Blockchain-based fair payment smart contract for public cloud storage auditing

摘要

云存储在当今的云生态系统中扮演着重要的角色。越来越多的客户倾向于将数据外包给云端。尽管它具有丰富的优势，完整性一直是一个重要的问题。审计方法通常用于确保云场景中的完整性。然而，传统的审计方案需要一个第三方审计者（TPA），而这在现实世界中并不总是可行的。此外，先前的方案意味着有限的现收现付服务，因为它要求客户提前支付服务费用。

在本文中，我们旨在通过采用区块链替代TPA，并设计一个基于区块链的公共云存储审计公平支付智能合约来解决上述缺点。在我们的系统中，数据所有者和云服务提供商（CSP）将运行一个基于区块链的智能合约。该合约确保CSP需要定期提交**数据占有证明**。只有验证通过，CSP才会得到报酬；否则，它不会得到报酬，但必须支付罚款。为了减少合同执行过程中的交互数量，我们提出了**非交互式公共可证明数据占有**的概念，并在此基础上设计了一个基于区块链的公共云存储审计智能合约。

---

1 Our Contribution

利用区块链的去中心化和自动触发的优势，我们设计了一个基于区块链的公共云存储审计公平支付智能合约。在我们的系统中，数据所有者和CSP将运行一个基于区块链的智能合约。该合约确保CSP需要定期提交数据占有证明。只有当验证通过后，CSP才会被支付，否则CSP不仅不会获得任何报酬，还要支付罚款。

当使用传统的公共审计协议时，验证者需要与CSP进行交互。在这个过程中，验证者通常生成一个随机的挑战，CSP基于这个挑战返回一个数据占有证明。这种交互式证明不适合在智能合约平台上执行，因为每个共识节点（作为验证者）都需要与CSP交互，系统通信的复杂度和CSP的计算成本将是不可接受的。为了避免智能合约平台与CSP在合约执行过程中的交互，我们提出了非交互式公共可证明数据占有（NI-PPDP）的概念，并在此基础上设计了一个基于区块链的云存储公平支付智能合约。具体地说，我们通过扩展Wang等人的交互式公共审计方案[33]，构造了一个有效的NI-PPDP方案。具体来说，本文的贡献主要包括以下三个方面：

- 提出了非交互式公共可证明数据占有（NI-PPDP）的概念；
- 构造了一个有效的NI-PPDP格式，并在随机喻言模型中给出了形式化证明；
- 基于NI-PPDP设计一种基于区块链的云存储公平支付智能合约

---

2 Preliminaries

2.1 Bilinear parings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups with prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a function from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. $\mathbb{G}$ and $\mathbb{G}_T$ are called bilinear groups, if

- (Bilinear) $\forall g_1, g_2 \in \mathbb{G}, x, y \in Z_p$, $e(g_1{}^x, g_2{}^y) = e(g_1, g_2)^{xy}$.
- (Non-degenerate) $\exists h_1, h_2 \in \mathbb{G}_1$, $e(h_1, h_2)$ is a generator of $\mathbb{G}_T$.

Furthermore, all group operations and function e should be computable.

2.2 Computational Diffiffiffie-Hellman (CDH) assumption

Definition 1. Suppose $\mathbb{G}$ is a q-order cyclic group, g is a random generator of $\mathbb{G}$. For $\forall x, y \in \{0, \ldots, q-1\}$, given (g, g^x, g^y), it is computationally intractable to compute g^{xy}.

2.3 Review Wang et al's interactive public auditing scheme

- Setup Phase:
  KeyGen $(1^\lambda) \to pk, sk$: Let g be a generator of bilinear group $\mathbb{G}$. The data owner chooses two hash functions, $H(\cdot)$: $\{0,1\}^* \to \mathbb{G}$, $h(\cdot) : \mathbb{G}_T \to \mathbb{Z}_p$, and generates public/private key pairs (spk, ssk) for a digital signature algorithm. Then, it chooses $x \leftarrow \mathbb{Z}_p$, $u \leftarrow \mathbb{G}$ randomly, and calculates $v \leftarrow g^x$. The secret key is $sk = (ssk, x)$ and the public key is $pk = (spk, g, u, v, e(u, v), H(\cdot), h(\cdot))$.
  TagGen $(F, pk, sk) \to \Phi$: We suppose that the data file can be expressed as $F = \{m_i\}_{1 \le i \le n}$, where $m_i \in \mathbb{Z}_p$. The data owner computes authenticators $\sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x \in \mathbb{G}$ for $i \in [1, n]$, where $W_i = name||i$, and $name \leftarrow \mathbb{Z}_p$ is chosen by the data owner randomly as the identifier of file F. Let $\Psi = \{\sigma_i\}_{1 \le i \le n}$.
  To ensure the correctness of the file identifier $name$, it runs a signing algorithm Sig on the $name$ under ssk, and sets $t = name \,||\, Sig_{ssk}(name)$ as the file identifier for F. The data owner then uploads data file F and corresponding data tags $\Phi = (\Psi, t)$ to CSP.

