryption (both under key $k$), respectively. We also use public key signature schemes, where each scheme is a tuple of algorithms: $DSIG = [INIT, SIG, VER]$, representing key set-up, signature generation and verification, respectively. $DSIG.INIT(\tau_2)$ returns a public/private key-pair, where $\tau_2$ is a polynomial function of $\tau$. $DSIG.SIG(SK, m)$ returns a signature $\sigma$ on message $m$, whereas, $DSIG.VER(PK, \sigma, m)$ returns 1 or 0 indicating that $\sigma$ is valid or invalid signature on $m$, under $PK$. Finally, we use $a \leftarrow A$ to designate that variable $a$ is chosen uniformly at random from set $A$.

**Players/Entities.** A PPIT scheme involves three players:

(1) Server (S): stores the set $I = \{(ID, D_{ID}) \mid ID \in \{0,1\}^l\}$. $ID$ uniquely identifies a record and $D_{ID}$ denotes the associated information.
(2) Client (C): has a pair $(\sigma, ID_C)$, where $\sigma$ is authorization for $ID_C$ issued by the court and $ID_C$ is an $l$-bit string.
(3) Court: a trusted third party, which issues authorizations for accessing a record identified by a given string $ID$.

**Components.** Without loss of generality, we assume that an authorization $\sigma$ for record with identifier $ID$ is a signature under CA's key on $ID$. Therefore we define a PPIT scheme as a tuple of the following three algorithms:

(1) Setup($\tau$): It is an algorithm executed by the court. Given a security parameter $\tau$, it generates – via $DSIG.INIT$ – a key-pair $(SK, PK)$ for the signature scheme $DSIG$. The court then publishes the public key $PK$.
(2) Authorize($SK, ID$): It is an algorithm executed by the court to issue an authorization $\sigma = DSIG.SIG(SK, ID)$ on an identifier string $ID$. Note that if $\sigma =$ Authorize(SK,ID) then $DSIG.VER(PK, \sigma, ID) = 1$.
(3) Transfer: It is an interactive algorithm (protocol) executed between server S and client C, on public input $PK$, on S's private input $(ID_S, D)$ and C's private input $(ID_C, \sigma)$. At the end of transfer, S has no outputs and C outputs $D$ if $ID_S = ID_C$ and $DSIG.VER(PK, ID_C, \sigma) = 1$.

## 3 Security Requirements

We now describe PPIT security requirements.

**Correctness.** A PPIT scheme is *correct* if, at the end of $\mathsf{transfer}$, C outputs $D$, given that:
(1) $(SK, PK) \leftarrow \mathsf{Setup}(1^\tau)$ and $\sigma = \mathsf{Authorize}(ID)$ for some $ID$,
(2) S and C respectively run the $\mathsf{transfer}$ protocol on input $(ID, D)$ and $(ID, \sigma)$.

**Security.** Informally, security of a PPIT scheme means that only clients authorized to access data $D$ can learn any information about $D$. Formally, we say that a PPIT scheme is *secure* if any polynomially bounded adversary $\mathcal{A}$ cannot win the following game, with probability non-negligibly over 1/2. The game is between $\mathcal{A}$ and a challenger $Ch$:

1. $Ch$ runs $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$
2. $\mathcal{A}$, on input $PK$, adaptively queries $Ch$ a number $n$ of times on a set of strings $Q = \{ID_i | ID_i \in \{0,1\}^l, i = 1, \cdots, n\}$. For every $ID_i$, $Ch$ responds by giving $\mathcal{A}$ a signature $\sigma_i \leftarrow DSIG.SIG(SK, ID_i)$
3. $\mathcal{A}$ announces a new identifier string, $ID^* \notin Q$, and generates two equal-length data record $({D_0}^*, {D_1}^*)$
4. $Ch$ picks one record by selecting a random bit $b \leftarrow \{0,1\}$, and executes the server's part of the $\mathsf{transfer}$ protocol on public input $PK$ and private inputs $(ID^*, {D_b}^*)$. We denote the protocol transcript by $T^*$.
5. $\mathcal{A}$ outputs $b'$ (and wins if $b' = b$).

**Server Privacy.** Informally, a PPIT scheme is server-private if only an authorized client learns any information about $ID_S$ which S inputs into the transfer protocol with C. Formally, we say that a PPIT scheme is *server-private* if no polynomially bounded adversary $\mathcal{A}$ can win the following game with probability non-negligibly over 1/2. The game is between $\mathcal{A}$ and $Ch$:

1. $Ch$ runs $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$
2. $\mathcal{A}$, on input $PK$, adaptively queries $Ch$ a number $n$ of times on a set of strings $Q = \{ID_i | ID_i \in \{0,1\}^l, i = 1, \cdots, n\}$. For every $ID_i$, $Ch$ responds by giving $\mathcal{A}$ a signature $\sigma_i \leftarrow DSIG.SIG(SK, ID_i)$
3. $\mathcal{A}$ announces two new identifier strings, $({ID_0}^*, {ID_1}^*) \notin Q$, and generates a data record $D^*$
4. $Ch$ picks one identifier by selecting a random bit $b \leftarrow \{0,1\}$, and executes the server's part of $\mathsf{transfer}$ on public input $PK$ and private inputs $({ID_b}^*, D^*)$. We denote the protocol transcript by $T^*$.
5. $\mathcal{A}$ outputs $b'$ (and wins if $b' = b$).

We note that security and server-privacy games could be merged into one. It is possible to modify $\mathcal{A}$ to announce two pairs $({ID_0}^*, {D_0}^*), ({ID_1}^*, {D_1}^*)$ and let $Ch$ pick a random bit $b$ and execute the server's part of $\mathsf{transfer}$ on input $({ID_b}^*, {D_b}^*)$. The *security* property alone is obtained by restricting $\mathcal{A}$'s challenge query so that $({ID_0}^* = {ID_1}^*)$, while *server-privacy* alone is obtained if $({D_0}^* = {D_1}^*)$.

**Client Privacy.** Informally, client privacy means no information is leaked about client's authorization and $ID$ to a malicious server. Formally, a PPIT scheme is *client-private* if no polynomially bounded adversary $\mathcal{A}$ can win the following game with the probability non-negligibly over 1/2. The game is between $\mathcal{A}$ and $Ch$:

1. $Ch$ executes $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$
2. $\mathcal{A}$, on input $SK$, chooses two strings $ID_0{}^*$, $ID_1{}^*$ and two strings $\sigma_0{}^*, \sigma_1{}^*$
3. $Ch$ picks a random bit $b \leftarrow \{0,1\}$ and interacts with $\mathcal{A}$ by following $\mathsf{transfer}$ on behalf of client on public input $PK$ and private inputs $(ID_b{}^*, \sigma_b{}^*)$
4. $\mathcal{A}$ outputs $b'$ (and wins if $b' = b$).

