symmetry

MDPI

Article

An Efficient and Universal Real-Time Data Integrity Verification Scheme Based on Symmetric Key in Stream Computing System

Hongyuan Wang, Wanting Zhu *, Baokai Zu and Yafang Li

Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; wanghongyuan@bjut.edu.cn (H.W.); bzu@bjut.edu.cn (B.Z.); yafangli@bjut.edu.cn (Y.L.)
* Correspondence: zhuwanting@bjut.edu.cn; Tel.: +86-138-1115-2938

Abstract: The integrity of real-time data streams has not been solved for a long time and has gradually become a difficult problem in the field of data security. Most of the current data integrity verification schemes are constructed using cryptographic algorithms with complex computation, which cannot be directly applied to real-time stream computing systems. Aiming at the above issue, this paper adopts the Carter–Wegman MAC method, pseudo-random function and symmetric cryptography mechanism to construct the Real-Time Data Integrity Verification scheme based on symmetric key in stream computing systems (RT-DIV), which converts a one-time MAC to a multiple-time MAC and retains the advantage of security performance. Then, a security analysis is given under the standard model. Finally, experiments and data analysis are conducted in a simulated environment, and the experimental results show that the RT-DIV scheme can effectively guarantee the integrity of real-time data streams. Furthermore, the RT-DIV scheme lays the foundation for the secure application of the stream computing system.

Keywords: data integrity verification; stream computing system; symmetric key

Citation: Wang, H.; Zhu, W.; Zu, B.; Li, Y. An Efficient and Universal Real-Time Data Integrity Verification Scheme Based on Symmetric Key in Stream Computing System. *Symmetry* **2023**, *15*, 1541. https://doi.org/10.3390/sym15081541

Academic Editor: José Carlos R. Alcantud

Received: 16 June 2023
Revised: 2 August 2023
Accepted: 3 August 2023
Published: 4 August 2023

1. Introduction

Data integrity is the basic premise in guaranteeing the accuracy of big data analysis and calculation and is an important research interest in the field of data security. There are two main computing modes of big data: batch offline computing and stream real-time computing. Batch computing is mainly oriented towards static, persistent data. Data are stored previously and then distributed with the computing logic to distributed computing nodes for calculation. Stream computing is mainly oriented towards data streams. Instead of storing all data, it directly performs data calculation in memory for the data stream within a certain period of time.

According to the survey, the current data integrity verification technology and research in the batch computing mode are relatively mature, while there is no perfect scheme to solve the data integrity problem in the stream computing mode. In the traditional batch computing mode, the data integrity verification mechanism is divided into two categories according to whether fault-tolerant recovery measures are performed on the data [1]: the Provable Data Possession (PDP) and the Proofs of Retrievability (POR). However, the current PDP and POR schemes are only suitable for the integrity verification of stored data and cannot be directly applied in stream computing systems. Due to the real-time, burst, disorder, and volatility characteristics of big data streams [2], the message and information in stream computing systems are easy to lose or repeat and inconsistent in state. The problem of incomplete data becomes more and more prominent, meaning the research of data integrity and is faced with unprecedented difficulties.

Currently, although most stream computing systems have a message acknowledgment mechanism (acker) [3] to check whether each message can be completely processed, the integrity of the message data itself cannot be guaranteed, and if the complex verification

calculation is run on the acker, it is easy to affect the efficiency of real-time message calculation. At the same time, because the stream computing process is not persistent, it is impossible to view the processing path of historical messages, which makes the problem of incomplete message data difficult to reproduce.

Nowadays, research on data integrity verification in stream computing systems is relatively rare. Most of the existing data integrity verification schemes adopt asymmetric cryptographic algorithms such as elliptic curve, RSA signature or bilinear mapping to construct data integrity verification schemes, which are computationally inefficient and usually used for static data in storage mode. They cannot meet the efficiency requirements of real-time data in stream computing systems, so an efficient and universal solution cannot be formed.

To address the above issues, this paper constructed a Real-Time Data Integrity Verification algorithm scheme (RT-DIV) based on symmetric key in the stream computing system. By using the Carter–Wegman MAC construction method, pseudo-random function and symmetric cryptography mechanism, the RT-DIV scheme can efficiently record, analyze and verify the data of each message in different processing periods in real time, actively discover the incompleteness of message and automatically alarm and replay erroneous data to ensure the global integrity and consistency of message data throughout the life cycle.

