Computers & Security 135 (2023) 103486

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Proof of retrievability with flexible designated verification for cloud storage

Xiao Tan [a,b,*], Qi Xie [a,b], Lidong Han [a,b], Shengbao Wang [a,b], Wenhao Liu [a,b]

[a] School of Information Science and Technology, Hangzhou Normal University, Hangzhou, China
[b] Key Laboratory of Cryptography Technology of Zhejiang Province, Hangzhou, China

ARTICLE INFO

Keywords:
Cloud storage
Proof of retrievability
Designated verification
Flexibility and scalability
Security model

ABSTRACT

With numerous applications in cloud storage, Proof of Retrievability (PoR) is an efficient solution for verification and retrieval of big data on a remote server. Existing PoR schemes can support two mutually exclusive verification modes, namely either private verifiability or public verifiability, depending on whether the outsourced data is verifiable by the data owner (user) only or by everyone. In order to allow the user to flexibly designate the capability of verification, we propose a novel framework for PoR schemes that enables fine-grained control of remote data verification for cloud storage. Our idea is enabling the user to dynamically release some tokens (verification keys) to one or more third parties (designated verifiers), such that only the user and the designated verifiers can verify and retrieve the outsourced data from the server. We formalize this notion as PoR with Flexible Designated Verification (PoR-FDV), and define an extended adversarial model for PoR-FDV based on Juel et al.'s PoR model. Under the new framework, we propose a generic, simple and elegant construction of PoR-FDV from public verifiable PoR and identity-based KEM/DEM, which then gives birth to a concrete and efficient PoR-FDV scheme. Compared to existing PoR schemes, PoR-FDV provides higher flexibility and scalability, as well as many useful features.

1. Introduction

In the era of information explosion, people and organizations with limited resources are more likely to outsource large dataset to storage service providers (SSP) who should keep the data reliably for users. Nowadays the most popular choices of SSP are some emerging cloud storage systems, such as Amazon S3, Google Drive, Dropbox. Cloud storage outperforms many traditional storage technologies since it can utilize the facilitates in cloud to offer hosted services that allow access of the outsourced data at anytime and from anywhere. A critical security issue of cloud storage is to guarantee that the user's data will never be discarded or modified by the untrusted cloud server. To solve this problem, generally we should allow the user to do data verification, in particular, by running an auditing protocol with the server to check the data integrity timely. For capability of processing big data, this auditing protocol is expected to be extremely efficient: (1) the user's local storage is very small in comparison to the scale of outsourced data, (2) the storage server needs not to read every block of data for the audit, and (3) it should be avoided to transmit the entire data in the communication. The above requirements imply the exclusion of conventional integrity checking solutions, e.g. keeping a local copy of data or downloading the whole data from the server.

In the recent years, researchers proposed numerous solutions for remote data integrity checking, including some storage systems of formalized models and provable security, in which there are two widely studied primitives called Provable Data Possession (PDP) Ateniese et al. (2007, 2008); Hanser and Slamanig (2013); Wang (2013); Wang et al. (2019); Guo et al. (2020a); Chen et al. (2020); Zhou et al. (2021) and Proof of Retrievability (PoR) Juels and Kaliski (2007); Shacham and Waters (2008); Wang et al. (2009, 2010, 2011); Xu and Chang (2012); Yuan and Yu (2013); Yang and Jia (2013); Binanda and Sushmita (2017); Chen and Chang (2022); Mishra et al. (2020); Susilo et al. (2022); Lu et al. (2022); Ji et al. (2023). Generally speaking, the two primitives conduct an efficient challenge-response protocol between the server and verifier in audit, by sampling a small fraction of data blocks for integrity checking. By comparison, PDP ensures that the server has to store most of the data to pass the verification, while PoR guarantees that the server has to maintain the knowledge of all the data. In other words, PoR is a stronger security notion which provides the feature called *retrievability* that allows the verifier to fully recover the data if

* Corresponding author.
E-mail addresses: xiaotan@hznu.edu.cn (X. Tan), qixie68@126.com (Q. Xie), ldhan@hznu.edu.cn (L. Han), googlewhat@qq.com (S. Wang), whl819_819@163.com (W. Liu).

https://doi.org/10.1016/j.cose.2023.103486
Received 2 December 2022; Received in revised form 17 April 2023; Accepted 16 September 2023
Available online 21 September 2023

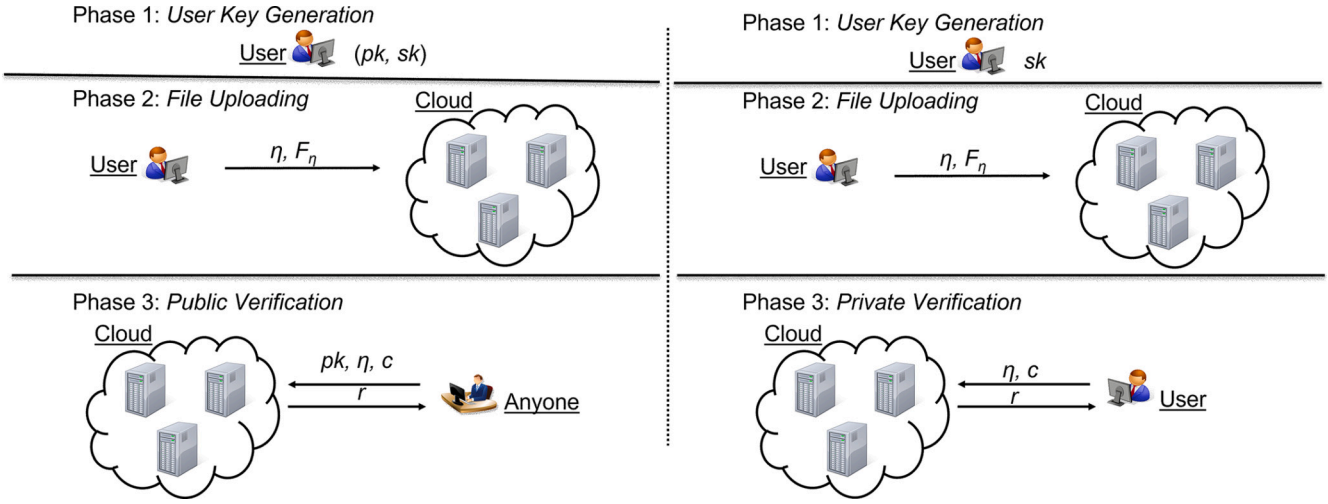

Fig. 1. PoR with public verifiability / private verifiability.

necessary. Technically, it is achieved by applying a redundant encoding scheme to preprocess the data before storage, such that it is sufficient to enforce recovery of the data from a certain fraction. As a trade-off, PoR usually requires for a higher storage cost on the server than PDP, due to the larger data size caused by encoding.

By the mode of data verification, the existing PDP or PoR schemes can be classified into two types: the schemes supporting *private verifiability* (PRV for short) in which only the user can verify the outsourced data, or the schemes supporting *public verifiability* (PUV for short) in which the outsourced data is verifiable by the public. Fig. 1 shows the mechanism of these two verification modes, and the major differences between them are: (1) in the key generation phase, a public key pair (pk, sk) is produced for PUV while a private key sk for PRV, where the keys are used for encoding the raw file, and sk in either PUR or PRV is kept secret by the user; (2) to audit the encoded file F_η (η is the file handler) in the verification phase, PRV requires for the knowledge of sk to verify the response r of the challenge c, while PUV just refers to pk as input which is available to everyone.

When dealing with sensitive data, obviously PRV is preferred, as PUV allows anyone to audit and retrieve the data and thus break the data privacy. On the other hand, PUV is more suitable for users with lightweight devices of limited resources, as it allows a third-party auditor to do the verification on behalf of the user. Unfortunately, the two verification modes are mutually exclusive, namely no existing PDP or PoR allows the user to *flexibly designate the capability of verification*, which is a desirable property for releasing the burden of verification from the user side and simultaneously keeping the data verifiable and retrievable by some trusted verifiers. Consider the following application scenario shown in Fig. 2: a project manager wants to share some collaborative documents among the team members by uploading the files to cloud. The members should be capable of running the challenge-response protocol to retrieve the documents when necessary. PUV is not a good solution as any outsider (i.e. other person not in the team) can also access the file contents; neither is PRV, in which only the manager who uploaded the files can recover the data. Therefore, this example cannot be trivially solved using traditional PUV or PRV, demanding for the proposal of a novel verification mode.

To the best of our knowledge, only three related works Hanser and Slamanig (2013); Wang (2013); Yang et al. (2021) attempted to solve the above issue. In Hanser and Slamanig (2013), the goal is to enable both private verifiability and public verifiability for different users in the system. The proposed construction allows the data owner to do private verification, but anyone else has to run a public verification algorithm. Essentially, it is a public verifiable PDP scheme that offers higher verification efficiency on the user side. Their scheme does not

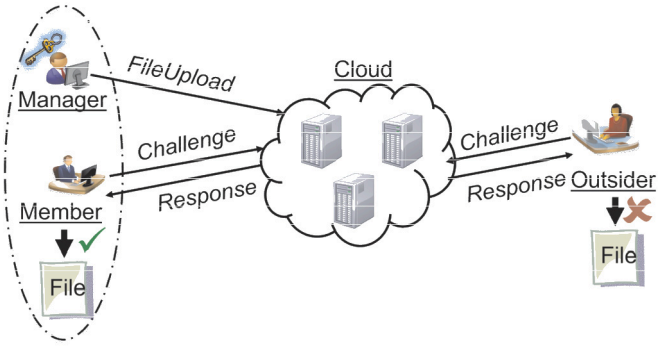

Fig. 2. One application scenario of delegated verification.

really support the delegation of verification capability since everyone is able to audit and learn the data without any credential from the user. Wang (2013) proposed a more meaningful approach called *proxy provable data possession* (PPDP), by enabling designation of verification capability to a proxy in public cloud. In particular, besides the user herself, only the proxy who has the delegation warrant of the user can audit the data on the server. However, PPDP has the following disadvantages: (1) not only the user, the proxy and the server must also have a public/private key pair, which might be impossible in the setting of many applications; (2) the metadata or tags for verification must be generated based upon the public key of the proxy and the public key of the server, implying that the proxy must be pre-appointed before uploading the data to the specified server, and that the size of metadata has linear growth to the number of proxies. In Yang et al. (2021), the user can generate its own key together with verification keys which could be delegated to any third party auditor, and also possesses the capability of updating the verification keys at the cost of refreshing authentication tags. However, the verification keys must be used in conjunction with the user's private key to generate authentication tags in the data preprocessing phase. Furthermore, it also suffers from the similar problem as Wang (2013), since the size of authentication tags is linear to the number of delegated verifiers. Some other related works about PoR in recent years either focus on dynamic update operations Chen et al. (2020); Binanda and Sushmita (2017); Ren et al. (2015); Anthoine et al. (2021) or replication Guo et al. (2020a,b), or applications in emerging scenarios such as medical storage systems Zhou et al. (2021); Li et al. (2022). As we can see, an appropriate solution is still on demand for realizing flexible delegation of auditing capabilities in cloud storage.

Our Results. In this paper, we propose a novel framework for PoR schemes that offers fine-grained control on the auditing capability of re-

Table 1
Delegation flexibility comparison of related works.