For the sake of simplicity, we say that a PPIT scheme is ***private*** if it is both server- and client-private.

**Client Unlinkability.** Informally, client unlinkability means that a malicious server cannot tell if any two instances of the $\mathsf{transfer}$ protocol are related, i.e., executed on the the same inputs $ID_C$ and/or $\sigma$. Formally, we say that a PPIT is *client-unlinkable* if no polynomially bounded adversary $\mathcal{A}$ can win the following game with probability non-negligibly over 1/2. The game is between $\mathcal{A}$ and $Ch$:

1) $Ch$ executes $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$
2) $\mathcal{A}$, on input $SK$, chooses two strings $ID_0{}^*$, $ID_1{}^*$ (where it could be that $ID_0{}^* = ID_1{}^*$) and two strings $\sigma_0{}^*, \sigma_1{}^*$

3a) $Ch$ interacts with $\mathcal{A}$ by following $\mathsf{transfer}$ on behalf of client on public input $PK$ and private inputs $(ID_0{}^*, \sigma_0{}^*)$

3b) $Ch$ picks a random bit $b \leftarrow \{0,1\}$ and interacts with $\mathcal{A}$ by following $\mathsf{transfer}$ on behalf of client on public input $PK$ and private inputs $(ID_b{}^*, \sigma_b{}^*)$.

4) $\mathcal{A}$ outputs $b'$ (and wins if $b' = b$).

In other words, observing $\mathsf{transfer}$ does not give $\mathcal{A}$ any advantage in the game described for client privacy.

## 4 Building blocks

In this section, we present three PPIT variants, based on RSA signatures scheme [19], Schnorr signature scheme [20], and Anonymous Identity-Based Encryption (IBE) [3]. For ease of presentation, we assume that S stores just one pair: $(ID_S, D_{ID_S})$. The full versions of these schemes, described in Section 5, work with multiple records on S and/or multiple authorizations on C.

### 4.1 RSA-PPIT

We show how to obtain PPIT by adapting the RSA-based Oblivious Signature Based Envelope (OSBE) scheme of [14].

**Setup.** On input of security parameter $\tau$, generate a safe RSA modulus $N = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, and $p, q, p', q'$ are primes. The set of all quadratic residues mod $N$ is denoted as $QR_N$. The algorithm picks a random element $g$ which is a generator of $QR_N$. RSA exponents $(e, d)$ are chosen in the standard way. The secret key is $SK = (p, q, d)$ and the public key $PK = (N, g, e)$. The algorithm also fixes a full-domain hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$, $H' : \{0,1\}^* \rightarrow \{0,1\}^{\tau_1}$.

**Authorize.** To issue an authorization on $ID$ to C court computes an RSA signature on $ID$, $\sigma = (h_{ID})^d \pmod N$, where $h_{ID} = H(ID)$. (The signature on $ID$ is verified by checking if $\sigma^e = H(ID)$.)

**Transfer.** This is a protocol between C and S where public input is $PK = (N, e, g)$, and C's private input is $(ID_C, \sigma)$, where $\sigma^e = H(ID_C) \bmod N$ and S's private input is $(ID_S, D)$. The protocol is shown in Figure 1.

| | | |
|---|---|---|
| **C** on input: $ID_C, \sigma, PK = (N, g, e)$ where $\sigma^e = H(ID_C)$ | | **S** on input: $ID_S, D, PK = (N, g, e)$ |
| $r \leftarrow \mathbb{Z}_{N/4}$, $\mu = \sigma^2 \cdot g^r \bmod N$ | $\xrightarrow{\mu}$ | If $\mu \notin \mathbb{Z}_N^*$ then abort. $R = g^{ez} \bmod N$, for $z \leftarrow \mathbb{Z}_{N/4}$ $K_S = (\mu)^{ez} \cdot (H_{ID_S})^{-2z} \bmod N$ $k_S = H'(K_S)$ |
| $K_C = R^r \bmod N$, $k_C = H'(K_C)$ $D' = Dec_{k_C}(C)$ | $\xleftarrow{\langle R, C\rangle}$ | $C = Enc_{k_S}(D)$ |

**Fig. 1.** RSA-PPIT

To see that RSA-PPIT is *correct*, observe that, when $ID_C = ID_S$:
$K_S = (\mu)^{ez} \cdot H(ID_S)^{-2z} = (H(ID_C))^{2z} \cdot g^{rez} \cdot H(ID_S)^{-2z} = g^{erz} = R^r = K_C$.

Recall that this scheme is based on RSA-OSBE from [14]. However, in the first step of the transfer, C picks $\mu = \sigma^2 \cdot g^r$ instead of $\sigma \cdot h^r$. The use of $g$ – instead of $h = H(ID)$ – allows C to batch computation in case it has multiple authorizations. (For more details, see Section 5). Also, we square $\sigma$ to guarantee that $\mu$ is in $QR_N$, as shown in the proof in Appendix C, where we present the complete proof of security, privacy, and client-unlinkability for RSA-PPIT.

### 4.2 Schnorr-PPIT

We show here a PPIT construction using Schnorr-OSBE scheme [6]. It's proof of security, privacy, and client-unlinkability is in Appendix C.

**Setup.** On input of a security parameter $\tau$, this algorithm creates a Schnorr key: $(p, q, g, a, y)$, where $p$, $q$ are primes, s.t. $q$ divides $p-1$ but $q^2$ does not divide $p - 1$, $g$ is a generator of a subgroup in $\mathbb{Z}_p^*$ of order $q$, $a$ is picked randomly in

$\mathbb{Z}_q^*$, and $y = g^a \bmod p$. The public key is $PK = (p, q, g, y)$ and the secret key is $SK = a$. The algorithm also defines hash functions $H : \{0,1\}^* \rightarrow\in \mathbb{Z}_q^*$, and $H' : \{0,1\}^n \rightarrow\in \{0,1\}^{\tau_1}$.

**Authorize.** To issue authorization on string $ID$, court computes a Schnorr signature on $ID$, $\sigma = (X, s)$ where $X = g^k \bmod p$ and $s = k + a \cdot H(ID, X) \bmod q$ for random $k \leftarrow \mathbb{Z}_q^*$. The signature on $ID$ is verified by checking whether $g^s = X \cdot y^{H(ID,X)} \bmod p$.