The paper is organized as follows: We give the introduction in Section 1, and the related work in Section 2. Then, the proposed model of our scheme is given in Section 3, and the preliminaries in Section 4. The detailed implementation of scheme design and system design is given in Section 5. We further give the security analysis and experiments in Sections 6 and 7, respectively. Finally, the conclusion is given in Section 8.

2. Related Works

Most of the existing data integrity schemes are constructed based on RSA algorithms, homomorphic verification tags, bilinear mapping, the third party, elliptic curve and other emerging technologies.

1. Schemes based on RSA

Ateniese et al. [4] proposed the first PDP scheme, which is one of the most classic data integrity verification schemes; it is based on the RSA scheme and randomly selects a fixed number of data blocks to aggregate into a smaller value to guarantee each verification, thus greatly reducing the communication and computational cost. Subsequently, Ateniese et al. [5] proposed an efficient and securely verifiable PDP scheme, which can support dynamic data manipulation, such as addition, deletion and modification, but the number of verifications is limited. Erway et al. [6] proposed a Dynamic PDP (DPDP) scheme, which is constructed by S-PDP with a jump table structure to add, modify and delete data. Wang et al. [7] proposed a non-repudiable PDP scheme, which constructed a reusable commitment function based on the Pederson commitment. It can guarantee the security of both the user and the server.

2. Schemes based on homomorphic verification tags (HVT)

Shacham et al. [8] proposed a CPOR scheme using HVT which greatly reduces the communication cost, then presented separate schemes for public and private verification which overcomes the shortcoming of limited verification of the POR scheme [9]. Curtmola et al. [10] proposed a scheme to guarantee the integrity of data stored on multiple cloud servers and combine the error correction codes. Wang et al. [11] use homomorphic tokens to verify the data integrity of multiple servers, which can reduce the communication cost, but leads to low security because it is easy to identify which server the data are stored on. Wang et al. [12,13] proposed an external data integrity tracking and verification system for stream computing systems, which meets the requirement of stream data integrity verification to a certain extent, but, because the verification module is outside the stream system, the real-time performance cannot be fully guaranteed.

3. Schemes based on bilinear mapping (BM)

Zhu et al. [14–16] presented a Cooperative PDP (CPDP) scheme based on hash index hierarchy and bilinear pairing to support data migration and scalability of service in hybrid cloud. But the operation of bilinear pairing is very time-consuming and leads to low computational efficiency.

4. Schemes based on the third party (TP)

Wang et al. [17,18] used Trusted Third Party (TTP) to protect the privacy of data and user identity; they use TTP instead of client to verify data integrity, and TTP cannot obtain the data content during verification. But the drawback is that TTP can cause additional overhead to the user. Armknecht et al. [19] proposed an outsourced POR (OPOR) model, which is named Fortress, using an Untrusted TP (UTP) instead of users to interact with server and simultaneously protect user, server and TP.

5. Schemes based on elliptic curve (EC)

Hanser et al. [20] proposed the first simultaneous private and public PDP scheme based on EC, which supports both private and public verification by using the same pre-processing process to generate verification tags. The drawback is still the exponential calculation. Wang et al. [21] presented an efficient PDP based on EC, which replaces homomorphic multiplication with homomorphic addition to improve efficiency.

6. Schemes based on the group

Tate et al. [22] proposed the first group DIV scheme based on trusted hardware for multi-user data sharing and data updating in cloud storage. Wang et al. [23] proposed an identity privacy preserving group scheme to support private and public authentication, where the group member identity will not be leaked during verification. Wang et al. [24] proposed a public scheme to support revocation of group members; it guarantees that revoked members cannot forge tags or proofs by using known information. Zhu et al. [25] proposed a group provable of storage (GPOS) scheme to efficiently distinguish malicious members and prevent selected member attacks and collusion attacks. Wang et al. [26] proposed a group PDP scheme supporting data deduplication and resisting selected member attacks. It ensures that group members use the same tags to complete the verification.