Schemes	$\mathbf{DV_{AP}}$	$\mathbf{DV_{AT}}$	NA-SA	NA-SC
Hanser and Slamanig (2013)	×	×	√	√
Wang (2013)	√	×	×	×
Yang et al. (2021)	√	×	√	×
PoR-FDV	√	√	√	√

a $\mathbf{DV_{AP}}$ and $\mathbf{DV_{AT}}$ denotes that the verification capability can be delegated to any third party, and at any time (before or after the data uploading phase), respectively.
b **NA-SA** and **NA-SC** denotes that the proposed scheme requires for no additional setup assumption, and no additional storage cost, respectively, when compared to traditional PDP or PoR schemes.

mote data on a storage server. In particular, the user can dynamically release a *verification key* to any third party, namely a *designated verifier*, such that only the designated verifiers and the user herself can audit and retrieve the outsourced data. We formalize this notion as *Proof of Retrievability with Flexible Designated Verification* (PoR-FDV), and propose a new adversarial model for PoR-FDV based on Juel *et al.*'s PoR model. PoR-FDV provides higher flexibility and scalability than existing PDP/PoR schemes, and enjoys plenty of advantages listed as below, where the useful features ($\mathbf{DV_{AP}}$, $\mathbf{DV_{AT}}$, $\mathbf{NA_{SA}}$, $\mathbf{NA_{SC}}$) are defined in Table 1.

(1) PoR-FDV allows dynamic designation of verification capability ($\mathbf{DV_{AP}}$), no matter before or after file uploading ($\mathbf{DV_{AT}}$);
(2) Compared to PDP or PoR, PoR-FDV requires no additional setup assumption ($\mathbf{NA_{SA}}$);
(3) In PoR-FDV, the growth in number of designated verifiers incurs no increase in storage cost ($\mathbf{NA_{SC}}$).

Table 1 shows the comparison results of our proposed PoR-FDV with the related works Hanser and Slamanig (2013); Wang (2013); Yang et al. (2021) regarding delegation flexibility. For the rest of this paper, we first review some cryptographic tools in Section 2, then formally define the new framework of PoR-FDV in Section 3. Applying the cryptographic tools as components, we propose a generic construction of PoR-FDV in Section 4, which gives birth to an efficient concrete PoR-FDV scheme in Section 5. The performance analysis show that our PoR-FDV scheme preserves comparable efficiency to the conventional PoR schemes.

2. Preliminary

2.1. Proof of retrievability with public verifiability

The formal definition of Proof of Retrievability (PoR) scheme was proposed by Juels *et al.* in Juels and Kaliski (2007). Roughly speaking, PoR is an efficient challenge-response auditing protocol, which enables compact proofs that the user's data is kept intact on the storage server and fully recoverable by the user. In this paper, we focus on *stateless* PoR scheme in the public key setting, where "stateless" refers to that no long-term state information is required to maintain or update between the sessions of auditing. This property is quite desirable in the applications where the user prefers to designate the verification capability to a third party or parties. PoR with public verifiability (PPoR for short) can be viewed as a special case in which the capability of verification is given to the public.

Based on Juel *et al.*'s formalization, a PPoR scheme consists of six algorithms (keyGen, encode, challenge, respond, verify, extract):

- $(pk, sk) \leftarrow \mathsf{keyGen}(1^k)$. The key generation algorithm takes a security parameter $k \in \mathbb{N}$ as input, outputs a public key pk along with the corresponding private key sk as the key pair of the user.
- $(\eta, F_\eta) \leftarrow \mathsf{encode}(sk, F)$. The file encoding algorithm takes sk and a file F as input, outputs a file handler η and the encoded file F_η.
- $c \leftarrow \mathsf{challenge}(pk, \eta)$. The challenge generation algorithm takes pk and a file handler η as input, outputs a challenge c.
- $r \leftarrow \mathsf{respond}(pk, \eta, c)$. The response computation algorithm takes pk, η, and a challenge c as input, outputs a response r.
- $1/0 \leftarrow \mathsf{verify}(pk, \eta, r, c)$. The proof verification algorithm takes pk, η, r, c as input, outputs 1 (accept) if r is a valid response to the challenge c; otherwise outputs 0 (reject).
- $F_\eta \leftarrow \mathsf{extract}(pk, \eta)$. The file extract algorithm takes pk and η as input, outputs the encoded file F_η. This algorithm is allowed to access an oracle $\mathcal{O}_{\mathsf{respond}}$ which outputs responses to any challenge under pk and η.

The *correctness* of PPoR scheme requires that for any key pair $(pk, sk) \leftarrow \mathsf{keyGen}(1^k)$ and challenge $c \leftarrow \mathsf{challenge}(pk, \eta)$ where $(\eta, F_\eta) \leftarrow \mathsf{encode}(sk, F)$, we have:

- If $r \leftarrow \mathsf{respond}(pk, \eta, c)$, then $\mathsf{verify}(pk, \eta, r, c) = 1$;
- If $F'_\eta \leftarrow \mathsf{extract}(pk, \eta)$, then $F'_\eta = F_\eta$.

The adversarial model for PPoR are defined as the following two experimental games, where the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ plays the role of a storage server whose goal is to compute valid responses to the challenge under a user's public key pk and a file handler η^*.

Experiment $\mathbf{Exp}^{\mathsf{Setup}}_{\mathcal{A},\mathsf{PPoR}}(1^k)$:
$(pk, sk) \leftarrow \mathsf{keyGen}(1^k)$;
$(\delta, \eta^*) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pk)$;
Output (δ, η^*).

Experiment $\mathbf{Exp}^{\mathsf{Challenge}}_{\mathcal{A},\mathsf{PPoR}}(\delta, \eta^*)$:
$c^* \leftarrow \mathcal{O}_{\mathsf{challenge}}(pk, \eta^*)$;
$r^* \leftarrow \mathcal{A}_2(\delta, \eta^*, c^*)$;
$b \leftarrow \mathcal{O}_{\mathsf{verify}}(pk, \eta^*, r^*, c^*)$;
Output $b \in \{0, 1\}$.

$\mathcal{A}_1^{\mathcal{O}}$ denotes that $\mathcal{A}_1$ has access to the set of oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{encode}}, \mathcal{O}_{\mathsf{challenge}}, \mathcal{O}_{\mathsf{verify}}, \mathcal{O}_{\mathsf{extract}}\}$:

- $\mathcal{O}_{\mathsf{encode}}(sk, \cdot)$: Given a file F, the oracle outputs $(\eta, F_\eta) \leftarrow \mathsf{encode}(sk, F)$;
- $\mathcal{O}_{\mathsf{challenge}}(pk, \cdot)$: Given a file handler η, the oracle outputs $c \leftarrow \mathsf{challenge}(pk, \eta)$;
- $\mathcal{O}_{\mathsf{verify}}(pk, \cdot)$: Given a file handler η and a challenge-response pair (c, r), the oracle outputs $1/0 \leftarrow \mathsf{verify}(pk, \eta, r, c)$;
- $\mathcal{O}_{\mathsf{extract}}(pk, \cdot)$: Given a file handler η, the oracle outputs $F_\eta \leftarrow \mathsf{extract}(pk, \eta)$.

The adversary is allowed to pass its state information δ from $\mathcal{A}_1$ to $\mathcal{A}_2$. Then we are ready for the security definition of PPoR scheme:

Definition 1. A PPoR scheme is said to achieve (ϵ, λ)-valid proof of retrievability, if for any probabilistic polynomial time λ-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined as:

$$\Pr[\boldsymbol{Exp}^{Challenge}_{\mathcal{A},PPoR}(\delta, \eta^*) = 1 | (\delta, \eta^*) \leftarrow \boldsymbol{Exp}^{Setup}_{\mathcal{A},PPoR}(1^k)] \geq \lambda,$$

we have:

$$\Pr[F = F_{\eta^*} | F \leftarrow \boldsymbol{extract}^{\mathcal{A}_2(\delta,\cdot)}(pk, \eta^*)] \geq \epsilon,$$

where $\mathcal{A}_2(\delta, \cdot)$ is casted as the oracle $\mathcal{O}_{respond}$ for the *respond* algorithm.

In the above definition, ϵ and λ are polynomials of the security parameter k, indicating that if the adversary $\mathcal{A}$ can output valid response with noticeable probability λ, then the probability ϵ that we can extract the file F_{η^*} from $\mathcal{A}$ is also non-negligible.

2.2. Identity-based key encapsulation

Key encapsulation mechanism is one of the two basic operations KEM & DEM in *hybrid encryption*, a notion formalized by Cramer and Shoup Cramer and Shoup (2003) for realizing practical encryption of long messages when no pre-shared key is available. Informally, hybrid encryption uses a key encapsulation mechanism (KEM) to encapsulate a random key, and uses a data encapsulation mechanism (DEM) to encrypt the message with the key. The ciphertext consists of two parts: the encapsulation of the key and the encryption of the message. To recover the plaintext, the decrypter first decapsulates the key, then uses the key to decrypt the message. When it comes to implementation, KEM and DEM can be instantiated by a public key encryption and a one-time symmetric key encryption respectively. The efficiency gain stems from the fact that public key encryption is generally more expensive in computational cost than symmetric encryption, and the encapsulated key is much shorter than the message. Later, Bentahar *et al.* Bentahar et al. (2008) extended KEM to the identity based setting, and proved that the combination of an identity based KEM (IB-KEM for short) and a standard DEM gives birth to an identity based hybrid encryption.

An IB-KEM can be specified by the following four algorithms:

- $(mpk, msk) \leftarrow \mathsf{setup}(1^k)$. The setup algorithm takes a security parameter $k \in \mathbb{N}$ as input, outputs the master public key mpk (or referred to as system parameters) along with the master secret key msk.
- $d_{\mathsf{ID}} \leftarrow \mathsf{extract}(mpk, msk, \mathsf{ID})$. The key extraction algorithm takes mpk, msk and an identity $\mathsf{ID} \in \{0,1\}^*$ as input, outputs a decryption key d_{ID} for ID.
- $(K, E) \leftarrow \mathsf{encap}(mpk, \mathsf{ID})$. The encapsulation algorithm takes mpk and ID as input, outputs a pair (K, E), where K is a key for symmetric encryption and E is the encapsulation of K.
- $K \leftarrow \mathsf{decap}(mpk, \mathsf{ID}, d_{\mathsf{ID}}, E)$. The decapsulation algorithm takes mpk, ID, a decryption key d_{ID} and a key encapsulation E as input, outputs the decapsulted key K or a failure symbol $\perp$.

The *correctness* of IB-KEM scheme requires that for any $(mpk, msk) \leftarrow \mathsf{setup}(1^k)$ and $(K, E) \leftarrow \mathsf{encap}(mpk, \mathsf{ID})$, we have $K' = K$ if $K' \leftarrow \mathsf{decap}(mpk, \mathsf{ID}, d_{\mathsf{ID}}, E)$ where $d_{\mathsf{ID}} \leftarrow \mathsf{extract}(mpk, msk, \mathsf{ID})$.

The indistinguishable security of IB-KEM against adaptive identity chosen ciphertext attacks (IND-ID-CCA) can be formalized as the following experiments, where the only restriction for the adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is that $\mathcal{B}$ can not extract the private key of the target identity ID^*, or directly obtain the decapsulated key of the challenge (ID^*, E^*).

Experiment $\mathbf{Exp}^{\mathrm{Setup}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(1^k)$:

$(mpk, msk) \leftarrow \mathsf{setup}(1^k)$;

$(\delta, \mathsf{ID}^*) \leftarrow \mathcal{B}_1^{\mathcal{O}}(mpk)$;

Output (δ, ID^*).