- Audit Phase:
  The verifier first retrieves the file identifier t, and verifies the signature $Sig_{ssk}(name)$ via spk, and aborts by outputing $\perp$ if verification fails. Otherwise, the verifier recovers the $name$. Then, the verifier picks a random c-element subset $I = \{s_1, \ldots, s_c\}$ of set $[1, n]$. For each index $i \in I$, the verifier chooses $v_j \leftarrow Z_p^*$ randomly. The verifier sends $chal = \{(i, v_i)\}_{i \in I}$ to the server.
  ProofGen $(pk, \Phi, F, chal) \to \Sigma$: It computes

$$\sigma = \prod_{j \in I} \sigma_j^{v_j},$$

  and

$$\mu' = \sum_{j \in I} v_j \cdot m_j.$$

  The CSP chooses $s \leftarrow \mathbb{Z}_p$ randomly, and calculates $T = e(u, v)^s \in \mathbb{G}_T$. Then, it computes: $\mu = s + \gamma \mu' \bmod p$, where $\gamma = h(T) \in Z_p$. It sends $\Sigma = \{\mu, \sigma, T\}$ as the data possession proof to the verifier.
  Verify (pk, Σ): The verifier computes $\gamma = h(T)$ and outputs 1 or 0, according to the correctness of equation below

$$T \cdot e(\sigma^\gamma, g) \stackrel{?}{=} e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^\gamma \cdot u^\mu, v).$$

3. Non-Interactive public provable data possession scheme

3.1 System Model

在非交互式公共可证明数据占有（NI-PPDP）方案中，有三种类型的实体，即数据所有者、CSP和验证者（如图3所示）。与传统的交互式公共可证明数据占有方案不同，CSP和验证者在审计过程中不需要交互。一个NI-PPDP方案由4种算法组成，这些算法分为两个阶段：

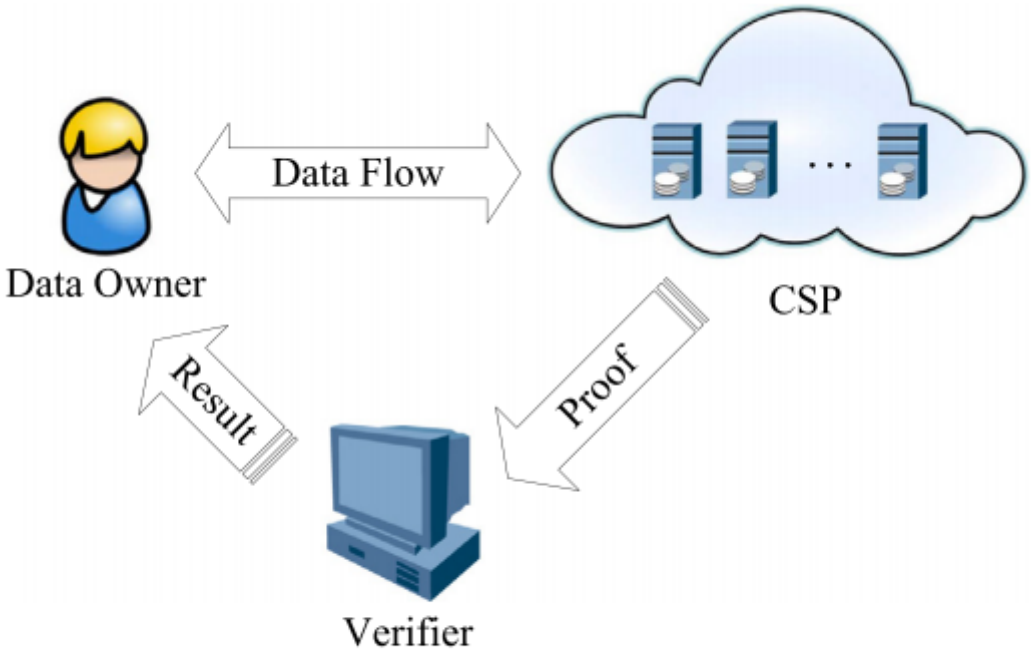

Fig. 3. System model.