**Transfer.** This protocol (see Figure 2) is between C and S, where public input is $PK = (p, q, g, y)$, and C's private input is $(ID_C, \sigma = (X, s))$ s.t. $g^s = X \cdot y^{H(ID_C,X)} \bmod p$ and S's private input is $(ID_S, D)$.

| | | |
|---|---|---|
| **<u>C</u>** on input: $ID_C, \sigma = (X, s)$ $PK = (p, q, g, y)$ | | **<u>S</u>** on input: $ID_S, D, PK = (p, q, g, y)$ |
| $g^s = X \cdot y^{H(ID_C,X)} \bmod p$ | $\xrightarrow{X}$ | If $X^{(p-1)/q} \neq 1 \bmod p$ then abort. $R = g^z \bmod p$, for $z \leftarrow \mathbb{Z}_q^*$ $K_S = (y^{H(ID_S,X)}X)^z \bmod p$, $k_S = H'(K_S)$ |
| $K_C = R^s \bmod p$, $k_C = H'(K_C)$ $D' = Dec_{k_C}(C)$ | $\xleftarrow{\langle R,C \rangle}$ | $C = Enc_{k_S}(D)$ |

**Fig. 2.** Schnorr-PPIT

To show that Schnorr-PPIT is *correct*, we observe that, when $ID_C = ID_S$:
$K_S = (y^{H(ID_S,X)}X)^z = (g^{aH(ID_S,X)}g^k)^z = (g^{aH(ID_C,X)+k})^z = g^{sz} = R^s = K_C$

## 4.3 IBE-PPIT

Here we show a PPIT construction using any anonymous Identity-Based Encryption (IBE) scheme, e.g. [3, 4]. Recall that IBE is a form of public key encryption where any string can be used as a public key. A trusted third party, called a Key Distribution Center (KDC), has a master key, which is used to generate the private key corresponding to any public key string.

**Setup.** On input of a security parameter $\tau$, the Court runs the setup algorithm of the IBE system to generate the KDC master key and global IBE system parameters, denoted as $PK$.

**Authorize.** As shown in [3], selective-ID semantically secure IBE implies CMA-secure signatures. The authorization on $ID$ is thus simply $\sigma$ – the IBE private key corresponding to the public key $ID$. The verification tests that the private key corresponds to the given ID.

**Transfer.** IBE implies a non-interactive PPIT scheme: S encrypts $D$ under the identifier string $ID_S$ and C decrypts it using its authorization $\sigma$, which is an IBE private key identifier corresponding to C's string $ID_C$.

We observe that this is similar to the IBE-based Signature Based Envelope (OSBE) scheme previously explored in [14]. However, in IBE-based OSBE, the use of anonymous IBE to achieve key-privacy (in the sense of [1]) is optional. Whereas, this is a fundamental requirement in our scheme: an adversary who correctly guesses the encryption key used to generate a ciphertext would immediately break server privacy.

To see that IBE-PPIT is *correct*, observe that, when $ID_C = ID_S$, C has the corresponding authorization from the court, i.e. the private decryption key, and hence will successfully decrypt data $D$. The complete proof of security, privacy, and client-unlinkability for IBE-PPIT are in Appendix C.

## 5 Extensions

Thus far, we considered solutions for a simple scenario where client is authorized to access at most one record and server stores at most one record. We now consider the case of multiple authorizations/records.

**Multiple Records.** Consider a setting where server stores a set of $n$ records, denoted by by $I = \{(ID_i, D_i) | ID_i \in \{0,1\}^l\}$, with $|I| = n$. Since one of the PPIT requirements prevents server to know which record is requested, server has to send all of its records. A naïve solution would be to reiterate the interaction presented in the previous section $n$ times. Specifically, the PPIT transfer protocol would require: (i) server to perform and send $n$ encryptions under $n$ different Diffie-Hellman [8] keys, and (ii) client to try decrypting *all* received encryptions to output the authorized record. This would result in $O(n)$ encryptions, bandwidth utilization, and decryptions.

Hence, we aim to speed up the computation by using a two-pronged approach:

1. We let server use same random values, $z$, across all records, so that encryptions can be batched.
2. We let server accompany every encryption with a tag, i.e. a hash function of each Diffie-Hellman key, $K_{s_i}$. In turn, client computes the tag on its own Diffie-Hellman key, $K_C$ so that it decrypts only the record accompanied by the matching the tag.

**Fast decryption in RSA-PPIT.** We assume that in the setup algorithm, an additional cryptographic hash function $H''$ is chosen. The protocol is shown in Figure 3. As a result, the computation for client is reduced to $O(1)$. However, we cannot speed up server-side computation without violating client-privacy. (See Appendix C for proofs.)

**Fast decryption in Schnorr-PPIT.** We assume that in the setup algorithm, an additional cryptographic hash function $H''$ is chosen. The protocol is shown in Figure 4. Similar to RSA-PPIT, client's computation cost becomes constant, but there does not seem to be way to speed up server-side computation without violating client-privacy. (The proofs are deferred to Appendix C.)

| | | |
|---|---|---|
| <u>**C**</u> on input $(ID_C, \sigma, PK = (N, g, e))$<br>where $\sigma^e = H(ID_C)$ | | <u>**S**</u> on input $(\{ID_i, D_i\}, PK = (N, g, e))$ |
| $r \leftarrow \mathbb{Z}_{N/4}$, $\mu = \sigma^2 \cdot g^r \bmod N$ | $\xrightarrow{\mu}$ | If $\mu \notin \mathbb{Z}_N^*$ then abort.<br>$R = g^{ez} \bmod N$, for $z \leftarrow \mathbb{Z}_{N/4}$<br>For every $i$ compute:<br>$K_{S_i} = (\mu)^{ez} \cdot (H_{ID_i})^{-2z} \bmod N$<br>$k_{S_i} = H'(K_{S_i})$<br>$C_i = Enc_{k_{S_i}}(D_i)$, $t_i = H''(K_{S_i})$ |
| $K_C = R^r \bmod N$, $k_C = H'(K_C)$<br>$t^* = H''(K_C)$<br>If $\exists i \in \{1, ..., n\}$, s.t. $t_i = t^*$<br>Output $D' = Dec_{k_C}(C_i)$. | $\xleftarrow{\langle R, \overrightarrow{C}, \overrightarrow{T} \rangle}$ | $\overrightarrow{C} = \{C_i\}$ and $\overrightarrow{T} = \{t_i\}$ |

**Fig. 3.** RSA-PPIT with multiple records

| | | |
|---|---|---|
| <u>**C**</u> on input: $(ID_C, \sigma = (X, s)$<br>$PK = (p, q, g, y))$ | | <u>**S**</u> on input: $(\{ID_i, D_i\}, PK = (p, q, g, y))$ |
| for $g^s = X \cdot y^{H(ID_C, X)} \bmod p$ | $\xrightarrow{X}$ | If $X^{(p-1)/q} \neq 1 \bmod p$ then abort.<br>$R = g^z \bmod p$, for $z \leftarrow \mathbb{Z}_q^*$<br>For every $i$ compute:<br>$K_{S_i} = (y^{H(ID_i, X)} \cdot X)^z \bmod p$<br>$k_{S_i} = H'(K_{S_i})$<br>$C_i = Enc_{k_{S_i}}(D_i)$, $t_i = H''(K_{S_i})$ |
| $K_C = R^s \bmod p$, $k_C = H'(K_C)$<br>$t^* = H''(K_C)$<br>If $\exists i \in \{1, ..., n\}$, s.t. $t_i = t^*$<br>Output $D' = Dec_{k_C}(C_i)$. | $\xleftarrow{\langle R, \overrightarrow{C}, \overrightarrow{T} \rangle}$ | $\overrightarrow{C} = \{C_i\}$ and $\overrightarrow{T} = \{t_i\}$ |

**Fig. 4.** Schnorr-PPIT with multiple records

**Fast decryption in IBE-PPIT.** Similarly to RSA-PPIT and Schnorr-PPIT extensions above, we label each encrypted record with a tag based on the corresponding encryption keys. Then client quickly retrieves and decrypts the record for which it has the decryption key.