Experiment $\mathbf{Exp}^{\mathrm{Challenge}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(\delta, \mathsf{ID}^*)$:

$b \leftarrow_{\$} \{0,1\}$;

$(K^*, E^*) \leftarrow \mathcal{O}_{\mathsf{encap}}(mpk, \mathsf{ID}^*)$;

If $b = 0$: $K^\dagger \leftarrow_{\$} \mathcal{K}$;

If $b = 1$: $K^\dagger \leftarrow K^*$;

$b' \leftarrow \mathcal{B}_2(\delta, \mathsf{ID}^*, K^\dagger, E^*)$;

Output $b' \in \{0,1\}$.

$\mathcal{B}_1^{\mathcal{O}}$ denotes that $\mathcal{B}_1$ has access to the set of oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{extract}}, \mathcal{O}_{\mathsf{decap}}\}$:

- $\mathcal{O}_{\mathsf{extract}}(mpk, msk, \cdot)$: Given an identity ID, the oracle outputs $d_{\mathsf{ID}} \leftarrow \mathsf{extract}(mpk, msk, \mathsf{ID})$;
- $\mathcal{O}_{\mathsf{decap}}(mpk, \cdot)$: Given an identity ID and a key encapsulation E, the oracle outputs $K \leftarrow \mathsf{decap}(mpk, \mathsf{ID}, d_{\mathsf{ID}}, E)$.

The adversarial state δ can be passed from $\mathcal{B}_1$ to $\mathcal{B}_2$. Let $\mathcal{Q}_{\mathsf{extract}}$ and $\mathcal{Q}_{\mathsf{decap}}$ denote the set of queries made by $\mathcal{B}$ to $\mathcal{O}_{\mathsf{extract}}$ and $\mathcal{O}_{\mathsf{decap}}$ respectively, then we can give the security definition of IB-KEM scheme:

Definition 2. An IB-KEM scheme is said to achieve ϵ-IND-ID-CCA security, if for any probabilistic polynomial time adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, we have:

$$\Pr[b' = b | b' \leftarrow \boldsymbol{Exp}^{Challenge}_{\mathcal{B},IB\text{-}KEM}(\delta, ID^*)] \leq \frac{1}{2} + \epsilon,$$

where $(\delta, ID^*) \leftarrow \boldsymbol{Exp}^{Setup}_{\mathcal{B},IB\text{-}KEM}(1^k)$, under the restriction that $ID^* \notin \mathcal{Q}_{extract}$, $(ID^*, E^*) \notin \mathcal{Q}_{decap}$.

Notice that the definition above is for non-adaptive chosen ciphertext security, which is a weaker notion than adaptive chosen ciphertext security (IND-ID-CCA2). However, an IB-KEM scheme achieving this weaker security is sufficient for our construction given in Section 4.

The security for IB-KEM scheme against selective-ID (non-adaptive) chosen ciphertext attack (IND-sID-CCA) can be defined in a similar way, where the adversary should determine the target identity ID^* before seeing the master public key mpk in $\mathbf{Exp}^{\mathrm{Setup}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(1^k)$.

2.3. One-time symmetric key encryption

As mentioned in the previous section, one-time symmetric key encryption (OT-SKE for short) is a widely applied tool for implementing a standard DEM. Formally, a OT-SKE scheme can be specified by the following two deterministic algorithms:

- $C \leftarrow \mathsf{encrypt}(1^k, K, M)$. The encryption algorithm takes a security parameter $k \in \mathbb{N}$, a key K, and a message M as input, outputs a ciphertext C.
- $M \leftarrow \mathsf{decrypt}(1^k, K, C)$. The decryption algorithm takes a security parameter $k \in \mathbb{N}$, a key K, and a ciphertext C as input, outputs a message M or a failure symbol $\perp$.

The *correctness* of OT-SKE scheme requires that for any $k \in \mathbb{N}$, key K and message M, we have $M' = M$ where $M' \leftarrow \mathsf{decrypt}(1^k, K, \mathsf{encrypt}(1^k, K, M))$.

The indistinguishability security of OT-SKE against passive attacks (IND-PA) can be formalized as the following two experiments.

Experiment $\mathbf{Exp}^{\mathrm{Setup}}_{C,\mathsf{OT\text{-}SKE}}(1^k)$:

$(\delta, M_0, M_1) \leftarrow C_1(1^k)$;

Output (δ, M_0, M_1).

Experiment $\mathbf{Exp}^{\mathrm{Challenge}}_{C,\mathsf{OT\text{-}SKE}}(\delta, M_0, M_1)$:

$b \leftarrow_{\$} \{0,1\}$;

$K^* \leftarrow_{\$} \mathcal{K}$;

$C^* \leftarrow \mathcal{O}_{\mathsf{encrypt}}(1^k, K^*, M_b)$;

$b' \leftarrow C_2(\delta, C^*, M_0, M_1)$;

Output $b' \in \{0,1\}$.

The adversarial state δ can be passed from C_1 to C_2. Notice that $K^* \leftarrow_{\$} \mathcal{K}$ just requires that K^* is randomly distributed in the key space, and in practice K^* usually is generated by a key derivation function. For instance, if K^* is the partial output of IB-KEM.encap, then C_2 can also take the encapsulation E^* as an additional input. Then we can give the security definition of OT-SKE scheme:

Definition 3. A OT-SKE scheme is said to achieve ϵ-IND-PA security, if for any probabilistic polynomial time adversary $\mathcal{C} = (C_1, C_2)$, we have:

$$\Pr[b' = b | b' \leftarrow \boldsymbol{Exp}^{Challenge}_{C,OT\text{-}SKE}(\delta, M_0, M_1)] \leq \frac{1}{2} + \epsilon,$$

where $(\delta, M_0, M_1) \leftarrow \boldsymbol{Exp}^{Setup}_{C,OT\text{-}SKE}(1^k)$.

The definition above is a weaker notion than indistinguishable security against chosen ciphertext attacks (IND-CCA) in which C_2 can access an oracle for decrypting ciphertexts under the key K^* except the challenge ciphertext

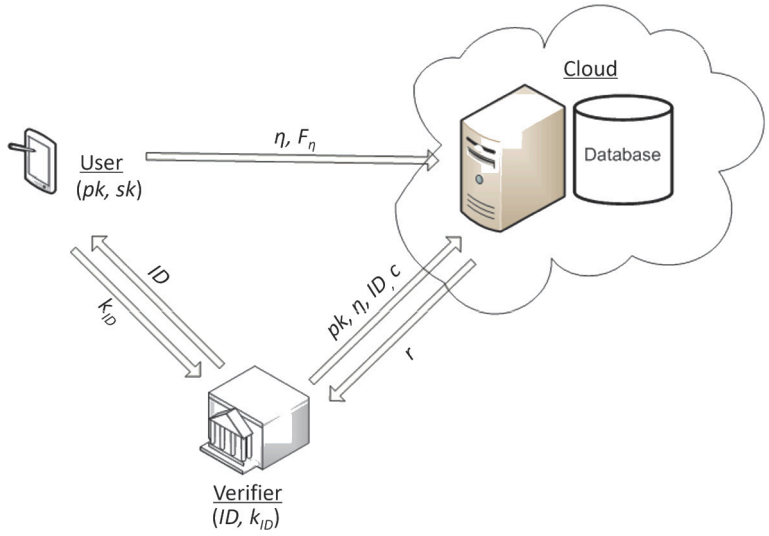

Fig. 3. System model of identity-based PoR-FDV.

C^*. However, a OT-SKE scheme achieving IND-PA security is sufficient for our PoR-FDV construction in Section 4.

A folklore construction of OT-SKE with IND-PA security is as follows: given a key $K \in \mathcal{K}$, a message $M \in \mathcal{M}$, and a pseudo-random bit generator $\mathcal{G} : \mathcal{K} \to \mathcal{M}$, it encrypts the message as $C = \mathcal{G}(K) \oplus M$. In other words, it expands the key K and works as a one-time pad.

3. Definition: PoR-FDV

3.1. System model

In the system model of PoR-FDV, there are three types of participants involved: users, verifiers and cloud server. Users are data owners who upload files onto cloud server after preprocessing, while verifiers are the entities designated by users to audit the data, by running a challenge-response PoR protocol with cloud server. Each user possesses a public key pair (pk, sk) for encoding the file and generating verification keys k_{ID} for verifiers with identity ID. Fig. 3 presents the system model, in which F_η is the encoded file with the file handler η, and r is the proof generated by cloud server as response to the challenge c.

3.2. Algorithms

A PoR-FDV scheme consists of seven algorithms (keyGen, encode, deriveVerKey, challenge, respond, verify, extract):

- $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{keyGen}(1^k)$. The key generation algorithm takes a security parameter $k \in \mathbb{N}$ as input, outputs a public key $\tilde{pk}$ along with the corresponding private key $\tilde{sk}$ as the key pair of the user.
- $(\eta, F_\eta) \leftarrow \mathsf{encode}(\tilde{sk}, F)$. The file encoding algorithm takes $\tilde{sk}$ and a file F as input, outputs a file handler η and the encoded file F_η with some metadata in it. Notice that the encoded file has no dependence on any verifier's identity or the number of verifiers in the system.
- $k_{\mathsf{ID}} \leftarrow \mathsf{deriveVerKey}(\tilde{pk}, \tilde{sk}, \mathsf{ID})$. The verification key derivation algorithm takes $\tilde{pk}$, $\tilde{sk}$ and the identity ID of a verifier as input, outputs a verification key k_{ID} for the verifier.
- $\tilde{c} \leftarrow \mathsf{challenge}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$. The challenge generation algorithm takes $\tilde{pk}$, a file handler η, an identity ID and a verification key k_{ID} as input, outputs a challenge $\tilde{c}$.
- $\tilde{r} \leftarrow \mathsf{respond}(\tilde{pk}, \eta, \mathsf{ID}, \tilde{c})$. The response generation algorithm takes $\tilde{pk}$, η, ID and a challenge $\tilde{c}$ as input, outputs a response $\tilde{r}$.
- $1/0 \leftarrow \mathsf{verify}(\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}})$. The proof verification algorithm takes $\tilde{pk}$, η, $\tilde{r}$, $\tilde{c}$, and a verification key k_{ID} as input, outputs 1 (accept) if $\tilde{r}$ is a valid response to the challenge $\tilde{c}$; otherwise outputs 0 (reject).
- $F_\eta \leftarrow \mathsf{extract}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$. The file extract algorithm takes $\tilde{pk}$, η, ID and k_{ID} as input, outputs the encoded file F_η. This algorithm has access to an oracle $\tilde{\mathcal{O}}_{\mathsf{respond}}$ which outputs responses to any challenge under $\tilde{pk}$, η, ID.

A PoR-FDV system works in three phases as below, of which the framework is clarified by Fig. 4. Notice that verifier designation can be either executed before or after file uploading, as in Phase 2- or in Phase 2+. And in Phase 3, the designated verification can be executed for unlimited number of times.