7. Schemes based on the emerging technologies

With the development of emerging technologies, data integrity verification has received new development. Li et al. [27] proposed a data integrity audit scheme, which is based on biometric identification technology to achieve the target of identity fuzzy; they solved the complex key management problem but caused a large computation and communication cost. Xu et al. [28] proposed an arbitrable and distributed remote data auditing scheme based on blockchain, which adopts the smart contract to notarize the outsourced data integrity and achieves non-repudiation verification. Guo et al. [29] proposed outsourced dynamic PDP scheme based on Merkle hash tree, which achieves multiple updates at one time without repeated computation and transmission. Subsequently, they proposed a dynamic PDP and replication (DPDPR) scheme [30], which stores the indexed Merkle hash to reduce the local storage cost; however, the computational cost is large when the data are uploaded, and the verification cost is expensive when data are updated. Li et al. [31] proposed a cloud-side collaborative stream DIV scheme based on chameleon authentication tree; it can guarantee integrity verification and data confidentiality with stream data insertion and query by a trusted third party. Zhou et al. [32] proposed a lattice-based PDP scheme for cloud-based smart grid data management systems. Wang et al. [33] analyzed and improved a lattice-based public data integrity verification scheme, which can guarantee proxy-oriented secure data outsourcing and storage correctness. Qi et al. [34] proposed a blockchain-based light-weighted PDP scheme which enables "hash-sign-switch" tag computing, which is suitable for low performance devices. However, these emerging technologies do not meet real-time requirements.

In summary, the current data integrity verification schemes cannot fully meet the requirements of real-time and lightweight computing of stream computing systems, which seriously restricts the wide application of stream systems in various fields. Therefore, it is necessary to build an efficient and universal real-time data integrity verification scheme in stream computing systems.

3. Proposed Model

The design objective and proposed model design are presented in this section.

3.1. Design Objective

The real-time data integrity verification subsystem inside the general stream computing system needs to meet the objectives of efficiency, accuracy and real-time data recoverability.

Efficiency: The most important characteristic of a stream computing system is efficiency, which needs to efficiently process the data stream in cache. Therefore, a data integrity verification scheme suitable for stream computing systems must be efficient enough to meet the computational performance of the original system.

Accuracy: A key consideration for stream computing systems is accuracy. Only systems with high accuracy can have a wide range of applications. Therefore, the stream computing system with data integrity guarantee also needs to meet the requirement of accuracy.

Real-time data recoverability: Stream computing systems are prone to data loss or data errors, which can lead to errors in calculation results or system deviations. Therefore, it is necessary to promptly recover the erroneous data to guarantee the stability of the system. Meanwhile, the verification subsystem is inside the stream system, making it possible to trace and recover erroneous data as soon as possible.

3.2. Model Design

The proposed model design is shown in Figure 1.