- Setup Phase: In this phase, data owners generate the data tags Φ corresponding to their data file F and store F along with Φ on the cloud storage service. They will run the key generation algorithm and tag generation algorithm as follows.
  KeyGen $(1^{\lambda}) \rightarrow pk, sk$: The key generation algorithm is run by data owner. It takes security parameter λ as input, and outputs public key pk, secret key sk.
  TagGen $(F, pk, sk) \rightarrow \Phi$: The tag generation algorithm is run by the data owner. It takes data file F, public key pk, secret key sk as input, and outputs the corresponding data tags Φ. The data owner then uploads F and Φ to the CSP.

- Audit Phase: In this phase, CSP will prove that it stores complete data. It gives proof based on data tags Φ, data file F and current state τ, by running a proof generation algorithm. Everybody could verify the proof publicly, by running a verifying algorithm. Usually, the verifier is a third-party auditor (TPA), who has a higher computing capability than the data owner, and can check data integrity for data users.
  ProofGen $(pk, \Phi, F, \tau) \rightarrow \Sigma$: The proof generation algorithm is run by CSP. It takes public key pk, data tags Φ, data file F and current state τ as input, and outputs a proof Σ. We suppose the current state τ is some time-varying public information, which cannot be controlled by CSP.
  Verify $(pk, \Sigma) \rightarrow 0, 1$: This is a publicly verifiable algorithm, that can be executed by anyone in this system. It takes public key pk, proof Σ as input and outputs 1 or 0 based on the correctness of Σ.

3.2 Threat model

- 我们假设CSP没有向外部各方透露其托管数据的动机，也没有放弃其托管数据的动机。但是，由于一些无法控制的因素，如软件漏洞、硬件故障、网络路径中的漏洞、有经济动机的黑客、恶意或意外的管理错误，用户数据的完整性可能会被破坏。此外，为了其自身的利益，CSP甚至可能决定向数据所有者隐藏此数据损坏事件。
- 验证者可以根据CSP提供的证明，为数据所有者验证数据的完整性。但是，如果验证者能够从证明中了解到外包数据的相关信息，则可能会损害数据所有者。
- 我们假设CSP不会与任何验证者串通。

3.3 Design goals

The NI-PPDP scheme should achieve:

- Correctness: For all keypairs $(pk, sk) \leftarrow \mathbf{KeyGen}(1^{\lambda})$, for all data files F, and for all states τ, the verification algorithm always outputs

$$1 \leftarrow \mathbf{Verify}(pk, \mathbf{ProofGen}(\mathbf{TagGen}(F, sk), F, \tau)).$$

- Soundness(Data integrity): to ensure that the verification can be passed only if the integrity of data is achieved.
- Privacy preserving: to ensure that auditing process does not disclose any information data.
- Non-interactive: to ensure that the CSP and verifier do not need to interact during auditing.

- Public auditability: to ensure that anyone can verify the integrity of the remote data only depending on the data possession proof given by the cloud storage provider and the public key of data owners.

3.4. Formal security defifinition

在上述设计目标中，数据完整性和隐私保护是NI-PPDP关键的安全特性。因此，我们给出的正式定义如下。

3.4.1 Data integrity (soundness)

We use the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ to define the soundness of data integrity:

1. $\mathcal{C}$ calls key generation algorithm $\mathbf{KeyGen}(1^{\lambda})$ to generate keypair (pk, sk), and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ can interact with $\mathcal{C}$ repeatedly and make queries for some file F. Then, $\mathcal{C}$ returns $\Phi \leftarrow \mathbf{TagGen}(F, pk, sk)$ to $\mathcal{A}$.
3. Finally, $\mathcal{A}$ outputs Σ for some data file F and data tag Φ on state τ.

Define the advantage of $\mathcal{A}$ is $Adv_{\mathcal{A}} = Pr[\mathbf{Verify}(pk, \Sigma) = 1]$. We say the adversary wins the above game, if $Adv_{\mathcal{A}}$ is non-negligible.

Definition 2. A non-interactive public provable data possession scheme is sound if exists an efficient extraction algorithm **Extr** such that, for every adversary $\mathcal{A}$, who outputs Σ for some data file F and data tag Φ on state τ and wins above game, the extraction algorithm recovers file F from Φ and Σ, i.e., $\mathbf{Extr}(pk, \Phi, \Sigma) = F$.