As described in Section 4.3, PPIT can be instantiated using any efficient anonymous IBE scheme. We now describe how to compute key tags using Boneh-Franklin's IBE [3], presented in Appendix A. To support multiple records, we add key tagging, where tags are computed using BF-IBE, as we show in Figure 5. We assume that two cryptographic hash function $H, H'$ are chosen during setup. Recall that, in BF-IBE, $s$ is the master private key and $(P, Q = sP)$ are public parameters. We modify the Authorize so that, when the court issues to client an authorization for ID, client also receives a signature $\sigma' = s \cdot H(ID)$. Note that $PK_i^{IBE}$ is the IBE private key corresponding to public key $ID_i$ and $SK^{IBE}$ is the IBE private key corresponding to public key $ID^*$ for which client holds a court-issued authorization.

**C** on input: $(ID_C, \sigma, \sigma', P, Q)$
for $\sigma = SK^{IBE}, \sigma' = s \cdot H(ID)$

**S** on input: $(\{ID_i, D_i\}, \{PK_i^{Ibe}\}, P, Q)$

$R = zP$, for random $z \leftarrow \mathbb{G}_1$
For every $i$ compute:
$C_i = \mathsf{IBE\text{-}Encrypt}(PK_i^{IBE}, D_i)$
$t_i = H'(e(Q, H(ID_i))^z)$

Compute $t^* = H'(e(R, \sigma'))$ $\xleftarrow{\langle R, \overrightarrow{C}, \overrightarrow{T} \rangle}$ $\overrightarrow{C} = \{C_i\}$ and $\overrightarrow{T} = \{t_i\}$

If $\exists i \in \{1, ..., n\}$, s.t. $t_i = t^*$, output
$D' = \mathsf{IBE\text{-}Decrypt}(SK^{IBE}, C_i)$

**Fig. 5.** IBE-PPIT with multiple records

We emphasize, that re-use of randomness $z$ for each tag in the IBE scheme is similar to [5]. However, our approach provides multi-encryption (i.e., encryption of different messages) instead of broadcast encryption [9]. Moreover, we embed the tags to reduce the number of decryptions to $O(1)$. Proofs appear in Appendix C.

**Multiple Authorizations.** We now consider the scenario where client receives multiple authorizations, allowing access to multiple records with only one instance of PPIT. We say that client stores $\Sigma = \{(ID_j, \sigma_j) | ID_j \in \{0,1\}^l\}$, with $|\Sigma| = n'$, the set of $n'$ pairs defining a record identifier along with the court's authorization. For completeness, we consider server to store the set $I$ of pairs $(ID_i, D_i)$ with $|I| = n$.

As discussed above, server has to send all of its records during the transfer. RSA-PPIT and Schnorr-PPIT require a different Diffie-Hellman key for each record. In these interactive instantiations, each key depends on some partial information of client's alleged authorization. For this reason, in the RSA-PPIT transfer protocol client should send $\mu_j = {\sigma_j}^2 \cdot g^{r_j}$ for every $j$. In the Schnorr-PPIT, client should send $X_j = g^{s_j} \cdot y^{-e_j}$ for every $j$. This implies that server should compute and send $n' \cdot n$ encryptions under $n' \cdot n$ different Diffie-Hellman keys, resulting in $\mathrm{O}(n \cdot n')$ computation time for both entities, as well as $\mathrm{O}(n \cdot n')$ bandwidth utilization. Using the tag extensions presented above, the number of decryptions will be reduced to the linear $O(n')$.

Specifically, the computation on server is burdened by performing $O(n \cdot n')$ exponentiations needed to compute $n \cdot n'$ Diffie-Hellman keys. However, in RSA-PPIT the number of exponentiations can be easily reduced to $O(n + n')$. In fact, it is possible to separately compute: (i) $n'$ different exponentiations for the received $\mu_j$, for $j = 1, ...n'$, (ii) $n$ different exponentiations for ${H_{ID_i}}^{-2z}$, for $i = 1, ..., n$. Then for each Diffie-Hellman key computation, server should only perform a multiplication, thus resulting in $O(n \cdot n')$ total multiplications.

In contrast, the IBE-based extension presented for multiple records can be applied unaltered to the scenario with client's multiple authorizations.. Therefore, the server computation and the bandwidth utilization remains O($n$), as well as the client computation remains linear in the number of authorizations, i.e. $O(n')$. This is possible because the transfer protocol in IBE-PPIT is a one-round interaction and no information is sent from client to server. Hence, the encryption keys do not depend on any information sent by client.

## 6 Discussion

We now evaluate and compares the proposed schemes for the case of S with $n$ records and C with $n'$ authorizations.

**Performance Analysis.** Table 1 summarizes the performance of the proposed PPIT schemes. Both S's and C's run-times are measured in terms of public key operations, **ops**, i.e., exponentiations in case of RSA-PPIT and Schnorr-PPIT (exponent sizes are, respectively, 1024 and 160 bits), and bilinear map operations in case of IBE-PPIT. However, for server operations in RSA-PPIT, we distinguish between exponentiations, **exp**, and multiplications **mul**. Recall that $n$ is the number of records stored by S, and $n'$ – the number of authorizations held by the C.

| | **RSA** | **Schnorr** | **IBE** |
|---|---|---|---|
| Transfer Rounds | 2 | 2 | 1 |
| Server ops | $O(n+n')$ **exp** $O(n \cdot n')$ **mul** | $O(n \cdot n')$ | $O(n)$ |
| Client ops | $O(n')$ | $O(n')$ | $O(n')$ |
| Bandwidth | $O(n \cdot n')$ | $O(n \cdot n')$ | $O(n)$ |

**Table 1.** Performance Comparison for scenarios where $n' >> 1$.

IBE-PPIT is the most efficient by all counts, since it: (1) takes one round, (2) requires a linear number of public key operations for both S and C, and (3) consumes linear amount of bandwidth. Whereas, both Schnorr-PPIT and RSA-PPIT are two-round protocols. Schnorr-PPIT has quadratic – $O(n \cdot\ n')$ – computation and bandwidth overheads, while RSA-PPIT requires $O(n + n')$ exponentiations and $O(n \cdot n')$ multiplication on S. However, for small $n'$ values the Schnorr-PPIT and the RSA-PPIT protocols might be faster because they use less expensive operations (modular exponentiations versus bilinear maps).