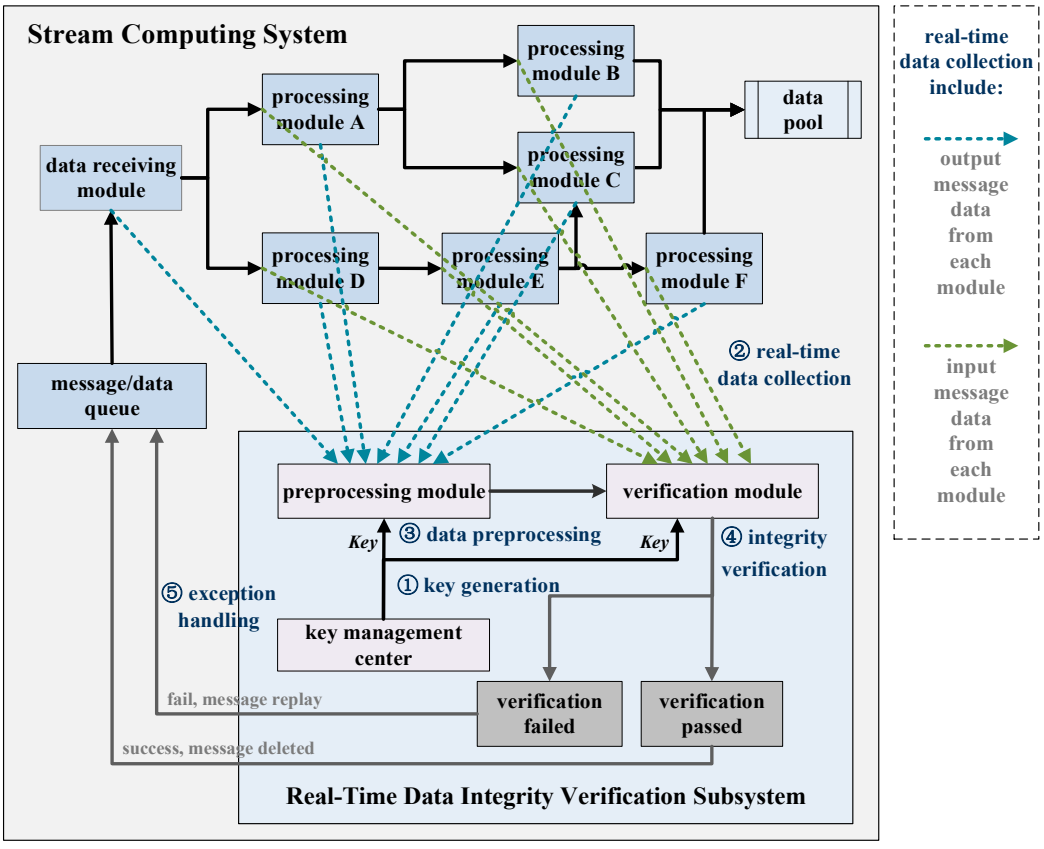


Figure 1. The proposed model of the real-time data integrity verification subsystem.

The real-time data integrity verification subsystem is constructed inside the stream computing system and consists of a key management center, a preprocessing module and a verification module. The input and output messages of each processing module in the stream system are collected into the data integrity verification subsystem for verification, and the results are fed back and processed in real time.

Each message data has a unique message ID (*MID*) and a session ID (*SID*) while passing through each processing module in the stream computing system. When a message data passes through different processing modules, the message data will be changed and the *SID* will also be changed accordingly, but the *MID* is the unique identifier of this message throughout its life cycle which will never be changed.

The working process of the real-time data integrity verification subsystem is as follows:

1. The phase of key generation: The key management center respectively sends the pre-generated keys to the preprocessing module and the verification module.
2. The phase of real-time data collection: The data collection point is added to the input end and output end of each processing module in the stream computing system, and the real-time collected data are sent to the verification module and the preprocessing module, respectively.
3. The phase of data preprocessing: In this stage, all message data collected at the output of each processing module in stream computing system are preprocessed, and verification tags are generated and sent to the message verification module.
4. The phase of data integrity verification: The verification module uses the key to perform integrity verification according to the collected input data of each processing module and the corresponding verification tag. If the integrity verification of the entire life cycle of the message is correct, it will return the success information and clear the relevant information in cache; if there is a verification exception or termination exception, it will actively alarm and handle the exception.
5. The phase of exception handling: According to the intermediate data of integrity verification in cache, the exception message is recovered, and the original message is replayed.

4. Preliminaries

In this section, we introduce Carter–Wegman MAC, pseudo-random function and symmetric cryptography mechanism, which serve as the basis of the proposed scheme.

4.1. Carter–Wegman MAC

The Carter–Wegman MAC method [35] can convert a one-time MAC to a many-times MAC. It is well known that the one-time MAC is safe from any attacker and is much faster than the MAC calculation based on other cryptographic primitives, but the one-time MAC calculation requires a key change at a time, which severely limits its practicality. It is significant to adopt the Carter–Wegman MAC construction method to transform the secure one-time MAC algorithm into a secure multiple-use MAC, while retaining the advantage of the one-time MAC security performance.

Let $(\boldsymbol{S}, \boldsymbol{V})$ be a secure one-time MAC in $(0,1)^n$, and let $F : K \times \{0,1\}^n \to \{0,1\}^n$ be a secure pseudo-random function, where K is the key space.

The Carter–Wegman MAC is defined as follows:

Definition 1. *The Carter–Wegman MAC is composed of three polynomial-time algorithms* $(\boldsymbol{Key}, \boldsymbol{Sign}, \boldsymbol{Verif})$:

$\boldsymbol{Key}(1^\lambda) \to (k_1, k_2)$ *is a polynomial-time algorithm for generating keys; it inputs the security parameter* λ*, and outputs the pair of keys* (k_1, k_2).

$\boldsymbol{Sign}(k_1, k_2, m) \to t$ *is a polynomial-time algorithm for generating MAC value; it inputs keys* (k_1, k_2) *and message* m*, computes the MAC value* $t = (r, F(k_1, r) \oplus \boldsymbol{S}(k_2, m))$*, where* r *is a random number and* $\boldsymbol{S}$ *is the algorithm of one-time MAC, and finally outputs* t.