3.4.2. Privacy preserving

We use the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ to define privacy preserving:

1. $\mathcal{C}$ calls key generation algorithm $\mathbf{KeyGen}(1^{\lambda})$ to generate keypair (pk, sk), and gives pk to $\mathcal{A}$.
2. A can interact with $\mathcal{C}$ repeatedly and make queries for some file F. Then, $\mathcal{C}$ returns $\Phi \leftarrow \mathbf{TagGen}(F, pk, sk)$ to $\mathcal{A}$.
3. At some point, $\mathcal{A}$ submits two files F_0^* and F_1^* to $\mathcal{C}$. Then, C selects $b \in \{0, 1\}$ randomly, and returns $\mathbf{ProofGen}(\mathbf{TagGen}(F_b^*, sk), F_b^*, \tau)$ to A.
4. Finally, $\mathcal{A}$ outputs a guess bit b'.

Defining the advantage of $\mathcal{A}$ as $Adv_{\mathcal{A}} = Pr[b' = b] - 1/2$. We say the adversary wins the above game, if $Adv_{\mathcal{A}}$ is non-negligible.

Definition 3. A non-interactive public provable data possession scheme is privacy preserved if for any probability polynomial time adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in above game is negligible.

4. Blockchain-based fair payment smart contract for cloud storage

在传统的云存储系统中，数据所有者在使用云存储之前必须支付租金。一旦数据被云存储服务提供商丢失或损坏，数据所有者就很难恢复经济损失。为了解决这一问题，我们引入了一种新的云存储支付模式，即数据所有者在享受服务后，根据服务质量支付费用。为了保护数据所有者和云存储服务提供商的权利，我们在本系统中使用了基于区块链的智能合约平台和非交互式公共可证明数据占有方案。

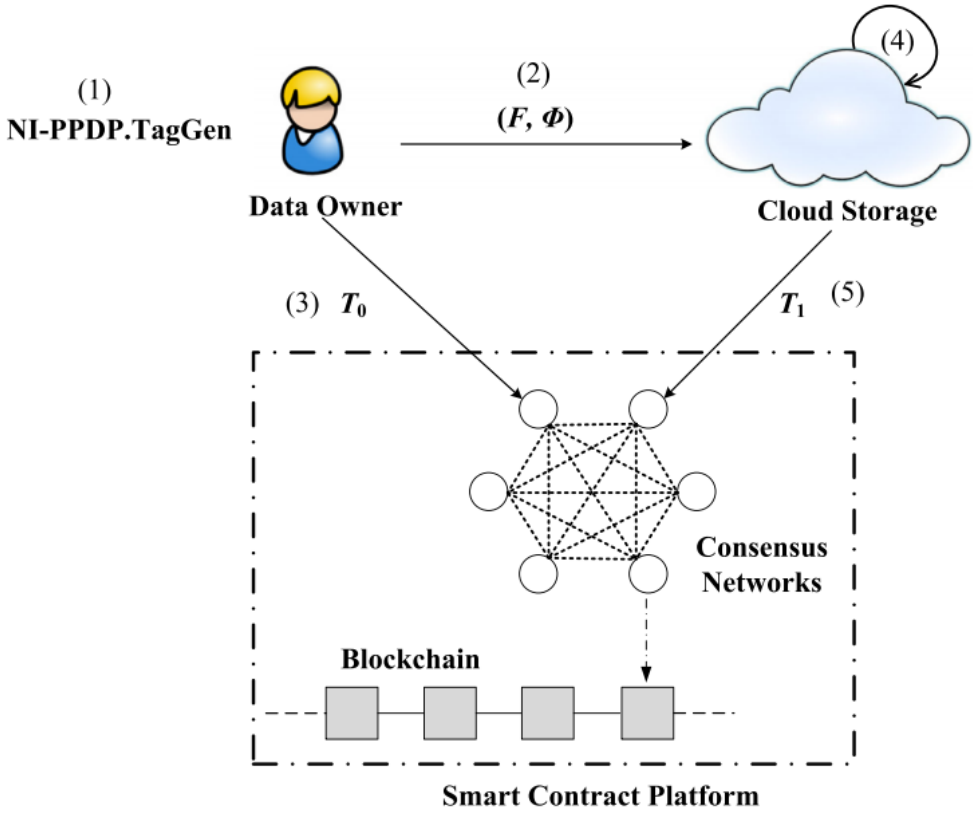

Fig. 4. Data storage process.