The dominant cost factor varies with the scheme: (1) in RSA-PPIT it is a 1024-bit exponentiation mod $N$, (2) in Schnorr-PPIT, it is a 160-bit exponentiation mod 1024-bit prime $p$, and (3) in IBE-PPIT, it is the bilinear map function.

**Experimental Results.** To assess the performance of proposed schemes, we now present some experimental results. All tests were performed on S with two quad-core CPUs Intel Xeon at 1.60 GHz with 8GB RAM. All schemes were implemented in ANSI C, using the well-known OpenSSL toolkit [23], except pairing operations in IBE-PPIT for which we used the PBC Library [15]. We used 1024-bit moduli for RSA-PPIT, 1024- and 160-bit primes for Schnorr-PPIT, and 512-bit group elements and 160-bit primes for IBE-PPIT [3]. We note that all our tests measured total computation time for PPIT transfer, i.e. the sum of client's and server's computation times. We did not measure time for setup or authorization, since these algorithms are performed only once, at initialization time. We also note that both client and server were running on the same machine; thus, measurements do not take into account the transmission time.

**Multiple records and one authorization.** In the first test, we timed the performance of the transfer protocol between S storing an increasing number of records and C holding a single authorization. Figure 6(a) shows that Schnorr-PPIT is the fastest, while IBE-PPIT is (not surprisingly) the slowest one.

**Multiple records and multiple authorizations.** In the second test, we experimented with the case of C holding 100 **authorizations** and S having an increasing number of records. Figure 6(b) shows that IBE-PPIT becomes faster than Schnorr-PPIT, yet, remains slower that RSA-PPIT.

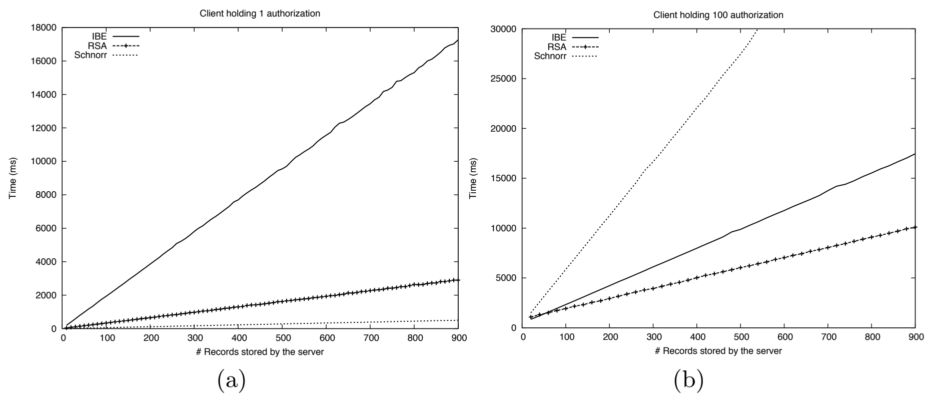

**Fig. 6.** Total Computation time for PPIT-Transfer for 1 (100) client authorization(s) and increasing server records

We also timed the case where S has a fixed number of records (**100 records**) and C holds an increasing number of authorizations. Figure 7 shows that IBE-PPIT is clearly faster than RSA-PPIT when C has more than 200 authorizations.

However, note that, in this scenario, transmission time can be a relevant factor. In IBE-PPIT, this is linear in the number of S's records. Whereas, in Schnorr-PPIT and RSA-PPIT, the bandwidth is proportional to the product of

[3] Details on the curves we used can be found in PBC library documentation at `http://crypto.stanford.edu/pbc/manual/ch06s01.html` and `http://crypto.stanford.edu/pbc/manual/ch10s03.html`

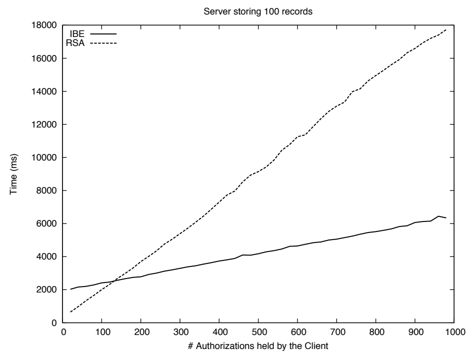

**Fig. 7.** Total Computation time for PPIT-Transfer for increasing client authorizations and 100 server records

S's records and C's authorizations. For instance, if S has $1,000$ records, C holds 100 authorizations, and each record is $1,000$ bits, the bandwidth for RSA-PPIT and Schnorr-PPIT would be on the order of 100Mb but only 1Mb for IBE-PPIT.

**Observation.** Experimental results yield several observations:

1. Schnorr- and RSA-PPIT are preferred over IBE-PPIT in settings where C holds a few authorizations (e.g., Scenario 1 in Section 1.). As shown in Figure 6(a), IBE-PPIT is much slower than others and the speed gap grows linearly with the number of records. In particular, Schnorr-PPIT is efficient enough for quite large databases.
2. IBE-PPIT is preferred for settings where C holds many authorizations (e.g., Scenario 2 in Section 1). IBE allows us to avoid interaction, which saves a lot of computation and bandwidth, especially, if C holds many authorizations.

**Unlinkability and Forward Security.** We now discuss some differences in terms of security features provided by the three schemes.

First, note that, unlike RSA-PPIT, Schnorr-PPIT does not offer *client-unlinkability*, since the value $X = g^k$ sent by C stays fixed for a given $ID$. IBE-PPIT is trivially unlinkable.

Whenever multiple transfers take place, *forward security* becomes important. We say that a PPIT scheme *forward-secure* if:

1. Adversary who learns all of S's data (ID-s and records) cannot violate client-privacy of prior transfer interactions.
2. Adversary who learns C's authorization(s) cannot violate security and server-privacy of past transfer interactions.

Note that the first part of the forward-security requirement is already achieved through the notion of *client-privacy*. The second part is not obvious.

RSA-PPIT provides built-in forward security due to the use of $g^r$ in computing each $\mu$. Schnorr-PPIT and IBE-PPIT schemes do not provide forward security, but this can be easily added by requiring both C and S to establish an ephemeral Diffie-Hellman key [8]. This modification increases the computation cost by adding an extra exponentiation. In addition, it makes the IBE-PPIT transfer protocol interactive.