Verif$(k_1, k_2, m, t) \rightarrow \{0,1\}$ *is a deterministic algorithm for verifying the MAC value; it inputs keys* (k_1, k_2)*, message* m *and the MAC value* t*, and runs and computes the algorithm* $\boldsymbol{V}(k_2, m, F(k_1, r) \oplus t)$ *where* $\boldsymbol{V}$ *is the algorithm of one-time MAC. If the result is correct, output 1; otherwise output 0.*

It has been proven [32] *that if* $(\boldsymbol{S}, \boldsymbol{V})$ *is a secure one-time MAC and F is a secure pseudo-random function, then the Carter–Wegman MAC is a secure MAC in* $(0,1)^{2n}$.

4.2. Pseudo-Random Function

The pseudo-random function (PRF) [36] is defined as follows:

Definition 2. *Assume that* H_k *is the set of all functions mapped from* I_k *to* I_k *where* I_k *is the set of bit strings of length k.*

Then, the pseudo-random function $F_k \subseteq H_k$ *satisfies:*

Indexability: With a unique index k, a function $f \in F_k$ *can be easily selected from the set according to k.*

Polynomial-time: Given a function $f \in F_k$ *and an input x, it can easily compute the value of* $f(x)$ *in polynomial-time.*

Pseudo-randomness: It is impossible to distinguish a function in F_k *from a function in* H_k *in polynomial-time, nor can it distinguish a value of* $f(x)$ *from a random number in* I_k *in polynomial-time.*

4.3. Symmetric Cryptography Mechanism

Symmetric cryptography [37], also known as the traditional cryptographic algorithm, refers to using the same key for encryption and decryption, or the ability that the encryption key and decryption key can be inferred from each other. Due to the advantages of high efficiency, fast computation speed and easy implementation, symmetric cryptography is widely used.

The formal definition is as follows:

Definition 3. *A symmetric cryptography mechanism is a five tuple* $(\boldsymbol{P}, \boldsymbol{C}, \boldsymbol{K}, \boldsymbol{E}, \boldsymbol{D})$ *that satisfies the condition:*

$\boldsymbol{P}$ *represents a finite set of all possible plaintexts.*

$\boldsymbol{C}$ *represents a finite set of all possible ciphertexts.*

$\boldsymbol{K}$ *represents the key space, which is a finite set of all possible keys.*

$\boldsymbol{E}$ *and* $\boldsymbol{D}$ *represent the encryption algorithm and decryption algorithm, respectively. For each* $k \in K, e_k \in E, d_k \in D$*, and* $e_k : P \rightarrow C, d_k : C \rightarrow P, x \in P$*, satisfied:* $d_k(e_k(x)) = x$.

5. Implementation

In this section, we provide the detailed design of the RT-DIV scheme and system, and then list advantages of the RT-DIV scheme.

5.1. Scheme Design

The definition and detailed construction of RT-DIV is given in the following sub-section.

5.1.1. The Definition of Real-Time Data Integrity Verification Scheme Based on Symmetric Key in Stream Computing System

Definition 4. *The Real-Time Data Integrity Verification Scheme based on symmetric key in stream computing system (RT-DIV) is composed of three polynomial-time algorithms* $(\boldsymbol{KeyGen}, \boldsymbol{TagGen}, \boldsymbol{VerifTag})$*:*

KeyGen$(1^{\lambda}) \rightarrow Key$ *is a probabilistic algorithm to generate keys; it takes a security parameter* λ *as input, and outputs the secret symmetric key Key.*

TagGen$(Key, M_{(MID,SID)}) \rightarrow T_{(MID,SID)}$ *is an algorithm to generate verification tags; it inputs the secret key Key and the collected output message data* $M_{(MID,SID)}$ *of processing module in the stream computing system, and generates the corresponding tag value* $T_{(MID,SID)}$.

VerifTag$(Key, T_{(MID,SID)}, M'_{(MID,SID)}) \rightarrow \{1,0\}$ *is a deterministic algorithm to verify the integrity verification tags; it inputs the secret key Key, the verification tag* $T_{(MID,SID)}$ *and the collected input message data* $M'_{(MID,SID)}$ *of the processing module with the same identifier of verification tag (i.e., the same MID and SID), and outputs whether the verification is passed.*

5.1.2. A Construction of Real-Time Data Integrity Verification Scheme Based on Symmetric Key in Stream Computing System

Let $f : \kappa \times I \rightarrow F_q$ be a pseudo-random function, and let q be the order of the finite field F_q, which size depends on the security parameter λ (usually the value is $q = 2^\lambda$), and let κ be the key space.