如图4所示，数据所有者运行NI-PPDP方案的密钥生成算法和标签生成算法，将文件F和数据标签Φ上传到CSP。同时将合约T_0（图5）提交给智能合同平台。T_0包括文件名、文件大小、文件哈希、上传时间、存储期限、服务费、数据所有者帐户（加密货币）、云存储帐户（加密货币）、数据所有者的公钥、数据所有者的签名和智能合约代码。本合约确保如果云存储服务提供商能够按时提交正确的数据占有证明（使用NI-PPDP方案），数据所有者将按时支付服务费。

Contract T_0
File Name: FN File Size: FS File Hash: FH Upload time: UT Storage Period: CP Service Charges: SC Data owner Account: DOA Cloud Storage Account: CSA Data owner's public key: pk_D Data owner's signature: sig_D

Contract Content:
promise
{
if (NI-PPDP.Verify(pk_D, Σ)==1)
pay SC from DOA to CSA;
}

Fig. 5. Contract T_0.

在收到文件F和数据标签Φ后，云存储提供商将检查数据的完整性和来源的身份验证。如果所有验证检查都通过，云存储服务器将合约T_1（图6）提交给智能合约平台。T_1确认文件F已被CSP接收，并确保如果数据占有证明没有通过，CPS将向数据所有者支付补偿。即T_1有两个功能：(1)确认收到F，(2)作出赔偿承诺。请注意，补偿不是必要的，但补偿反映了云存储提供商的声誉。

Contract T_1
File Name: FN File Size: FS File Hash: FH Receiving Time: RT Storage Period: CP Penalty: Pen Data owner Account: DOA Cloud Storage Account: CSA CSP's public key: pk_C CSP's signature: sig_C

Contract Content:
confirm F; /*T_0 takes effect*/
promise
{
if (NI-PPDP.Verify(pk_D, Σ)==0)
pay Pen from CSA to DOA;
}

Fig. 6. Contract T_1.

具体的工作流可描述为：

1. 数据所有者采用NI-PPDP方案，在文件F上运行其TagGen算法，获得相应的数据标签Φ。
2. 数据所有者上传文件F和数据标签Φ到CSP。
3. 数据所有者向智能合约平台提交合约T_0（图5）。
4. CSP检查数据的完整性和来源的身份认证。
5. CSP将合约T_1（图6）提交给智能合约平台。