- **Phase 1**: User Key Generation
  The user (data owner) runs the key generation algorithm to get her key pair $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{keyGen}(1^k)$.
- **Phase 2-**: Verifier Designation
  For any third party $\mathcal{T}$ with identity $\mathsf{ID}_\mathcal{T}$, the user can adaptively issues a verification key by running $k_{\mathsf{ID}_\mathcal{T}} \leftarrow \mathsf{deriveVerKey}(\tilde{pk}, \tilde{sk}, \mathsf{ID}_\mathcal{T})$.
- **Phase 2**: File Uploading
  To outsource a file $F = \{m_1, \cdots, m_n\}$ onto the server, the user first encodes the file as $(\eta, F_\eta) \leftarrow \mathsf{encode}(\tilde{sk}, F)$, then sends $\{\tilde{pk}, \eta, F_\eta\}$ to the server for storage. Given a file handler η, the server can efficiently read the blocks of the encoded file F_η.
- **Phase 2+**: Verifier Designation
  The same as Phase 2-.
- **Phase** 3: Designated Verification
  To check the integrity of a file with file handler η, a designated verifier $\mathcal{T}$ with identity $\mathsf{ID}_\mathcal{T}$ performs a challenge-response auditing protocol with the server as follows:
  1. $\mathcal{T}$ generates a challenge $\tilde{c} \leftarrow \mathsf{challenge}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, k_{\mathsf{ID}_\mathcal{T}})$, and sends $(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, \tilde{c})$ to the server.
  2. The server runs $\tilde{r} \leftarrow \mathsf{respond}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, \tilde{c})$, and responds with $\tilde{r}$ to $\mathcal{T}$.
  3. Finally, $\mathcal{T}$ checks the validity of $\tilde{r}$ by running $\mathsf{verify}(\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}_\mathcal{T}})$.

  When necessary, $\mathcal{T}$ can also retrieve the file $F_\eta \leftarrow \mathsf{extract}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, k_{\mathsf{ID}_\mathcal{T}})$ by interacting with the server who serves as the oracle $\tilde{\mathcal{O}}_{\mathsf{respond}}$. The retrieval process consists of multiple runnings of the auditing protocol on the same pk, η and $\mathsf{ID}_\mathcal{T}$.

The *correctness* of PoR-FDV scheme requires that for any key pair $(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{keyGen}(1^k)$, any verification key $k_{\mathsf{ID}_\mathcal{T}} \leftarrow \mathsf{deriveVerKey}(\tilde{pk}, \tilde{sk}, \mathsf{ID}_\mathcal{T})$, and any challenge $\tilde{c} \leftarrow \mathsf{challenge}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, k_{\mathsf{ID}_\mathcal{T}})$ where $(\eta, F_\eta) \leftarrow \mathsf{encode}(\tilde{sk}, F)$, we have:

- If $\tilde{r} \leftarrow \mathsf{respond}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, \tilde{c})$, then $\mathsf{verify}(\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}_\mathcal{T}}) = 1$;
- If $F'_\eta \leftarrow \mathsf{extract}(\tilde{pk}, \eta, \mathsf{ID}_\mathcal{T}, k_{\mathsf{ID}_\mathcal{T}})$, then $F_\eta = F'_\eta$.

3.3. Security model

Generally speaking, there are two basic security requirements for PoR-FDV. The first is *proof of retrievability* originated from that of PPoR, and the difference is that proof of retrievability of PoR-FDV additionally takes the identity of designated verifiers as a parameter in the computation, implying that only the designated verifiers (and the user herself) can retrieve the file by running the auditing protocol. The second is *soundness of designated verification*, refer-

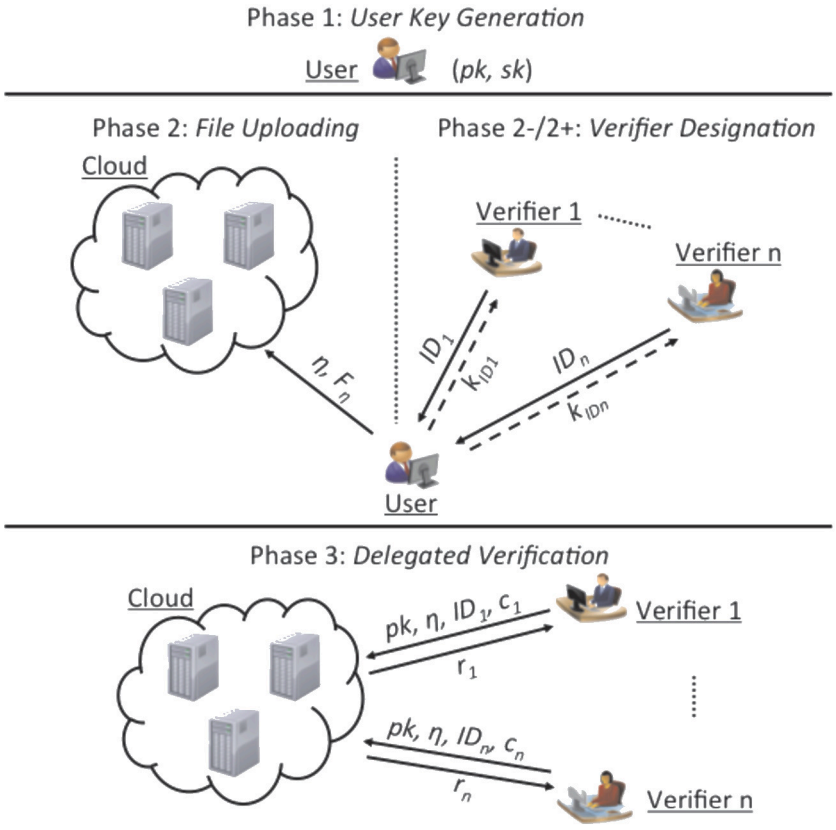

Fig. 4. Framework of identity-based PoR-FDV.

ring to that others without verification keys have no capability of verifying the proofs. Both the security notions are considered under the adaptive-ID adversarial model, in which the adversary can choose the target identity ID^* after seeing the system parameters and having a learning phase by access to some oracles.

Firstly, to formalize the notion of proof of retrievability of PoR-FDV, we define the following two experimental games, where the adversary $\tilde{\mathcal{A}} = (\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2)$ plays the role of a storage server whose goal is to compute valid responses to the challenges under a public key $\tilde{pk}$ and a file handler η^* from a designated verifier with identity ID^*.

Experiment $\mathbf{Exp}^{\mathrm{Setup}}_{\tilde{\mathcal{A}},\text{PoR-FDV}}(1^k)$:

$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{keyGen}(1^k)$;

$(\tilde{\delta}, \eta^*, \mathsf{ID}^*) \leftarrow \tilde{\mathcal{A}}_1^{\tilde{\mathcal{O}}}(\tilde{pk})$;

Output $(\tilde{\delta}, \eta^*, \mathsf{ID}^*)$.

Experiment $\mathbf{Exp}^{\mathrm{Challenge}}_{\tilde{\mathcal{A}},\text{PoR-FDV}}(\tilde{\delta}, \eta^*, \mathsf{ID}^*)$:

$k_{\mathsf{ID}^*} \leftarrow \tilde{\mathcal{O}}_{\mathsf{deriveVerKey}}(\tilde{pk}, \tilde{sk}, \mathsf{ID}^*)$;

$\tilde{c}^* \leftarrow \tilde{\mathcal{O}}_{\mathsf{challenge}}(\tilde{pk}, \eta^*, \mathsf{ID}^*, k_{\mathsf{ID}^*})$;

$\tilde{r}^* \leftarrow \tilde{\mathcal{A}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$;

$\tilde{b} \leftarrow \tilde{\mathcal{O}}_{\mathsf{verify}}(\tilde{pk}, \eta^*, \tilde{r}^*, \tilde{c}^*, k_{\mathsf{ID}^*})$;

Output $\tilde{b} \in \{0,1\}$.

$\tilde{\mathcal{A}}_1^{\tilde{\mathcal{O}}}$ denotes that $\tilde{\mathcal{A}}_1$ has access to the set of oracles $\tilde{\mathcal{O}} = \{\tilde{\mathcal{O}}_{\mathsf{encode}}, \tilde{\mathcal{O}}_{\mathsf{deriveVerKey}}, \tilde{\mathcal{O}}_{\mathsf{challenge}}, \tilde{\mathcal{O}}_{\mathsf{verify}}, \tilde{\mathcal{O}}_{\mathsf{extract}}\}$:

- $\tilde{\mathcal{O}}_{\mathsf{encode}}(\tilde{sk}, \cdot)$: Given a file F, the oracle outputs $(\eta, F_\eta) \leftarrow \mathsf{encode}(\tilde{sk}, F)$;
- $\tilde{\mathcal{O}}_{\mathsf{deriveVerKey}}(\tilde{pk}, \tilde{sk}, \cdot)$: Given an identity ID, the oracle outputs $k_{\mathsf{ID}} \leftarrow \mathsf{deriveVerKey}(\tilde{pk}, \tilde{sk}, \mathsf{ID})$;
- $\tilde{\mathcal{O}}_{\mathsf{challenge}}(\tilde{pk}, \cdot)$: Given a file handler η and an identity ID, the oracle outputs $\tilde{c} \leftarrow \mathsf{challenge}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$;
- $\tilde{\mathcal{O}}_{\mathsf{verify}}(\tilde{pk}, \cdot)$: Given a file handler η, an identity ID, and a challenge-response pair $(\tilde{c}, \tilde{r})$, the oracle outputs $1/0 \leftarrow \mathsf{verify}(\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}})$;
- $\tilde{\mathcal{O}}_{\mathsf{extract}}(\tilde{pk}, \cdot)$: Given a file handler η and an identity ID, the oracle outputs $F_\eta \leftarrow \mathsf{extract}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$.

The state information $\tilde{\delta}$ can be passed from $\tilde{\mathcal{A}}_1$ to $\tilde{\mathcal{A}}_2$. Then the proof of retrievability of PoR-FDV can be defined as follows, for ensuring that the validity of proofs implies the retrievability of data stored on the server.

Definition 4. A PoR-FDV scheme is said to achieve (ϵ, λ)-valid adaptive-ID proof of retrievability, if for any PPT λ-adversary $\tilde{\mathcal{A}} = (\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2)$ defined as:

$$(\tilde{\delta}, \eta^*, ID^*) \leftarrow \boldsymbol{Exp}^{Setup}_{\tilde{\mathcal{A}},PoR\text{-}FDV}(1^k),$$

$$\Pr[\boldsymbol{Exp}^{Challenge}_{\tilde{\mathcal{A}},PoR\text{-}FDV}(\tilde{\delta}, \eta^*, ID^*) = 1] \geq \lambda,$$

we have:

$$\Pr[F_\eta = F_{\eta^*} | F_\eta \leftarrow extract^{\tilde{\mathcal{A}}_2(\tilde{\delta},\cdot)}(\tilde{pk}, \eta^*, ID^*, k_{ID^*})] \geq \epsilon,$$

where $\tilde{\mathcal{A}}_2(\tilde{\delta}, \cdot)$ is casted as the oracle $\tilde{\mathcal{O}}_{respond}$ for the *respond* algorithm.

Then we propose the following experiments to model the soundness of designated verification. Formally, for any PPT adversary who does not know the target verification key k_{ID^*}, it is infeasible to verify the proofs in response to any challenge c^* from the designated verifier with identity ID^*. In particular, the adversary $\tilde{\mathcal{B}} = (\tilde{\mathcal{B}}_1, \tilde{\mathcal{B}}_2)$ in the adversarial games below should be unable to distinguish the response $\tilde{r}^*$ from a random value in the response space $\tilde{\mathcal{R}}$.