- ***KeyGen***$(1^\lambda) \rightarrow (k_1, k_2)$

Input the security parameter λ, choose the random number $k_1, k_2 \xleftarrow{R} \boldsymbol{\kappa}$, and output symmetric key pair $(k_1, k_2) \in \boldsymbol{\kappa}$.

- ***TagGen***$((k_1, k_2), M_{(MID,SID_i)}) \rightarrow T_{(MID,SID_i)}$

For all messages whose identifier is *MID* (i.e., the session identifier is SID_i, where $i = 1, \ldots, n$, and n is the total number of sessions), the preprocessing module calculates the verification tag:

$$T_{(MID,SID_i)} = f_{k_1}(MID||SID_i) \oplus (k_2 \cdot M_{(MID,SID_i)}) \tag{1}$$

where $||$ is the splicing operation, $\oplus$ is the XOR operation, $M_{(MID,SID_i)}$ is the message data collected from output port of processing module in stream computing system.

Output verification tag $T_{(MID,SID_i)}$.

- ***VerifTag***$((k_1, k_2), T_{(MID,SID_i)}, M'_{(MID,SID_i)}) \rightarrow \{1,0\}$

For the message data $M'_{(MID,SID_i)}$ collected from the input port of the processing module with the consistent identification *MID* and *SID* as verification tag, the verification module uses the key (k_1, k_2) to verify the message and tag as follows:

$$\text{Compute } a = T_{(MID,SID_i)} \oplus (k_2 \cdot M'_{(MID,SID_i)}) \tag{2}$$

$$\text{Compute } b = f_{k_1}(MID||SID_i) \tag{3}$$

Verify whether the equation $a = b$ holds; if true, output 1, otherwise output 0.

5.2. System Design

The RT-DIV subsystem is constructed from the RT-DIV scheme in five phases, which correspond to the working process of the real-time data integrity verification model (i.e., Figure 1).

The phase of key generation: The key management center runs the algorithm ***KeyGen*** to generate the symmetric key (k_1, k_2) and send it to the preprocessing module and the verification module, respectively.

The phase of real-time data collection: According to the identifier *MID* and *SID* of the message, the message data are collected from the output port and the input port of each processing module in stream computing system, and then sent to the preprocessing module and the verification module of the RT-DIV subsystem, respectively.

The phase of data preprocessing: The preprocessing module runs the algorithm ***TagGen***, calculates and generates a verification tag $T_{(MID,SID_i)}$ for each collected message $M_{(MID,SID_i)}$, and sends them to the verification module.

The phase of data integrity verification: The verification module receives the tag value $T_{(MID,SID_i)}$ from the preprocessing module, collects the message data $M'_{(MID,SID_i)}$ with the same identification *MID* and *SID* from the input port of the processing module in stream computing system, and then runs the algorithm ***VerifTag*** to verify the integrity of

the message data. If the verification is correct, the next stream processing of the message is carried out, and the intermediate data in the cache is deleted. If the verification fails, the message exception processing is carried out.

The phase of exception handling: When the subsystem receives the error information of verification, according to the *MID*, the error message is called out from the message queue and resent to the real-time stream computing system to reprocess.

5.3. Advantages

The main advantages of the RT-DIV scheme are as follows:

1. The previous research shows that the one-time MAC is safe from any attacker and is much faster than the MAC calculation based on other cryptographic primitives, but the one-time MAC calculation requires a key change at a time, which severely limits its practicality. This paper adopts the Carter–Wegman MAC construction method to transform the secure one-time MAC algorithm into a secure multiple-use MAC, while retaining the advantage of the one-time MAC security performance.
2. The RT-DIV scheme only uses pseudo-random function calculation, XOR calculation and one-time MAC calculation, mainly for bit calculation, multiplication and addition calculation operations. Compared with most current data integrity verification schemes, which are mainly used for multiplication, exponentiation or bilinear mapping calculations, the efficiency is significantly improved, and it is suitable for data integrity verification in stream real-time computing systems.
3. The RT-DIV scheme is based on symmetric cryptography mechanism, and uses a symmetric key for preprocessing and verification calculations, which is easy to implement and has high computational efficiency. Compared with other schemes based on asymmetric cryptography mechanism, the RT-DIV scheme is sufficiently efficient for real-time data.