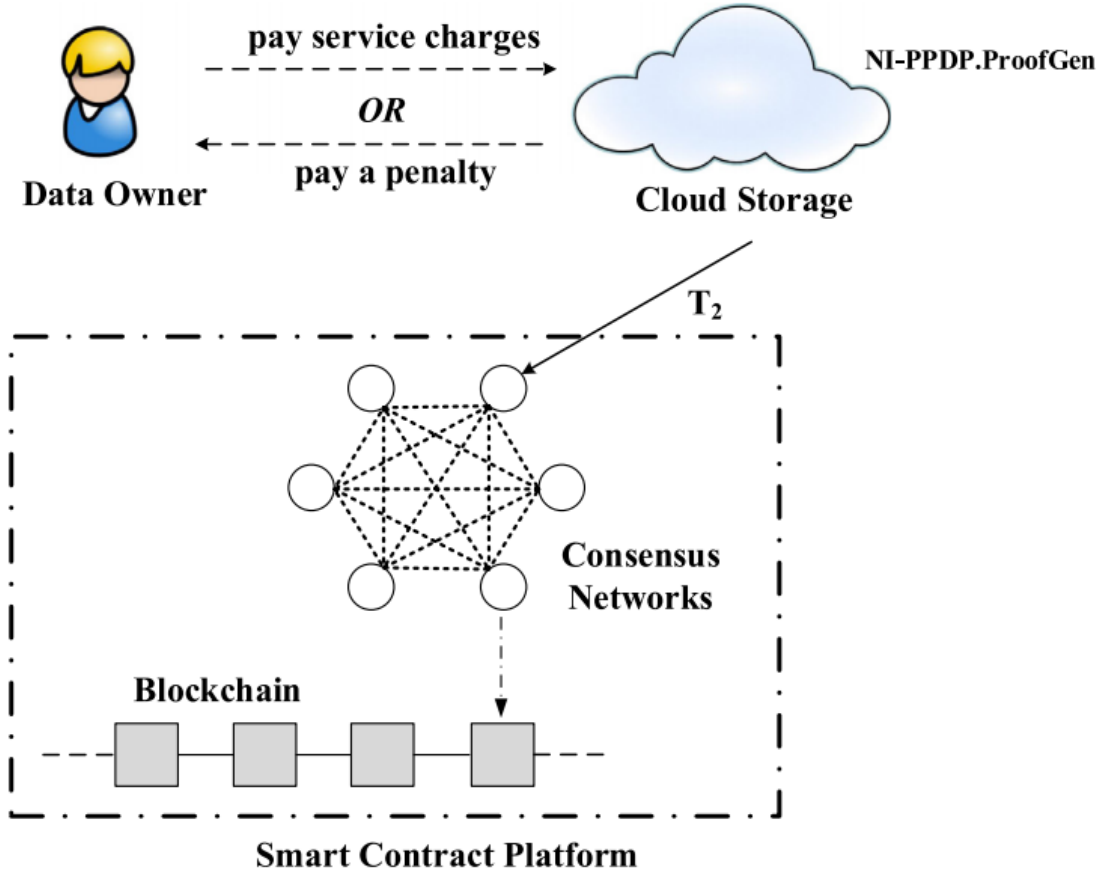

Fig. 7. Data validation process.

如图7所示，为了获取服务费，云服务器定期提交一个合约T_2（图8），其中包含一个非交互式数据占有证明Σ。共识网络将在T_2中验证该数据占有证明，并根据验证结果激活T_0或T_1。如果验证成功，T_0将被激活，数据所有者将向云服务器支付服务费用，否则T_1将被激活，CPS将向数据所有者支付补偿。

Contract T_2
Payment contract: T_0 Compensation contract: T_1 Data possession proof: Σ
Contract Content: activate T_0 or T_1 depended on the validation result of Σ;

Fig. 8. Contract T_2.

5. A specific NI-PPDP scheme

下面，我们提出了一个非交互式公共可证明数据占有方案的具体构造。我们的建设是通过扩展由Shacham和Waters以及Wang等人引入的交互式公共审计方案来实现的。

5.1. Our construction

我们的NI-PPDP方案是基于Wang等人的公共审计方案构建的。主要区别在于在审核阶段验证者和CSP之间没有交互。为了模拟挑战过程，我们对当前状态的输入使用了伪随机函数。在我们的方案中也有两个阶段。

- Setup Phase:
  KeyGen $(1^{\lambda}) \rightarrow pk, sk$: Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, select g as a generator of group $\mathbb{G}$, select two cryptographic hash functions $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{G}$, $h(\cdot) : \mathbb{G}_T \rightarrow \mathbb{Z}_p$, and select pseudorandom function $\mathcal{F}(\cdot)$: $\{0,1\}^* \rightarrow [1, n]$, which maps arbitrary values uniformly to an integer range $[1, n]$.
  The data owner generates public/private key pairs (spk, ssk) for a digital signature algorithm. Then, it chooses $x \leftarrow \mathbb{Z}_p$, $u \leftarrow \mathbb{G}$ randomly, and calculates $v \leftarrow g^x$. The secret key is $sk = (ssk, x)$ and the public key is $pk = (spk, g, u, v, e, H(\cdot), h(\cdot), \mathcal{F}(\cdot))$.
  TagGen $(F, pk, sk) \rightarrow \Phi$: Suppose that the data file can be expressed as $F = \{m_i\}_{1 \le i \le n}$, where $m_i \in \mathbb{Z}_p$. The data owner computes authenticators $\sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x$ for $i \in [1, n]$, where $W_i = name||i$, and $name \leftarrow \mathbb{Z}_p$ is chosen by the data owner randomly as the identifier of file F. Let $\Psi = \{\sigma_i\}_{1 \le i \le n}$.
  To ensure the correctness of the file identifier $name$, it runs signature algorithm Sig on $name$ under ssk, and sets $t = name \,||\, Sig_{ssk}(name)$ as the file identifier for F. The data owner then uploads the data file F and corresponding data tags $\Phi = (\Psi, t)$ to CSP.

- Audit Phase:
  In this phase, the verifier does not need to choose the challenge set. The CSP uses the current state as the input of pseudorandom function $\mathcal{F}(\cdot)$, to simulate the challenge process.
  ProofGen $(pk, \Phi, F, \tau) \rightarrow \Sigma$: In input, τ represents the current public status information, which contains current time and some other public information and cannot be controlled by CSP. We assume that τ will change in each running of **ProofGen** algorithm. In our blockchain-based fair payment model, τ should also include the header information of the current block of blockchain, which can not be controlled by CSP.
  First of all, CSP choose an appropriate number $c < n$. For $i \in [1, c]$, CSP computes

$$s_i \leftarrow \mathcal{F}(\tau||i).$$

  $I = \{s_1, s_2, \ldots, s_c\}$ is a c-element multiset, $\forall s_i \in [1, n]$. Note, the multiset I is allowed to contain repeated elements.
  For $j \in I$, the CSP computes