## 7 Conclusion

This paper introduced a new cryptographic notion of Privacy-preserving Policy-based Information Transfer (PPIT). We constructed and compared three different PPIT instantiations, based, respectively, on: RSA, Schnorr and IBE techniques. We also proposed simple techniques for improving server and/or client performance for cases where either or both parties have multiple records and authorizations, respectively. In our future work, we plan to investigate other solutions to obtain linear complexity using standard digital signatures.

## References

1. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Asiacrypt'02*, pages 566–582, 2002.
2. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key Encryption with Keyword Search. In *Eurocrypt'04*, pages 506–522, 2004.
3. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
4. X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Crypto'06*, pages 290–307, 2006.
5. R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *CCS'04*, pages 146–157, 2004.
6. C. Castelluccia, S. Jarecki, and G. Tsudik. Secret Handshakes from CA-Oblivious Encryption. In *Asiacrypt'04*, pages 293–307, 2004.
7. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.
8. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
9. A. Fiat and M. Naor. Broadcast Encryption. In *Crypto'93*, pages 480–491, 1993.
10. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt'04*, pages 1–19, 2004.
11. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *STOC'98*, pages 151–160, 1998.
12. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC'08*, pages 155–175, 2008.
13. S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, pages 577–594, 2009.
14. N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. *Distributed Computing*, 17(4):293–302, 2005.
15. B. Lynn. PBC: The Pairing-Based Cryptography Library. `http://crypto.stanford.edu/pbc/`.
16. M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *STOC'99*, pages 245–254, 1999.
17. S. Nasserian and G. Tsudik. Revisiting oblivious signature-based envelopes. In *Financial Cryptography'06*, pages 221–235, 2006.
18. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Eurocrypt'96*, pages 387–398, 1996.

19. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
20. C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
21. B. Waters, D. Balfanz, G. Durfee, and D. Smetters. Building an encrypted and searchable audit log. In *NDSS'04)*, 2004.
22. A. Yao. Protocols for secure computations. In *FOCS'82*, pages 160–164, 1982.
23. E. Young and T. Hudson. OpenSSL: The Open Source toolkit for SSL/TLS. `http://www.openssl.org`.

## A Boneh and Franklin's IBE

We recall that the BF-IBE scheme is composed by four algorithms: *setup*, *extract*, *encrypt*, *decrypt*.

*Setup*, given a security parameter $k$, is used to generate a prime $q$, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order $q$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then a random $s \in \mathbb{Z}_q^*$, a random generator $P \in \mathbb{G}_1$, $P$ are chosen and $Q$ is set such that $Q = sP$. $(P, Q)$ are public parameters. $s$ is the private master key. Finally, two cryptographic hash function, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^n \rightarrow \mathbb{G}_2$ for some $n$, are chosen.

*Extract*, given a string $ID \in \{0,1\}^*$, is used to compute the corresponding private key $s \cdot H(ID)$.

*Encrypt* is used to encrypt a message $M$ under a public key $ID$: for a picked random $r \in Z_q^*$ the ciphertext is set to be $C = \langle rP, M \oplus H_2(e(Q, H_1(ID)^r)\rangle$.

*Decrypt* is used to decrypt a ciphertext $C = \langle U, V \rangle$, by computing $M = V \oplus H_2(e(U, sH(ID))$.

## B Cryptographic Assumptions

**RSA assumption.** Let $RSASetup(\tau)$ be an algorithm that outputs so-called safe RSA instances, i.e. pairs $(N, e)$ where $N = pq$, $e$ is a small prime that satisfies $gcd(e, \phi(N)) = 1$, and $p, q$ are randomly generated $\tau$-bit primes subject to the constraint that $p = 2p' + 1$, $q = 2q' + 1$ for prime $p', q'$, $p' \neq q'$. We say that the RSA problem is $(\tau, t)$-hard on $2\tau$-bit safe RSA moduli, if for every algorithm $\mathcal{A}$ that runs in time $t$ we have

$$Pr[(N, e) \leftarrow RSASetup(\tau), \alpha \leftarrow \mathbb{Z}_N^* : \mathcal{A}(n, e, \alpha) = \beta \text{ s.t. } \beta^e = \alpha \pmod N] \leq \tau.$$

**CDH Assumption.** Let $G$ be a cyclic group of prime order $q$ with a generator $g$. We say that the Computational Diffie-Hellman Problem (CDH) in $G$ is $(\epsilon, t)$-hard if for every algorithm $\mathcal{A}$ running in time $t$ we have

$$\Pr[x \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y) = g^{xy}] \leq \epsilon.$$

**DDH oracle:** A DDH oracle in group $G$ is an algorithm that returns 1 on queries of the form $(g, g^x, g^y, g^z)$ where $z = xy \bmod q$, and 0 on queries of the form $(g, g^x, g^y, g^z)$ where $z \neq xy \bmod q$.

**GDH Assumption.** We say that the Gap Diffie-Hellman Problem (GDH) in group $G$ is $(\epsilon, t)$-hard if for every algorithm $\mathcal{A}$ running in time $t$ on access to the DDH oracle $\mathsf{DDH_G}$ in group $G$ we have

$$\Pr[x \leftarrow \mathbb{Z}_q : \mathcal{A}^{\mathsf{DDH_G}}(g, g^x, g^y) = g^{xy}] \leq \epsilon.$$

## C Proofs

**Basic RSA-PPIT.** RSA-PPIT is *secure, private, and client-unlinkable* under the RSA assumption described in Appendix B on safe RSA moduli and the GDH assumption in the Random Oracle Model, given semantically secure symmetric encryption.

*Proof.* We first prove the security and server-privacy by demonstrating that no efficient $\mathcal{A}$ (acting as a client) has a *non-negligible* advantage over 1/2 against $Ch$ in the following game:

1. $Ch$ executes $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$ and gives $PK$ to $\mathcal{A}$.
2. $\mathcal{A}$ invokes Authorize on $ID_j$ of its choice and obtains the corresponding signature $\sigma_j$.
3. $\mathcal{A}$ generates $ID_0^*$, $ID_1^*$ and two equal-length data records $D_0{}^*, D_1{}^*$.
4. $\mathcal{A}$ participates in transfer as a client with message $\mu^*$.
5. $Ch$ selects one record pair by selecting a random bit $b$ and executes the server's part of the transfer protocol on public input $PK$ and private inputs $(ID_b^*, D_b{}^*)$ with message $(R, C)$.
6. $\mathcal{A}$ outputs $b'$ and wins if $b = b'$.

Let $\mathsf{HQuery}$ be an event that $\mathcal{A}$ ever queries $H'$ on input $K^*$, where $K^*$ is defined (as the combination of message $\mu^*$ sent by $\mathcal{A}$ and message C sent by $Ch$), as follows: $K^* = (\mu^*)^{ez} \cdot (h^*)^{-2z} \bmod N$, where $R = (g)^{ez}$ and $h^* = H(ID^*)$. In other words, $\mathsf{HQuery}$ is an event that $\mathcal{A}$ computes (and enters into hash function $H'$) the key-material $K^*$ for the challenging protocol.