6. Security Analysis

In this section, we introduce the security model and security analysis of RT-DIV scheme.

6.1. Security Model

We state the security model with a game that capture the property of data integrity.

Game 1. *Data integrity game*

Setup: The challenger runs $\boldsymbol{KeyGen}(1^{\lambda}) \rightarrow (k_1, k_2)$*, and keep* (k_1, k_2) *secret.*

Query: The adversary queries adaptively: It selects and sends a message M to the challenger, then the challenger runs $\boldsymbol{TagGen}((k_1, k_2), M) \rightarrow T$ *to compute and send the tag to the adversary. These queries can be executed multiple times. Then, the adversary stores the message set {M} together with the tag set {T}.*

Challenge: The challenger generates a challenge, then requests the adversary to generate a verification tag for the message M'*, where* $M' \notin \{M\}$*.*

Forge: The adversary computes a tag $T(T \in \{T\})$ *and returns it to challenger.*

If $\boldsymbol{VerifTag}((k_1, k_2), T, M') \rightarrow 1$*, then the adversary wins the game.*

6.2. Security Analysis

Theorem 1. *Assume that f is a secure pseudo-random function, then RT-DIV scheme is a secure data integrity verification scheme in the standard model.*

Proof of Theorem 1. The security simulation method is used to analyze the security of the scheme, specifically by using the pseudo-random function f and pure random number r to simulate and execute the data integrity game of the RT-DIV scheme in the real environment and the ideal environment, respectively.

In the real environment:

The challenger runs the algorithm ***KeyGen*** to generate the pair of keys (k_1, k_2) and keeps them private. The adversary selects the message M to ask the challenger for the verification tag. The challenger runs the algorithm ***TagGen*** to generate the tag $T = f_{k_1} \oplus (k_2 \cdot M)$ and sends it to the adversary. After performing multiple queries, the adversary has the message set $\{M\}$ and corresponding tag set $\{T\}$, and finally the adversary forges the message $M'(M' \notin \{M\})$ and tag $T(T \in \{T\})$, and sends them to the challenger, at which point the challenger uses the algorithm ***VerifTag*** to verify the correctness of message and tag.

In the ideal environment:

The operation of the challenger and the adversary is the same as the real environment, the only difference is that the random number r is used instead of the pseudo-random function f, and the generated tag is $T = r \oplus (k_2 \cdot M)$.

Assume that the adversary can forge M' to pass the verification, that is, under ideal conditions, the adversary can find a value M' that satisfies $T = r \oplus (k_2 \cdot M')$, i.e., $M' = (r \oplus T) \cdot k_2^{-1}$.

Since the adversary is unknown to k_2, and r is a pure random number, then the probability of successful forgery is:

$$\mathrm{A}^{\mathrm{ideal}}_{\mathbf{RT-DIV}} = \Pr[M'|M' = (r \oplus T) \cdot k_2^{-1}] = \Pr[M'|M' = R] \leq \frac{1}{2^{\lambda}} \tag{4}$$

where R is a purely random number.

Because f is a secure pseudo-random function, within polynomial-time, the adversary cannot distinguish whether the scheme is executed in the ideal environment or in the real environment.

Therefore, the probability of adversary forgery in the real environment:

$$\mathrm{A}^{\mathrm{real}}_{\mathbf{RT-DIV}} \cong \mathrm{A}^{\mathrm{ideal}}_{\mathbf{RT-DIV}} \leq \frac{1}{2^{\lambda}} \tag{5}$$

is negligible.

Proof completed. □

7. Experiments

Experimental environment: All data are stored in Kafka [38] message queue with version 2.8.1 and calculated in the storm framework [39] simulation environment, and the cryptographic library is OpenSSL [40] with version 0.9.7a. Because the results of each experiment will be slightly different (caused by environmental deviation), all data are obtained by averaging multiple experiments. Most existing schemes adopt the sampling method to achieve probabilistic security; for accuracy and fairness, these experiments use all messages for calculation, and the verification accuracy can reach 100%.