$$v_j \leftarrow h(\tau||j).$$

  Then, it computes

$$\sigma = \prod_{j \in I} \sigma_j^{v_j},$$

  and

$$\mu' = \sum_{j \in I} v_j \cdot m_j.$$

  The CSP chooses $s \leftarrow \mathbb{Z}_p$ randomly, and calculates $T = e(u, v)^s \in \mathbb{G}_T$. Then, it computes: $\mu = s + \gamma\mu' \bmod p$, where $\gamma = h(T) \in Z_p$. It sends $\Sigma = \{\mu, \sigma, T, \tau, c\}$ as the data possession proof to the verifier.
  Verify (pk, Σ): The verifier runs the verification algorithm of $Ver(spk, id, Sig_{ssk}(name))$ to verify integrity of id and verifies the authenticity of state information τ via blockchain. It aborts, if any verification fails. Otherwise, the verifier recovers the $name$. Then, the verifier computes $I = \{\mathcal{F}(\tau||1), \mathcal{F}(\tau||2), \ldots, \mathcal{F}(\tau||c)\}$, $\{v_j = h(\tau||j)\}_{j \in I}$, $\{h(N_j)\}_{j \in I}$, $\gamma = h(T)$ and then outputs 1 or 0, according to the correctness of equation below

$$T \cdot e(\sigma^{\gamma}, g) \stackrel{?}{=} e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v).$$

5.2. Correctness

$$
\begin{aligned}
T \cdot e(\sigma^{\gamma}, g) &= e(u, v)^{s} \cdot e((\prod_{i=s_1}^{s_c} (H(W_i) \cdot u^{m_i})^{x \cdot v_i})^{\gamma}, g) \\
&= e(u^{s}, v) \cdot e((\prod_{i=s_1}^{s_c} (H(W_i)^{v_i} \cdot u^{m_i v_i}))^{\gamma}, g)^{x} \\
&= e(u^{s}, v) \cdot e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu' \gamma}, v) \\
&= e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu' \gamma + s}, v) \\
&= e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v)
\end{aligned}
$$

6. Design goals analysis

6.1. Data integrity (soundness)

We use the hybrid argument technique to prove soundness as in [31]. First of all, we define the following games:

Game-0. Game-0 is the original game defined in Section 3.4.1.

Game-1. Game-1 is the same as Game-0, except that the challenger $\mathcal{C}$ records all the tags it signed in a local list. If adversary $\mathcal{A}$ ever submits a tag Φ, that (1) has a valid signature under *ssk* but (2) is not signed by $\mathcal{C}$, then $\mathcal{C}$ announces failure and aborts.

Game-2. Game-2 is the same as Game-1, except that $\mathcal{C}$ records all the responses to **TagGen** queries from $\mathcal{A}$. If $\mathcal{A}$ is successful (i.e., **Verify** output 1) but $\mathcal{A}$'s aggregate signature σ is not equal to $\prod_{j \in I} \sigma_j^{v_j}$, then the challenger $\mathcal{C}$ announces failure and aborts.

Game-3. Game-3 is the same as Game-2, except that challenger $\mathcal{C}$ announces failure and aborts, if at least one of the aggregate messages μ' is not equal to $= \sum_{j \in I} v_j \cdot m_j$.

Lemma 1. *If there is an algorithm $\mathcal{A}$ can distinguish between **Game-0** and **Game-1** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has a non-negligible advantage to break the existentially unforgeability.*

Analysis. If $\mathcal{A}$ causes $\mathcal{C}$ to abort in Game-1, then we can use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ against the existentially unforgeability of the signature scheme.

Lemma 2. *If there is an algorithm $\mathcal{A}$ can distinguish between **Game-1** and **Game-2** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has non-negligible advantage to break the computation Diffie-Hellman assumption.*

Analysis. Suppose g^x and g^y are the elements of CDH problem, we set $v = g^x$, $u = g^y$. Suppose $\mathcal{A}$ can respond a signature σ', which is different from the expected signature σ. We can calculate

$$e(\sigma'/\sigma, g) = e(\prod_{j \in I} u^{\Delta \mu_j}, v) = e(g^{\sum_{j \in I} \Delta \mu_j \cdot x \cdot y}, g)$$

Therefore, we can calculate $g^{x \cdot y} = (\sigma'/\sigma)^{\frac{1}{\sum_{j \in I} \Delta \mu_j}}$