Experiment $\mathbf{Exp}^{\mathrm{Setup}}_{\tilde{\mathcal{B}},\text{PoR-FDV}}(1^k)$:

$(\tilde{pk}, \tilde{sk}) \leftarrow \mathsf{keyGen}(1^k)$;

$(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*) \leftarrow \tilde{\mathcal{B}}_1^{\tilde{\mathcal{O}}}(\tilde{pk})$;

Output $(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$.

Experiment $\mathbf{Exp}^{\mathrm{Challenge}}_{\tilde{\mathcal{B}},\text{PoR-FDV}}(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$:

$\tilde{b} \leftarrow_{\$} \{0,1\}$;

If $\tilde{b} = 0$: $\tilde{r}^* \leftarrow_{\$} \tilde{\mathcal{R}}$;

If $\tilde{b} = 1$: $\tilde{r}^* \leftarrow \tilde{\mathcal{O}}_{\mathsf{respond}}(\tilde{pk}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$;

$\tilde{b}' \leftarrow \tilde{\mathcal{B}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{r}^*, \tilde{c}^*)$;

Output $\tilde{b}' \in \{0,1\}$.

$\tilde{\mathcal{B}}_1^{\tilde{\mathcal{O}}}$ denotes that $\tilde{\mathcal{B}}_1$ has access to the set of oracles $\tilde{\mathcal{O}}$ defined in the same way as above. The adversarial state δ can be passed from $\tilde{\mathcal{B}}_1$ to $\tilde{\mathcal{B}}_2$. $\tilde{\mathcal{R}}$ is the response space. Let $\tilde{Q}_{\mathsf{deriveVerKey}}$ and $\tilde{Q}_{\mathsf{verify}}$ denote the set of queries made by $\tilde{\mathcal{B}}$ to

$\tilde{\mathcal{O}}_{\text{deriveVerKey}}$ and $\tilde{\mathcal{O}}_{\text{verify}}$ respectively, then the soundness of designated verification of PoR-FDV scheme can be formalized as follows:

Definition 5. A PoR-FDV scheme is said to achieve ϵ-sound adaptive-ID designated verification (SND-ID-DV), if for any PPT adversary $\tilde{\mathcal{B}} = (\tilde{\mathcal{B}}_1, \tilde{\mathcal{B}}_2)$, we have:

$$\Pr[\tilde{b}' = \tilde{b} | \tilde{b}' \leftarrow \boldsymbol{Exp}^{Challenge}_{\tilde{\mathcal{B}},PoR\text{-}FDV}(\tilde{\delta}, \eta^*, ID^*, \tilde{c}^*)] \leq \frac{1}{2} + \epsilon,$$

where $(\tilde{\delta}, \eta^*, ID^*, \tilde{c}^*) \leftarrow \boldsymbol{Exp}^{Setup}_{\tilde{\mathcal{B}},PoR\text{-}FDV}(1^k)$, under the restriction that $ID^* \notin \tilde{\mathcal{Q}}_{deriveVerKey}$, $(\eta^*, ID^*, \tilde{r}^*, \tilde{c}^*) \notin \tilde{\mathcal{Q}}_{verify}$.

Similarly, we can also define the soundness of selective-ID designated verification (SND-sID-DV), where the adversary should determine the target identity ID^* of the verifier before seeing the public key $\tilde{pk}$ of the user in $\mathbf{Exp}^{\text{Setup}}_{\tilde{\mathcal{B}},\text{PoR-FDV}}(1^k)$.

4. Generic construction of PoR-FDV

We propose a generic construction of PoR-FDV scheme from a PPoR scheme, an IB-KEM scheme and a OT-SKE scheme. Formal proofs are provided to show that security (Definition 4 and 5) of the PoR-FDV scheme are guaranteed by security (Definition 1, 2 and 3) of the underlying PPoR scheme, IB-KEM scheme and OT-SKE scheme.

4.1. The generic construction

$\text{keyGen}(1^k)$:
1. Run $(pk, sk) \leftarrow \text{PPoR.keyGen}(1^k)$;
2. Run $(mpk, msk) \leftarrow \text{IB-KEM.setup}(1^k)$;
3. Output $\tilde{pk} = (pk, mpk)$, $\tilde{sk} = (sk, msk)$.

$\text{encode}(\tilde{sk}, F)$:
1. Run $(\eta, F_\eta) \leftarrow \text{PPoR.encode}(sk, F)$;
2. Output (η, F_η).

$\text{deriveVerKey}(\tilde{pk}, \tilde{sk}, \mathsf{ID})$:
1. Run $d_{\mathsf{ID}} \leftarrow \text{IB-KEM.extract}(mpk, msk, \mathsf{ID})$;
2. Output $k_{\mathsf{ID}} = d_{\mathsf{ID}}$.

$\text{challenge}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$:
1. Run $c \leftarrow \text{PPoR.challenge}(pk, \eta)$;
2. Set $\tilde{c} = c$ and output $\tilde{c}$.

$\text{respond}(\tilde{pk}, \eta, \mathsf{ID}, \tilde{c})$:
1. Set $c = \tilde{c}$ and run $r \leftarrow \text{PPoR.respond}(pk, \eta, c)$;
2. Run $(K, E) \leftarrow \text{IB-KEM.encap}(mpk, \mathsf{ID})$;
3. Run $C \leftarrow \text{OT-SKE.encrypt}(K, r)$;
4. Output $\tilde{r} = (C, E)$.

$\text{verify}(\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}})$:
1. Run $K \leftarrow \text{IB-KEM.decap}(mpk, \mathsf{ID}, k_{\mathsf{ID}}, E)$;
2. Run $r \leftarrow \text{OT-SKE.decrypt}(K, C)$;
3. Set $c = \tilde{c}$ and run $b \leftarrow \text{PPoR.verify}(pk, \eta, r, c)$;
4. Output b.

$\text{extract}(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$: we can get $F_\eta \leftarrow \text{PPoR.extract}(pk, \eta)$ if only the oracle $\mathcal{O}_{\text{respond}}(pk, \cdot)$ for the PPoR scheme can be simulated by the oracle $\tilde{\mathcal{O}}_{\text{respond}}(\tilde{pk}, \cdot)$ for the PoR-FDV scheme. In particular, for each query (η, c_i) to the oracle $\mathcal{O}_{\text{respond}}$, the response r_i is computed as below:
1. Query (η, ID, c_i) to the oracle $\tilde{\mathcal{O}}_{\text{respond}}$, and get the output $\tilde{r}_i = (C_i, E_i)$;
2. Run $K_i \leftarrow \text{IB-KEM.decap}(mpk, \mathsf{ID}, k_{\mathsf{ID}}, E_i)$;
3. Run $r_i \leftarrow \text{OT-SKE.decrypt}(K_i, C_i)$;
4. Return r_i.

4.2. Security proof

Theorem 1. *If the underlying PPoR scheme achieves (ϵ, λ)-valid proof of retrievability, then the proposed PoR-FDV scheme achieves (ϵ, λ)-valid adaptive-ID proof of retrievability.*

Proof. First, we show that given any PPT λ-adversary $\tilde{\mathcal{A}} = (\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2)$ against the PoR-FDV scheme, it implies the existence of a λ-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the PPoR scheme. The detailed construction of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ are as follows.

- $\mathcal{A}_1$: Given pk as input, $\mathcal{A}_1$ runs $(mpk, msk) \leftarrow \text{IB-KEM.setup}(1^k)$, and sets $\tilde{pk} = (pk, mpk)$. Then $\mathcal{A}_1$ calls $\tilde{\mathcal{A}}_1^{\tilde{\mathcal{O}}}(\tilde{pk})$ to get $(\tilde{\delta}, \eta^*, \mathsf{ID}^*)$, sets the adversarial state $\delta = (\tilde{\delta}, \mathsf{ID}^*, \tilde{pk}, msk)$, and outputs (δ, η^*) in the experiment $\mathbf{Exp}^{\text{Setup}}_{\mathcal{A},\text{PPoR}}(1^k)$.

  When $\tilde{\mathcal{A}}_1$ makes queries to the set of oracles $\tilde{\mathcal{O}} = (\tilde{\mathcal{O}}_{\text{encode}}, \tilde{\mathcal{O}}_{\text{deriveVerKey}}, \tilde{\mathcal{O}}_{\text{challenge}}, \tilde{\mathcal{O}}_{\text{verify}}, \tilde{\mathcal{O}}_{\text{extract}})$ for the PoR-FDV scheme, we can utilize $\mathcal{O} = (\mathcal{O}_{\text{encode}}, \mathcal{O}_{\text{challenge}}, \mathcal{O}_{\text{verify}}, \mathcal{O}_{\text{extract}})$ for the PPoR scheme to answer the queries as follows:
  - $\tilde{\mathcal{O}}_{\text{encode}}(\tilde{sk}, \cdot)$: On a query of a file F_i, get (η_i, F_{η_i}) from $\mathcal{O}_{\text{encode}}(sk, F_i)$, and output (η_i, F_{η_i}).
  - $\tilde{\mathcal{O}}_{\text{deriveVerKey}}(\tilde{pk}, \tilde{sk}, \cdot)$: On a query of a verifier identity ID_i, run $d_{\mathsf{ID}_i} \leftarrow \text{IB-KEM.extract}(mpk, msk, \mathsf{ID}_i)$, set $k_{\mathsf{ID}_i} = d_{\mathsf{ID}_i}$ and output k_{ID_i}.
  - $\tilde{\mathcal{O}}_{\text{challenge}}(\tilde{pk}, \cdot)$: On a query of (η_i, ID_i), get c_i from $\mathcal{O}_{\text{challenge}}(pk, \eta_i)$, set $\tilde{c}_i = c_i$ and output $\tilde{c}_i$.
  - $\tilde{\mathcal{O}}_{\text{verify}}(\tilde{pk}, \cdot)$: On a query of $(\eta_i, \mathsf{ID}_i, \tilde{r}_i, \tilde{c}_i)$ where $\tilde{r}_i = (C_i, E_i)$,
    1. run $d_{\mathsf{ID}_i} \leftarrow \text{IB-KEM.extract}(mpk, msk, \mathsf{ID}_i)$, and set $k_{\mathsf{ID}_i} = d_{\mathsf{ID}_i}$;
    2. run $K_i \leftarrow \text{IB-KEM.decap}(mpk, \mathsf{ID}_i, k_{\mathsf{ID}_i}, E_i)$;
    3. run $r_i \leftarrow \text{OT-SKE.decrypt}(K_i, C_i)$ and set $c_i = \tilde{c}_i$;
    4. get $b \in \{0, 1\}$ from $\mathcal{O}_{\text{verify}}(pk, \eta_i, r_i, c_i)$, and output b.
  - $\tilde{\mathcal{O}}_{\text{extract}}(\tilde{pk}, \cdot)$: On a query of (η_i, ID_i), get F_{η_i} from $\mathcal{O}_{\text{extract}}(pk, \eta_i)$, and output F_{η_i}.
- $\mathcal{A}_2$: Given (δ, η^*, c^*) as input, $\mathcal{A}_2$ sets $\tilde{c}^* = c^*$ and calls $\tilde{\mathcal{A}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$ to get $\tilde{r}^* = (C^*, E^*)$. Then $\mathcal{A}_2$ runs $d_{\mathsf{ID}^*} \leftarrow \text{IB-KEM.extract}(mpk, msk, \mathsf{ID}^*)$, sets $k_{\mathsf{ID}^*} = d_{\mathsf{ID}^*}$, and gets $K^* \leftarrow \text{IB-KEM.decap}(mpk, \mathsf{ID}^*, k_{\mathsf{ID}^*}, E^*)$ then $r^* \leftarrow \text{OT-SKE.decrypt}(K^*, C^*)$, and outputs r^* in the experiment $\mathbf{Exp}^{\text{Challenge}}_{\mathcal{A},\text{PPoR}}(\delta, \eta^*)$.