[Claim 1]. Unless $\mathsf{HQuery}$ happens, $\mathcal{A}$'s view of interaction with $Ch$ on bit $b = 0$ is indistinguishable from $\mathcal{A}$'s view of the interaction with $Ch$ on bit $b = 1$.

Since the distribution of $R = g^{ez}$ is independent from $(ID_b, D_b)$, it reveals no information about which of $(ID_b, D_b)$ is related in the protocol. Since PPIT uses a semantically secure symmetric encryption, the distribution with $b = 0$ is indistinguishable from that with $b = 1$, unless $\mathcal{A}$ computes $k^* = H'(K^*)$, in the random oracle model, by querying $H'$, i.e., $\mathsf{HQuery}$.

[Claim 2]. If event $\mathsf{HQuery}$ happens with non-negligible probability, then $\mathcal{A}$ can be used to violate the RSA assumption.

We describe a reduction algorithm called $RCh$ using a modified challenger algorithm. Given the RSA challenge $(N, e, \alpha)$, $RCh$ sets the public key as $(N, e, g)$ where $g$ is a generator of $QR_N$. $RCh$ simulates signatures on each $ID_j$ by assigning $H(ID_j)$ as $\sigma_j^e \bmod N$ for some random value $\sigma_j$. In this way $RCh$ can present the certificate of $ID_j$ as $\sigma_j$. $RCh$ embeds $\alpha$ to each $H$ query, by setting $H(ID_i) = \alpha(a_i)^e$ for random $a_i \in \mathbb{Z}_N$. Note that given $(H(ID_i))^d$ for any $ID_i$ the simulator can extract $\alpha^d = (H(ID_i))^d / a_i$.

We describe how $RCh$ responds to $\mathcal{A}$ in the transfer protocol and how $RCh$ computes $(H(ID_i))^d$ for certain $ID_i$. On $\mathcal{A}$'s input message $\mu^*$, $RCh$ picks a random $m \leftarrow \mathbb{Z}_{N/4}$, computes $R = g^{(1+em)}$, and sends C and a random encryption

$C$ to $\mathcal{A}$. We remark that $g^{1+em} = g^{e(d+m)}$. On the HQuery event, $RCh$ gets $K^* = (\mu^*)^{e(d+m)}(h^*)^{-2(d+m)}$ from $\mathcal{A}$. Since $RCh$ knows $\mu^*$, $h^*$, $e$, and $m$, $RCh$ can compute $(h^*)^{2d}$. Since $gcd(2, e) = 1$, computing $(h^*)^{2d}$ leads to computing $(h^*)^d$.

We prove client-privacy and unlinkability. In the following description, we use $U \approx_S V$ to denote that distribution $U$ is statistically close to $V$ in the sense that the difference between these distributions is at most $O(2^\tau)$. We show that $\{h^{2d}g^x\}_{x \leftarrow \mathbb{Z}_{N/4}} \approx_S QR_N$. Take any $h \in \mathbb{Z}_N^*$ and compute $\sigma = h^d \bmod N$. (Note that since $N - (p'q')$ is on the order of $\sqrt{N}$, which is negligible compared to $N$, the distribution of $h$ chosen in $\mathbb{Z}_N$ is statistically close to uniform in $\mathbb{Z}_N^*$.) Since multiplication by $h^{2d}$ is a permutation in $QR_N$, we have

$$\{h^{2d}g^x\}_{x \leftarrow \mathbb{Z}_{p'q'}} \equiv QR_N.$$

Since $\mathbb{Z}_{N/4} \approx_S \mathbb{Z}_{p'q'}$, the above implies that

$$\{h^{2d}g^x\}_{x \leftarrow \mathbb{Z}_{N/4}} \approx_S QR_N.$$

Since client selects a random value for each protocol instance, it is easy to know that RSA-PPIT scheme also provides client-unlinkability. □

**Basic Schnorr-PPIT.** Schnorr-PPIT is *secure, private* (but *not* client-unlinkable) under the GDH assumption (described in Appendix B) in the Random Oracle Model, given semantically secure symmetric encryption.

*Proof.* Client-privacy is easy to know since $X = g^k$ for random $k$ is independent from the $ID$ value. We now prove security and server privacy. For the security and privacy, we use the same game as in RSA-PPIT security and server-privacy, except $\mathcal{A}$ sends $X^*$ instead of $\mu^*$.

Let HQuery be an event that $\mathcal{A}$ ever queries $H'$ on input $K^*$, where $K^*$ is defined via the combination of the message $X^*$ sent by $\mathcal{A}$ and message $R$ sent by $Ch$, as follows: $K^* = (X^*)^z \cdot (y)^{cz} \bmod p$, where $R = g^z$ and $c = H(i, X^*)$.

[Claim 3]: Unless HQuery happens, $\mathcal{A}$'s view of the interaction with the challenger on bit $b = 0$ is indistinguishable from $\mathcal{A}$'s view of the interaction with the challenger on bit $b = 1$.

[Claim 4]: If event HQuery happens with non-negligible probability, then $\mathcal{A}$ can be used to break the CDH assumption (described in Appendix B).

We describe a reduction algorithm called $RCh$ using a modified challenger algorithm. The goal of a CDH problem on $(p, q, g, y = g^a, R = g^z)$ is to compute $g^{az} \bmod p$. $RCh$ takes $(p, q, g, y = g^a)$ as its public key and simulates the signatures $(X_j, s_j)$ on each ID $j$ by taking random $s_j$, $e_j$ and computing $X_j = g^{s_j} \cdot y^{e_j}$ and assigning $H(j, X_j)$ to $e_j$. Since the verification equation is satisfied and $s_j$, $e_j$ are picked at random, this is indistinguishable from receiving real signatures. In the protocol on $\mathcal{A}$'s input $X^*$, $RCh$ responds with $R = g^z$ and random encryption $C$.

Assume that HQuery happens, which can be detected by querying to $DDH$ oracle on $(g, X^* \cdot y^e, g^z, Q_H)$ for every query input $Q_H$ to $H$. Then, as in the forking lemma argument of [18], we know that $\mathcal{A}$ can be executed twice in a row with the same value $X = g^k \bmod p$ and different hash values such that $(e \neq e')$ and $\mathcal{A}$ wins both games with non-negligible probability of at least $\frac{\epsilon^2}{q_h}$, where $q_H$ is the number of queries $\mathcal{A}$ makes to the hash function. This means, $\mathcal{A}$ can compute with non-negligible probability the values $K = g^{z(ea+k)} \bmod p$ and $K' = g^{z(e'a+k)} \bmod p$ with $e \neq e'$. Consequently, $\mathcal{A}$ can also efficiently compute $g^{az}$: $(K/K')^{(e-e')^{-1}} = (g^{zea-ze'a})^{(e-e')^{-1}} = (g^{za(e-e')})^{(e-e')^{-1}} = g^{az} \bmod p$. □

**Basic IBE-PPIT.** IBE-PPIT is *secure, private and client-unlinkable* if IBE is semantically secure and key-private under selective ID attack.