Lemma 3. *If there is an algorithm $\mathcal{A}$ that can distinguish between **Game-2** and **Game-3** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has a non-negligible advantage to break the computation Diffie-Hellman assumption.*

Analysis. We only introduce the main ideas. We suppose that $h(\cdot)$ is a random oracle controlled by an extractor, who answers the hash query asked by the adversary (CSP). For $\gamma = h(T)$ from extractor, the adversary outputs $\{\mu, \sigma, T, t, \tau, c\}$ makes:

$$T \cdot e(\sigma^{\gamma}, g) = e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v).$$

Then, the extractor rewinds $h(T)$ to be $\gamma^* \neq \gamma$. The adversary outputs $\{\mu^*, \sigma, T, t, \tau, c\}$ makes:

$$T \cdot e(\sigma^{\gamma^*}, g) = e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma^*} \cdot u^{\mu^*}, v).$$

Divide above two equations, we have

$$\begin{aligned}
e(\sigma^{\gamma-\gamma^*}, g) &= e(u^{\sum_{j\in I}(h(N_j)v_j)(\gamma-\gamma^*)} \cdot u^{\mu-\mu^*}, v) \\
e(\sigma^{\gamma-\gamma^*}, g) &= e(u^{\sum_{j\in I}(h(N_j)v_j)(\gamma-\gamma^*)} \cdot u^{\mu-\mu^*}, g^x) \\
\sigma^{\gamma-\gamma^*} &= u^{\sum_{j\in I}(h(N_j)v_j)x(\gamma-\gamma^*)} \cdot u^{x(\mu-\mu^*)} \\
(\Pi_{j\in I}\sigma_j^{v_j})^{\gamma-\gamma^*} &= (\Pi_{j\in I}u^{h(N_j)v_jx(\gamma-\gamma^*)} \cdot u^{x(\mu-\mu^*)} \\
u^{x(\mu-\mu^*)} &= (\Pi_{j\in I}(\sigma_j/u^{h(N_j)x})^{v_j})^{\gamma-\gamma^*} \\
u^{x(\mu-\mu^*)} &= (\Pi_{j\in I}(u^{xm_j})^{v_j})^{\gamma-\gamma^*} \\
\mu-\mu^* &= (\sum_{j\in I} m_jv_j) \cdot (\gamma-\gamma^*) \\
\sum_{j\in I} m_jv_j &= (\gamma-\gamma^*)/(\mu-\mu^*)
\end{aligned}$$

Finally, $\{\sigma, \mu' = (\mu-\mu^*)/(\gamma-\gamma^*)\}$ can be treated as a response for the extractor.

Theorem 1. *If the signature scheme is existentially unforgeable and computational Diffie-Hellman assumption holds in bilinear groups, then no probabilistic polynomial time adversary can break the soundness of our NI-PPDP scheme with non-negligible probability.*

Proof. Any adversary's advantage in **Game 3** must be 0 since the challenger always announces failure and aborts if there is no integral file F, i.e. at least one of the aggregate messages μ' is not equal to $= \sum_{j\in I} v_j \cdot m_j$. By the games sequence and Lemmas 1–3, an adversary's advantage in the original game **Game 0** must be negligibly close to 0. □

6.2. Privacy preserving

This theorem shows that the verification process do not reveal any information of the users' data.

Theorem 2. *Our NI-PPDP scheme is privacy preserved.*

Proof. This proof follows from [31,33]. We only introduce the main ideas, i.e. the data possession proof $\Sigma = \{\mu, \sigma, T, t, \tau, c\}$ can not reveal any information about μ'. In the random oracle model, the simulator can construct the response without knowing μ'. It randomly chooses γ, μ from Z_p, and sets

$$T \leftarrow e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v)/e(\sigma^{\gamma}, g).$$

Then, the simulator sets random oracle $h(\cdot)$, makes $\gamma = h(T)$. □

6.3. Non-Interactive

Compared with the traditional interactive public provable data possession scheme, there is no challenge stage in our scheme. CSP does not have to interact with the verifier when it makes proof. To achieve this goal, we use the pseudorandom function, $\mathcal{F}(\cdot)$, to generate the challenge set. Since the input of the $\mathcal{F}(\cdot)$ contains the current state information, this information is related to the current time and the current block chain state, so it cannot be controlled by the CPS. Due to the pseudo randomness, the challenge set generated in this way is indistinguishable to the random choice of verifier.

6.4. Public auditability

It is clear that our scheme has achieved public auditability. The validation algorithm does not depend on any secret inputs.