According to the correctness of PoR-FDV scheme and PPoR scheme, if $\tilde{r}^*$ is a valid response to $\tilde{c}^*$, then $\tilde{b} = 1$; if r^* is a valid response to c^*, then $b = 1$. In the above simulation, the validity of $\tilde{r}^*$ implies the validity of r^* because r^* is the correct decryption of $\tilde{r}^*$ on the same challenge $c^* = \tilde{c}^*$. As $\tilde{\mathcal{A}}$ is a PPT λ-adversary against the PoR-FDV scheme, we have:

$$\Pr[\mathbf{Exp}^{\text{Challenge}}_{\mathcal{A},\text{PPoR}}(\delta, \eta^*) = 1] = \Pr[\mathbf{Exp}^{\text{Challenge}}_{\tilde{\mathcal{A}},\text{PoR-FDV}}(\tilde{\delta}, \eta^*, \mathsf{ID}^*) = 1] \geq \lambda,$$

where $\delta = (\tilde{\delta}, \mathsf{ID}^*, \tilde{pk}, msk)$. In other words, the constructed $\mathcal{A}$ is a PPT λ-adversary against the PPoR scheme.

Assuming the PPoR scheme achieves (ϵ, λ)-valid proof of retrievability, by Definition 1 we have that F_{η^*} is retrievable from $\mathcal{O}_{\text{extract}}(pk, \eta^*)$ with probability greater than ϵ by oracle access to $\mathcal{A}_2(\delta, \cdot)$. As shown above, $\tilde{\mathcal{O}}_{\text{extract}}(\tilde{pk}, \eta^*, \mathsf{ID}^*)$ and $\mathcal{O}_{\text{extract}}(pk, \eta^*)$ output the same F_{η^*}, and $\mathcal{A}_2(\delta, \eta^*, c^*)$ can be simulated by $\tilde{\mathcal{A}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$, hence equivalently, F_{η^*} is retrievable from $\tilde{\mathcal{O}}_{\text{extract}}(\tilde{pk}, \eta^*, \mathsf{ID}^*)$ with probability greater than ϵ by oracle access to $\tilde{\mathcal{A}}_2(\tilde{\delta}, \cdot)$. So by Definition 4, the generic PoR-FDV scheme achieves (ϵ, λ)-valid adaptive-ID proof of retrievability. □

Theorem 2. *If the underlying IB-KEM scheme achieves ϵ-IND-ID-CCA security and the underlying OT-SKE scheme achieves ϵ-IND-PA security, then the proposed PoR-FDV scheme achieves ϵ-SND-ID-DV security.*

Proof. Suppose there exists a PPT adversary $\tilde{\mathcal{B}} = (\tilde{\mathcal{B}}_1, \tilde{\mathcal{B}}_2)$ that breaks the ϵ-SND-ID-DV security of the PoR-FDV scheme, we can utilize it to construct either an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the ϵ-IND-ID-CCA security of the IB-KEM scheme, or an adversary $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ that breaks the ϵ-IND-PA security of the OT-SKE scheme. Below we analyze these two cases in details.

Case 1: assume the OT-SKE scheme is secure, then insecurity of the PoR-FDV scheme must imply insecurity of the IB-KEM scheme. The construction for $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is as follows.

- $\mathcal{B}_1$: Given mpk as input, $\mathcal{B}_1$ runs $(pk, sk) \leftarrow \text{PPoR.keyGen}(1^k)$ and sets $\tilde{pk} = (pk, mpk)$. Then $\mathcal{B}_1$ calls $\tilde{\mathcal{B}}_1^{\tilde{\mathcal{O}}}(\tilde{pk})$ to get $(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$, and sets the adversarial state $\delta = (\tilde{\delta}, \eta^*, \tilde{c}^*, \tilde{pk}, sk)$. Finally, $\mathcal{B}_1$ outputs (δ, ID^*) in the experiment $\mathbf{Exp}^{\text{Setup}}_{\mathcal{B},\text{IB-KEM}}(1^k)$.
  The simulation for the set of oracles $\tilde{\mathcal{O}} = \{\tilde{\mathcal{O}}_{\text{encode}}, \tilde{\mathcal{O}}_{\text{deriveVerKey}}, \tilde{\mathcal{O}}_{\text{challenge}}, \tilde{\mathcal{O}}_{\text{verify}}, \tilde{\mathcal{O}}_{\text{extract}}\}$ for the PoR-FDV scheme is given as below, using the set of oracles $\mathcal{O} = \{\mathcal{O}_{\text{extract}}, \mathcal{O}_{\text{decap}}\}$ for the IB-KEM scheme:
  - $\tilde{\mathcal{O}}_{\text{encode}}(\tilde{sk}, \cdot)$: On a query of a file F_i, run $(\eta_i, F_{\eta_i}) \leftarrow \text{PPoR.encode}(sk, F)$, and output (η_i, F_{η_i}).
  - $\tilde{\mathcal{O}}_{\text{deriveVerKey}}(\tilde{pk}, \tilde{sk}, \cdot)$: On a query of a verifier identity ID_i, get d_{ID_i} from $\mathcal{O}_{\text{extract}}(mpk, msk, \mathsf{ID}_i)$, set $k_{\mathsf{ID}_i} = d_{\mathsf{ID}_i}$ and output k_{ID_i}. Notice that $\mathsf{ID}_i \neq \mathsf{ID}^*$ as $\mathsf{ID}^* \notin \tilde{\mathcal{Q}}_{\text{deriveVerKey}}$, and it is consistent with the restriction $\mathsf{ID}^* \notin \mathcal{Q}_{\text{extract}}$.
  - $\tilde{\mathcal{O}}_{\text{challenge}}(\tilde{pk}, \cdot)$: On a query of (η_i, ID_i), run $c_i \leftarrow \text{PPoR.challenge}(pk, \eta_i)$, set $\tilde{c}_i = c_i$ and output $\tilde{c}_i$.
  - $\tilde{\mathcal{O}}_{\text{verify}}(\tilde{pk}, \cdot)$: On a query of $(\eta_i, \mathsf{ID}_i, \tilde{r}_i, \tilde{c}_i)$ where $\tilde{r}_i = (C_i, E_i)$,
    1. get K_i from $\mathcal{O}_{\text{decap}}(mpk, \mathsf{ID}_i, E_i)$;

2. run $r_i \leftarrow$ OT-SKE.decrypt(K_i, C_i), and set $c_i = \tilde{c}_i$;
3. run $b \leftarrow$ PPoR.verify(pk, η_i, r_i, c_i), and output $b \in \{0,1\}$.

We stress that the probability that $(\mathsf{ID}_i, E_i) = (\mathsf{ID}^*, E^*)$ is negligible, since that the value of E^*, which is to be generated by the probabilistic algorithm IB-KEM.encap (i.e. $\mathcal{O}_{\mathsf{encap}}(mpk, ID^*)$) in the Challenge experiment, remains unpredictable at this stage.

– $\tilde{\mathcal{O}}_{\mathsf{extract}}(\tilde{pk}, \cdot)$: On a query of (η_i, ID_i), run $F_{\eta_i} \leftarrow$ PPoR.extract(pk, η_i), and output F_{η_i}.

- $\mathcal{B}_2$: Given $(\delta, \mathsf{ID}^*, K^\dagger, E^*)$ as input, $\mathcal{B}_2$ sets $c^* = \tilde{c}^*$ and runs $r^* \leftarrow$ PPoR.respond(pk, η^*, c^*), $C^* \leftarrow$ OT-SKE.encrypt($K^\dagger, r^*$), and sets $\tilde{r}^* = (C^*, E^*)$. Finally, $\mathcal{B}_2$ calls $\tilde{\mathcal{B}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{r}^*, \tilde{c}^*)$ to get $\tilde{b}' \in \{0,1\}$. Then $\mathcal{B}_2$ sets $b' = \tilde{b}'$ and outputs b' in the experiment $\mathbf{Exp}^{\mathsf{Challenge}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(\delta, \mathsf{ID}^*)$.

Since that $\tilde{b} = 0$ indicates $\tilde{r}^* \leftarrow_{\$} \tilde{\mathcal{R}}$, it is equivalent to the case of $b = 0$ that $\tilde{r}^* = (C^*, E^*)$ where the ciphertext $C^* \leftarrow$ OT-SKE.encrypt($K^\dagger, r^*$) is generated using a randomly picked key $K^\dagger \leftarrow_{\$} \mathcal{K}$. On the other hand, $\tilde{b} = 1$ indicates that $\tilde{r}^*$ is a valid response to $\tilde{c}^*$, and equivalent to the case of $b = 1$ that $(K^*, E^*) \leftarrow \mathcal{O}_{\mathsf{encap}}(mpk, \mathsf{ID}^*)$ and $C^* \leftarrow$ OT-SKE.encrypt(K^*, r^*) is a valid encryption of $r^* \leftarrow$ PPoR.respond(pk, η^*, c^*) where $c^* = \tilde{c}^*$. In other words, the simulation above ensures that $b = \tilde{b}$. Furthermore, as $\mathcal{B}_2$ outputs b' as $\tilde{b}'$ which is the output of $\tilde{\mathcal{B}}_2$, and assuming $\tilde{\mathcal{B}}$ breaks the ϵ-sound designated verification of the PoR-FDV scheme, then we have:

$$\begin{aligned}
&\Pr[b' = b | b' \leftarrow \mathbf{Exp}^{\mathsf{Challenge}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(\delta, \mathsf{ID}^*)] \\
&= \Pr[b' = \tilde{b} | b' \leftarrow \mathbf{Exp}^{\mathsf{Challenge}}_{\mathcal{B},\mathsf{IB\text{-}KEM}}(\delta, \mathsf{ID}^*)] \\
&= \Pr[\tilde{b}' = \tilde{b} | \tilde{b}' \leftarrow \mathbf{Exp}^{\mathsf{Challenge}}_{\tilde{\mathcal{B}},\mathsf{PoR\text{-}FDV}}(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)] \\
&> \frac{1}{2} + \epsilon.
\end{aligned}$$

Therefore, we constructed an adversary $\mathcal{B}$ that breaks the ϵ-IND-ID-CCA security of the IB-KEM scheme, resulting in a contradiction.

Case 2: assume the IB-KEM scheme is secure, then insecurity of the PoR-FDV scheme must imply insecurity of the OT-SKE scheme. The construction for $C = (C_1, C_2)$ is as follows.