*Proof.* Providing client-privacy and unlinkability is trivial since server does not receive any information from client in the transfer.

Assuming an underlying IBE system semantically secure under a chosen ciphertext attack and key-private, the resulting PPIT scheme is trivially *secure* and *server-private* against a malicious client. We prove this claim by contradiction. Assuming our claim is not true, then there exists a polynomial-bounded adversary $\mathcal{A}$ that wins the security game in Section 2. $\mathcal{A}$ is given the $PK$ = "$\mathcal{A}$ is authorized to access the record $ID$" and the IBE-encryption of $D_{ID}$ under the key $PK$ but not the corresponding $SK$. If $\mathcal{A}$ decrypts $D_{ID}$ with non-negligible probability, then we can construct a polynomial-bounded adversary $\mathcal{B}$ which uses $\mathcal{A}$ to break the CCA-security of IBE. This contradicts our assumption.

Finally, server-privacy is trivially achieved if the underlying IBE scheme is key-private. □

**RSA-PPIT Extension.** RSA-PPIT extension in Figure 3 is *secure, private, and client-unlinkable* under the RSA assumption on safe RSA moduli in the Random Oracle Model, given semantically secure symmetric encryption.

*Proof.* The proof for the client-privacy and unlinkability is the same as shown in the proof for Theorem 1.

We now prove the security and server-privacy. The game for the extended scheme is the same as for the basic scheme, except the adversary challenges the protocol on two pairs of input vectors $(\overrightarrow{ID_0^*}, \overrightarrow{D_0^*})$, $(\overrightarrow{ID_1^*}, \overrightarrow{D_1^*})$, instead of $(ID_0^*, D_0^*)$, $(ID_1^*, D_1^*)$. Namely, we demonstrate that no efficient $\mathcal{A}$ (acting as a client) has a *non-negligible* advantage over 1/2 against $Ch$ in the following game:

1. $Ch$ executes $(PK, SK) \leftarrow \mathsf{Setup}(1^\tau)$ and gives $PK$ to $\mathcal{A}$.
2. $A$ invokes Authorize on $ID_j$ of its choice and obtains the corresponding signature $\sigma_j$.
3. $\mathcal{A}$ generates two ID vectors:
   $\overrightarrow{ID_0^*} = \{ID_{0i}\}_{i=1,\ldots,n}$, $\overrightarrow{ID_1^*} = \{ID_{1i}\}_{i=1,\ldots,n}$,
   and two corresponding record vectors
   $\overrightarrow{D_0^*} = \{D_{0i}\}_{i=1,\ldots,n}$, $\overrightarrow{D_1^*} = \{D_{1i}\}_{i=1,\ldots,n}$.
4. $\mathcal{A}$ participates in transfer as a client with message $\mu^*$.
5. $Ch$ selects one record pair by selecting a random bit $b$ and executes the server's part of the transfer protocol on public input $PK$ and private inputs $(\overrightarrow{ID_b^*}, \overrightarrow{D_b^*})$ with message $(R, C)$.
6. $\mathcal{A}$ outputs $b'$ and wins if $b = b'$.

We define HQuery the same event as in the proof for RSA-PPIT. By the hybrid argument, if the adversary wins the above game with a non-negligible advantage over 1/2, HQuery happens on at least one pair $(ID_{bj}^*, D_{bj}^*)$ out of $(\overrightarrow{ID_b^*}, \overrightarrow{D_b^*})$. Using this adversary, we can build a reduction algorithm to break the RSA assumption, by the same argument as described in the proof for Theorem 1. □

**Schnorr-PPIT Extension.** Schnorr-PPIT extension in Figure 4 is *secure, private* (but *not* client-unlinkable) under the GDH assumption in the Random Oracle Model, given semantically secure symmetric encryption.

*Proof.* Again, as in the basic Schnorr-PPIT scheme, client-privacy is easy to know since $X = g^k$ for random $k$ is independent from the $ID$ value.

For the security and privacy, we use the same game used for Schnorr-PPIT security and server-privacy, except $A$ selects two pairs of vectors $(\overrightarrow{ID_0^*}, \overrightarrow{D_0^*})$, $(\overrightarrow{ID_1^*}, \overrightarrow{D_1^*})$, instead of $(ID_0^*, D_0^*)$, $(ID_1^*, D_1^*)$. We define HQuery the same event as in the proof for Schnorr-PPIT. By the hybrid argument, if the adversary wins the above game with a non-negligible advantage over 1/2, HQuery happens on at least one pair $(ID_{bj}^*, D_{bj}^*)$ out of $(\overrightarrow{ID_b^*}, \overrightarrow{D_b^*})$. Using this adversary, we can build a reduction algorithm to break the GDH assumption, by the same argument as described in the proof for Theorem 2. □

**IBE-PPIT Extension.** IBE-PPIT extension in Figure 5 is *secure, private and client-unlinkable* if IBE is semantically secure and key-private under selective ID attack.

*Proof.* Again, providing client-privacy and unlinkability is trivial since server does not receive any information from client in the transfer.

Assuming an underlying IBE system semantically secure under a chosen ciphertext attack, the resulting PPIT scheme is trivially *secure* and *server-private* against malicious C. This claim is true in the case of multiple records. The proof is similar to that described in Appendix C. Assuming our claim is not true then there exists a polynomial-bounded adversary $A$ that wins the game on vectors $(\overrightarrow{ID_0^*}, \overrightarrow{D_0^*})$, $(\overrightarrow{ID_1^*}, \overrightarrow{D_1^*})$. By the hybrid argument, $A$ decrypts at least one encrypted record out of $\overrightarrow{D_b^*}$ with $b \leftarrow \{0, 1\}$ without the corresponding secret key. This adversary can be used to construct an algorithm to break the CCA-security of IBE. This contradicts our assumption.

Furthermore, we argue that the use of key tags to reduce the number of client's decryptions do not affect the security and privacy of the scheme under the assumption that Boneh and Franklin's IBE instantiation [3]. Indeed, the key tags are bilinear maps operations as in BF-IBE. Hence, we claim that if the tags are not secure then there exists a polynomial-bounded adversary $A$ that breaks the security of BF-IBE, contradicting our assumption. Finally, the re-use of randomness $z$ (as described in Figure 5) has been proved to be CPA-secure in [5], thus we skip the entire proof for space limitation. □