- C_1: Given 1^k as input, C_1 runs $(pk, sk) \leftarrow$ PPoR.keyGen(1^k), (mpk, msk) $\leftarrow$ IB-KEM.setup(1^k), and sets $\tilde{pk} = (pk, mpk)$, $\tilde{sk} = (sk, msk)$. Then C_1 calls $\tilde{\mathcal{B}}_1^{\tilde{\mathcal{O}}}(\tilde{pk})$ to get $(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)$, sets $c^* = \tilde{c}^*$ and runs $r^* \leftarrow$ PPoR.respond(pk, η^*, c^*). Finally, C_1 sets $M_0 \leftarrow_{\$} \mathcal{R}$, $M_1 = r^*$, $\delta = (\tilde{\delta}, \eta^*$, ID^*, $\tilde{c}^*$, $\tilde{pk}$, $\tilde{sk})$, and outputs (δ, M_0, M_1) in the experiment $\mathbf{Exp}^{\mathsf{Setup}}_{C,\mathsf{OT\text{-}SKE}}(1^k)$. The simulation of the set of oracles $\tilde{\mathcal{O}} = \{\tilde{\mathcal{O}}_{\mathsf{encode}}, \tilde{\mathcal{O}}_{\mathsf{deriveVerKey}}, \tilde{\mathcal{O}}_{\mathsf{challenge}}, \tilde{\mathcal{O}}_{\mathsf{verify}}, \tilde{\mathcal{O}}_{\mathsf{extract}}\}$ for answering queries from $\tilde{\mathcal{B}}_1$ is given as below.
  – $\tilde{\mathcal{O}}_{\mathsf{encode}}(\tilde{sk}, \cdot)$, $\tilde{\mathcal{O}}_{\mathsf{challenge}}(\tilde{pk}, \cdot)$, $\tilde{\mathcal{O}}_{\mathsf{extract}}(\tilde{pk}, \cdot)$: Quires to the three oracles are answered in the same way as in Case 1 above.
  – $\tilde{\mathcal{O}}_{\mathsf{deriveVerKey}}(\tilde{pk}, \tilde{sk}, \cdot)$: On a query of a verifier identity ID_i, run $d_{\mathsf{ID}_i} \leftarrow$ IB-KEM.extract(mpk, msk, ID_i), set $k_{\mathsf{ID}_i} = d_{\mathsf{ID}_i}$ and output k_{ID_i}. Notice that C_1 has the knowledge of msk.
  – $\tilde{\mathcal{O}}_{\mathsf{verify}}(\tilde{pk}, \cdot)$: On a query of $(\eta_i, \mathsf{ID}_i, \tilde{r}_i, \tilde{c}_i)$ where $\tilde{r}_i = (C_i, E_i)$,
    1. run $K_i \leftarrow$ IB-KEM.decap(mpk, ID_i, k_{ID_i}, E_i) where $k_{\mathsf{ID}_i} \leftarrow$ IB-KEM.extract(mpk, msk, ID_i);
    2. run $r_i \leftarrow$ OT-SKE.decrypt(K_i, C_i), and set $c_i = \tilde{c}_i$;
    3. run $b \leftarrow$ PPoR.verify(pk, η_i, r_i, c_i), and output $b \in \{0,1\}$.
- C_2: As mentioned in Section 2.3, we can assume that the challenge ciphertext C^* is the encrypted M_b using a key K^* generated by $(K^*, E^*) \leftarrow$ IB-KEM.encap(mpk, ID^*). So the input of C_2 is (δ, C^*, E^*, M_0, M_1). Then C_2 sets $\tilde{r}^* = (C^*, E^*)$, calls $\tilde{\mathcal{B}}_2(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{r}^*, \tilde{c}^*)$ to get $\tilde{b}' \in \{0,1\}$, and outputs $b' = \tilde{b}'$ in the experiment $\mathbf{Exp}^{\mathsf{Challenge}}_{C,\mathsf{OT\text{-}SKE}}(\delta, M_0, M_1)$.

As $M_0 \leftarrow_{\$} \mathcal{R}$ and $M_1 = r^*$, $\tilde{r}^*$ must be either a random value in $\tilde{\mathcal{R}}$ when $b = 0$, or a valid encryption of r^* when $b = 1$. This implies that $b = \tilde{b}$. And as C_2 outputs $b' = \tilde{b}'$, similarly to that of Case 1, we have:

$$\begin{aligned}
&\Pr[b' = b | b' \leftarrow \mathbf{Exp}^{\mathsf{Challenge}}_{C,\mathsf{OT\text{-}SKE}}(\delta, M_0, M_1)] \\
&= \Pr[\tilde{b}' = \tilde{b} | \tilde{b}' \leftarrow \mathbf{Exp}^{\mathsf{Challenge}}_{\tilde{\mathcal{B}},\mathsf{PoR\text{-}FDV}}(\tilde{\delta}, \eta^*, \mathsf{ID}^*, \tilde{c}^*)] \\
&> \frac{1}{2} + \epsilon.
\end{aligned}$$

Therefore, we constructed an adversary C that breaks the ϵ-IND-ID-PA security of the OT-SKE scheme, resulting in a contradiction.

Combining the results of Case 1 and 2, the theorem follows. □

By a similar security analysis as Theorem 2, we can obtain the following theorem.

Theorem 3. *If the underlying IB-KEM scheme achieves ϵ-IND-sID-CCA security and the underlying OT-SKE scheme achieves ϵ-IND-PA security, then the proposed PoR-FDV scheme achieves ϵ-SND-sID-DV security.*

5. A concrete PoR-FDV scheme

Applying the generic construction, now we give the first PoR-FDV scheme derived from Shacham and Waters' PPoR scheme (Section 3.3 in Shacham and Waters (2008)), Chen *et al.*'s IB-KEM scheme (Section 3.3 in Chen et al. (2006)), and a standard DEM scheme using the folklore construction of OT-SKE shown in Section 2.3.

5.1. Scheme specification

Below is the specification of our PoR-FDV scheme. Notice that we implicitly use Zhang *et al.*'s short signature scheme for generating the file handlers and the verification keys.

keyGen(1^k). Choose an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order q. Let g be a generator of $\mathbb{G}$, g_T be a generator of $\mathbb{G}_T$, $H_1 : \{0,1\}^* \to \mathbb{Z}_q$, $H_2 : \{0,1\}^* \to \mathbb{G}$, $H_3 : \{0,1\}^* \to \mathbb{Z}_q$, $H_4 : \{0,1\}^* \to \mathbb{Z}_q$, $H_5 : \{0,1\}^* \to \{0,1\}^k$ be cryptographic hash functions, $\mathcal{G}$ be a pseudo-random bit generator. Pick $\alpha, \beta, \gamma \leftarrow_{\$} \mathbb{Z}_q$ and set $w = g^\alpha$, $v = g^\beta$, $z = g^\gamma$, $P = e(g, g)$. Let $\tilde{pk} = (q, G, G_T, e, g, H_1, H_2, H_3, H_4, H_5, \mathcal{G}, w, v, z, P)$, $\tilde{sk} = (\alpha, \beta, \gamma)$.

encode($\tilde{sk}, F$). First, use an erasure code (e.g. Reed-Solomon code) to encode the file F into F'. Then split F' into n blocks $m_1, \cdots, m_n$, and each block m_i into s sectors $m_{i,1}, \cdots, m_{i,s} \in \mathbb{Z}_q$.[1] Choose a random file identifier $\xi \leftarrow_{\$} \mathbb{Z}_q$ and an s-element vector $\mathbf{u} = (u_1, \cdots, u_s)$ where $u_j \leftarrow_{\$} \mathbb{G}$. Set $\tau = \xi \| n \| \mathbf{u}$, and compute:

$$S_\tau = g^{\frac{1}{H_1(\tau)+\alpha}}.$$

Then for $1 \le i \le n$, compute the metadata of each block as:

$$\sigma_i = (H_2(\xi \| i) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}})^\beta.$$

Finally, output $\eta = \tau \| S_\tau$ and $F_\eta = \{m_{i,j}\} \| \{\sigma_i\}$.

deriveVerKey($\tilde{pk}, \tilde{sk}, \mathsf{ID}$). Compute the verification key of a verifier with identity ID as:

$$k_{\mathsf{ID}} = g^{\frac{1}{H_3(\mathsf{ID})+\gamma}}.$$

challenge($\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}}$). First, check the validity of file handler $\eta = \tau \| S_\tau$, namely the equation $P = e(wg^{H_1(\tau)}, S_\tau)$ should hold. Then choose two l-element vectors $\mathsf{t} = (t_1, \cdots, t_l)$ and $\mathsf{v} = (v_1, \cdots, v_l)$, where $t_i \leftarrow_{\$} [1, n]$, $v_i \leftarrow_{\$} \{0,1\}^k$ for $i = 1, \cdots, l$. Here l is the parameter that affects the efficiency of data retrieval. Output the challenge as $\tilde{c} = \mathsf{t} \| \mathsf{v}$.

respond($\tilde{pk}, \eta, \mathsf{ID}, \tilde{c}$). Firstly, compute the proof as:

$$\mu_j = \sum_{i=1}^{l} v_i m_{t_i,j} \text{ for } 1 \le j \le s, \text{ and } \sigma = \prod_{i=1}^{l} \sigma_{t_i}^{v_i},$$

and let $r = \{\mu_j\} \| \sigma$. Then choose $K \leftarrow_{\$} \{0,1\}^k$, and set $x = H_4(K)$, $Y = zg^{H_3(\mathsf{ID})}$, $E_1 = Y^x$, $E_2 = K \oplus H_5(P^x)$, and $E = E_1 \| E_2$. Finally, compute $C = \mathcal{G}(K) \oplus r$, and output the response $\tilde{r} = (C, E)$.

verify($\tilde{pk}, \eta, \tilde{r}, \tilde{c}, k_{\mathsf{ID}}$). First, decapsulate the encryption key as:

$$K = H_5(e(k_{\mathsf{ID}}, E_1)) \oplus E_2.$$

Then compute $x = H_4(K)$, then $Y = zg^{H_3(\mathsf{ID})}$, $E_1' = Y^x$, and output $\perp$ if $E_1 \neq E_1'$. Otherwise, compute $r = \mathcal{G}(K) \oplus C = \{\mu_j\} \| \sigma$, and check if

$$e(\sigma, g) = e(\prod_{i=1}^{l} H_2(\xi \| t_i)^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, v).$$

[1] For instance, to offer 80-bit security where $k = 80$, an encoded 4 GB file F' can be splitted into $n = 10^6$ blocks of size 4 KB, each block into $s = 200$ sectors of 160 bits.

Table 2
Performance comparison of related works.

Schemes	||Chal||	||Resp||	$\mathbf{Comp_P}$	$\mathbf{Comp_V}$
Shacham and Waters (2008)	$l\|\mathbb{G}\| + l\log\frac{\|F\|}{s\|\mathbb{G}\|}$	$(s+1)\|\mathbb{G}\|$	lExp	$(s+l)$Exp + 2Pair
Yang et al. (2021)	$5\|\mathbb{G}\| + 280$	$10\|\mathbb{G}\|$	$5s$Exp	6Exp + 7Pair
PoR-FDV	$l\|\mathbb{G}\| + l\log\frac{\|F\|}{s\|\mathbb{G}\|}$	$(s+2)\|\mathbb{G}\| + k$	lExp	$(s+l+1)$Exp + 3Pair

a **||Chal||** and **||Resp||** denote the challenge size and the response size respectively, for evaluation of communicational costs.
b $\mathbf{Comp_P}$ and $\mathbf{Comp_V}$ denote the computational costs on the prover (cloud server) side and on the verifier side, respectively.
c l is the sampling size, $|\mathbb{G}|$ is the group element size, $|F|$ is the file size, s is the number of sectors in each block, k is the system security parameter. Exp denotes the time required for performing 1 exponentiation operation, Pair denotes the time required for performing 1 bilinear pairing operation.

Finally, output $b = 1$ if the checking holds, otherwise output $b = 0$. Notice that the verifier with identity ID can precompute and store the value Y before the auditing.

extract$(\tilde{pk}, \eta, \mathsf{ID}, k_{\mathsf{ID}})$. The file F can be recovered by running multiple auditing queries. In particular, after obtaining a valid response $\tilde{r}$ to a challenge $\tilde{c} = \mathsf{t}\|\mathsf{v}$, compute $r = \{\mu_j\}\|\sigma$ as in the above algorithm verify. Denote by $\bar{\mathsf{v}}$ the n-element vector that the t_i-th entry is v_i for $i = 1, \cdots, l$, while the other entries are all zeros. Denote by M the $n \times s$ matrix that the (i, j)-th entry is $m_{i,j}$ for $i = 1, \cdots, n$, $j = 1, \cdots, s$. Let $\mu = (\mu_1, \cdots, \mu_s)$. Then we have $\bar{\mathsf{v}} \cdot \mathsf{M} = \mu$. Therefore, by making n auditing queries to the server, we can obtain a $n \times n$ matrix A of rows $\bar{\mathsf{v}}_1$, $\cdots$, $\bar{\mathsf{v}}_n$ and a $n \times s$ matrix B of rows μ_1, $\cdots$, μ_n, such that $\mathsf{A} \cdot \mathsf{M} = \mathsf{B}$. The encoded file $F' = \{m_{i,j}\}$ represented by the matrix M is retrievable if A has an inverse matrix A^{-1}, which happens with non-negligible probability. Shacham and Waters (2008) has proved that if the server outputs valid proofs with probability λ, and F was encoded using a rate-ρ erasure code, then the success probability of recovering ρ fraction of F' is at least $\lambda - \frac{(\rho n)^l}{(n-l+1)^l}$. By correctness of the erasure code, the original file F can be extracted from the recovered fraction of F'.

5.2. Performance analysis

Shacham and Waters' PPoR scheme Shacham and Waters (2008) is a typical PoR based on BLS signature that many other PoR schemes are its variants with similar performance. For fair comparison with related works Hanser and Slamanig (2013); Wang (2013); Yang et al. (2021), we include Shacham and Waters (2008) as well as Yang et al. (2021) in Table 2, for that Hanser and Slamanig (2013); Wang (2013) are PDP schemes rather than PoR schemes. When compared to Shacham and Waters (2008), our PoR-FDV scheme incurs no increase in the challenge size, and raises the response size by only $|\mathbb{G}| + k$ bits, where the response in the original PPoR scheme is $(s+1)|\mathbb{G}|$-bit long. Without lose of generality, we suppose $k = 80$, $|\mathbb{G}| = 171$, and $s = 200$, then the communication cost increases by only 0.73%. Regarding the computational efficiency of PoR-FDV, the cost on the prover side remains the same as that of Shacham and Waters (2008), while the additional cost on the verifier side is about 1 pairing plus 1 exponentiation for computing $e(k_{\mathsf{ID}}, E_1)$ and Y^x respectively, assuming the pre-computation of Y is adopted. The verification algorithm in Shacham and Waters (2008) requires for 2 pairings plus $s + l$ exponentiations, so the auditing efficiency is also comparable to the original scheme. Regarding Yang et al. (2021), it has lower communicational costs then ours, but much higher computational cost (e.g. $5s\text{Exp} \gg l\text{Exp}$ for the general case $l = 300$) as the trade-off.

6. Conclusion

We proposed the first generic construction of PoR that achieves completely flexible delegation of auditing capability under the control of data owner, in the sense that only some designated verifiers are allowed to audit and retrieve users' data. A new framework and security model for such kind of PoR, which we denote by PoR-FDV, are also well formalized. Our generic construction of PoR-FDV is provably secure and sound, leading to the design of a concrete PoR-FDV scheme. The performance analysis shows that our PoR-FDV scheme still preserves comparable efficiency as many other existing PoR schemes, while offering better controllability in terms of verifier designation.

CRediT authorship contribution statement

Xiao Tan: Conceptualization, Formal analysis, Methodology, Writing – original draft. **Qi Xie:** Conceptualization, Supervision, Writing – review & editing. **Lidong Han:** Investigation, Visualization. **Shengbao Wang:** Investigation, Validation. **Wenhao Liu:** Data curation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xiao TAN reports financial support was provided by National Natural Science Foundation of China.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work is partly supported by National Natural Science Foundation of China (grant no. 61702152). The authors appreciate the anonymous reviewers for their valuable comments.

References

Anthoine, G., Dumas, J., de Jonghe, M., Maignan, A., Pernet, C., Hanling, M., Roche, D., 2021. Dynamic proofs of retrievability with low server storage. In: Proc. 30th USENIX Security Symposium'21, pp. 537–554.
Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D., 2007. Provable data possession at untrusted stores. In: Proc. 14th CCS'07, Alexandria, USA, pp. 598–609.
Ateniese, G., Pietro, R., Mancini, L., Tsudik, G., 2008. Scalable and efficient provable data possession. In: Proc. 4th SecureComm'08, Istanbul, Turkey, pp. 1–10.
Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N., 2008. Generic constructions of identity-based and certificateless KEM. J. Cryptol. 21, 178–199.
Binanda, S., Sushmita, R., 2017. Efficient proofs of retrievability with public verifiability for dynamic cloud storage. IEEE Trans. Cloud Comput. 8, 138–151.
Chen, Y., Chang, J., 2022. Identity-based proof of retrievability meets with identity-based network coding. Clust. Comput., 1–17.
Chen, L., Cheng, Z., Malone-Lee, J., Smart, N., 2006. Efficient ID-KEM based on the Sakai-Kasahara key construction. IEE Proc. Inf. Secur. 153, 19–26.
Chen, R., Li, Y., Yu, Y., Li, H., Chen, X., Susilo, W., 2020. Blockchain-based dynamic provable data possession for smart cities. IEEE Int. Things J. 7, 4143–4154.
Cramer, R., Shoup, V., 2003. Design and analysis of practical public key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. 33, 167–226.
Guo, W., Qin, S., Gao, F., Zhang, H., Li, W., Jin, Z., Wen, Q., 2020a. Dynamic proof of data possession and replication with tree sharing and batch verification in the cloud. IEEE Trans. Serv. Comput. https://doi.org/10.1109/TSC.2020.3022812.
Guo, W., Qin, S., Lu, J., Gao, F., Jin, Z., Wen, Q., 2020b. Improved proofs of retrievability and replication for data availability in cloud storage. Comput. J. 63, 1216–1230.
Hanser, C., Slamanig, D., 2013. Efficient simultaneous privately and publicly verifiable robust provable data possession from elliptic curves. In: Proc. 10th SECRYPT'13, Reykjavik, Iceland, pp. 1–12.
Ji, S., Zhou, W., Ma, C., Li, D., Zhu, K., Fang, L., 2023. Proofs of retrievability with tag outsourcing based on Goppa codes. Comput. Stand. Interfaces 86, 103719.
Juels, A., Kaliski Jr., B., 2007. Pors: proofs of retrievability for large files. In: Proc. 14th CCS'07, Alexandria, USA, pp. 598–609.

Li, X., Liu, S., Lu, R., Khan, M., Gu, K., Zhang, X., 2022. An efficient privacy-preserving public auditing protocol for cloud-based medical storage system. IEEE J. Biomed. Health 26, 2020–2031.
Lu, C., Chen, Z., Meng, X., Chen, L., Wu, G., Shen, J., 2022. Identity-based public auditing with tag compression. In: Proc. 8th International Conference on Computer and Communications'22, pp. 1411–1415.
Mishra, B., Jena, D., Somula, R., Sankar, S., 2020. Secure key storage and access delegation through cloud storage. Int. J. Knowl. Syst. Sci., 45–64.
Ren, Z., Wang, L., Wang, Q., Xu, M., 2015. Dynamic proofs of retrievability for coded cloud storage systems. IEEE Trans. Serv. Comput. 11, 685–698.
Shacham, H., Waters, B., 2008. Compact proofs of retrievability. In: Proc. 14th ASIACRYPT'08, Melbourne, Australia, pp. 90–107.
Susilo, W., Li, Y., Guo, F., Lai, J., Wu, G., 2022. Public cloud data auditing revisited: removing the tradeoff between proof size and storage cost. In: Proc. 27th European Symposium on Research in Computer Security'22, pp. 65–85.
Wang, H., 2013. Proxy provable data possession in public clouds. IEEE Trans. Serv. Comput. 6, 551–559.
Wang, Q., Wang, C., Ren, K., Lou, W., Li, J., 2011. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. 22, 847–859.
Wang, H., He, D., Fu, A., Li, Q., Wang, Q., 2019. Provable data possession with outsourced data transfer. IEEE Trans. Serv. Comput. 14, 1929–1939.
Wang, C., Wang, Q., Ren, K., Lou, W., 2010. Privacy-preserving public auditing for data storage security in cloud computing. In: Proc. 29th INFOCOM'10, San Diego, USA, pp. 525–533.
Wang, Q., Wang, C., Li, J., Ren, K., Lou, W., 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proc. 14th ESORICS'09, Saint-Malo, France, pp. 355–370.
Xu, J., Chang, E., 2012. Towards efficient proofs of retrievability. In: Proc. 7th ASIACCS'12, Seoul, Korea, pp. 79–80.
Yang, K., Jia, X., 2013. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distrib. Syst. 24, 1717–1726.
Yang, A., Xu, J., Weng, J., Zhou, J., Wong, D., 2021. Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage. IEEE Trans. Cloud Comput. 9, 212–225.
Yuan, J., Yu, S., 2013. Proofs of retrievability with public verifiability and constant communication cost in cloud. In: Proc. Cloud Computing'13, Hangzhou, China, pp. 79–80.
Zhou, L., Fu, A., Mu, Y., Wang, H., Yu, S., Sun, Y., 2021. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. Inf. Sci. 545, 254–276.

Xiao Tan received his B.S. and M.S. degrees from Fudan University in 2007 and 2010 respectively, and his Ph.D. degree from the City University of Hong Kong in 2014. He was employed as a senior research associate in the Department of Computer Science at City University of Hong Kong from February 2014 to July 2014. Presently, he is working as a lecturer at Hangzhou Key Laboratory of Cryptography and Network Security, Hangzhou Normal University. His main research interests include cryptography and information security, in particular, far exchange protocols, cloud storage systems, signature and encryption schemes.

Qi Xie received his Ph.D. in Applied Mathematics from Zhejiang University in 2005. He has been a postdoctoral researcher in the College of Computer Science at Zhejiang University from 2007 to 2010 and an academic visitor at the school of Computer Science at University of Birmingham in UK between September 2009 and September 2010. Currently, he is a professor at the School of Information Science and Engineering in Hangzhou Normal University. His research interests are digital signatures, authentication and key exchange protocols, and so on.

Lidong Han received his Ph.D. degree from School of Mathematics, Shandong University, China, in 2010. Currently, he is working at Hangzhou Normal University. His current research interests include Cryptography, cloud computing and remote user authentication.

Shengbao Wang received his Ph.D. degree in computer science from Shanghai Jiao Tong University in 2008 and is now working as an associate professor at the Department of Computer Science and Engineering, Hangzhou Normal University, China. His research interests lie in public key cryptography, especially focus on public key encryption and key agreement protocols.

Wenhao Liu received his Ph.D. degree from University of Electronic Science and Technology of China in 2010. Now, he work at Key Laboratory of Cryptography and Network Security of Hangzhou Normal University. His research interests include signature and data security in cloud.