7.1. Performance Comparison of Various Schemes

The comparison of various schemes is shown in Table 1, where n is the total number of data chunks or messages, c is the number of data chunks sampled in a proof, w is additional storage such as verification tags stored on the server, l is the number of small data blocks in each large data chunks and L is the number of members in a group.

It can be seen from Table 1 that most of the traditional data integrity verification schemes require larger computational costs, communication costs and storage costs than the RT-DIV scheme and are not suitable for real-time stream computing systems. The RT-DIV scheme omits the computation cost of proof generation and communication bandwidth, thus greatly improving efficiency.

Compared with the data integrity verification scheme in the traditional storage mode, the RT-DIV scheme adopts a more lightweight calculation operation. In the tag generation phase and verification phase, there are only XOR, multiplication and pseudo-random function calculation, respectively, while the existing solutions mainly use multiplication,

exponentiation or bilinear mapping and other calculations. For this reason, computing efficiency is significantly improved, and it can be applied to real-time stream computing systems.

Table 1. The comparison of the performance of data integrity verification schemes.

	Based on RSA [4]	**Based on HVT [8]**	**Based on BM [15]**	**Based on TP [19]**	**Based on EC [21]**	**Based on Group [26]**	**RT-DIV**
Data integrity	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sampling verification	Yes	Yes	Yes	Yes	Yes	Yes	No
Private verification	Yes	Yes	No	No	Yes	Yes	Yes
Symmetric key	No	No	No	No	No	No	Yes
Pre-processing	$O(n)$	$O(t \log n)$	$O(t \log n)$	$2O(n)$	$O(n)$	$O(n)$	$O(n)$
Proof generation	$O(c)$	$O(c)$	$O(c)$	$2O(c)$	$O(c)$	$O(c)$	/
Proof/Tag verification	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Communication	$O(1)$	$O(l)$	$O(l)$	$2O(l)$	$O(1)$	$O(L)$	/ [1]
Storage	$O(n+w)$	$O(n+w)$	$O(n+w)$	$O(n+w)$	$O(n+w)$	$O(n+w/L)$	$O(1)$
Real-time	No	No	No	No	No	No	Yes
Data stream	No	No	No	No	No	No	Yes

[1] This refers to the communication cost required during the data integrity verification phases, excluding the communication during the data collection and data replay phases of the stream computing system itself.

7.2. Overall Performance of the RT-DIV Scheme

Since the RT-DIV scheme is built inside the stream computing system, in order to ensure the real-time requirements of the system, the RT-DIV scheme omits the phase of proof generation, which is unlike other data integrity verification schemes. In the simulation experiment, the key generation phase is one-time and pre-generated, which does not affect the performance of the RT-DIV scheme. The time consumed in the real-time data collection phase entirely depends on the generating and processing time of the message in the original stream computing system and has no impact on the efficiency of the RT-DIV scheme. Thus, the time of key generation and data collection is ignored. In the phase of data preprocessing, the experiment only records the calculation time of generating tags and ignores the I/O times such as data retrieval, reading and transmission, etc. In the phase of data integrity verification, the experiment only records the calculation time of verification between messages and tags while ignoring the I/O time. The time consumed in the exception handling phase is determined by the efficiency of original system, so it is not considered in the performance of the RT-DIV scheme.

As can be seen from Figure 2, with the increase of message concurrency, the preprocessing time (TagGen) and verification time (VerifTag) increase linearly, which is because RT-DIV scheme calculates and verifies all messages. Nevertheless, this scheme is still sufficiently efficient to be used in real-world applications; for example, when the message concurrency is 1000, the preprocessing time is approximately 200 milliseconds (0.2 S) and the verification time is approximately 150 milliseconds (0.15 S). In short, this solution can meet the requirement of real-time integrity verification in the majority of stream computing systems.

Figure 3 records the calculation time of preprocessing and verification for five random messages, wherein m1-pre and m1-verify respectively refer to the preprocessing calculation time and verification calculation time of message 1, and the rest can be performed in the same manner. In practical experiments, due to the significant impact of I/O time to record a single message, we randomly selected one message to execute 100 times and calculated the average time of this message to reduce the impact of I/O. As can be seen from Figure 3, the entire process of data integrity verification of each message consumes about 0.35 milliseconds (3.5×10^{-4} S), which is a very efficient calculation. Therefore, the RT-DIV scheme can meet the real-time requirements of stream computing systems.

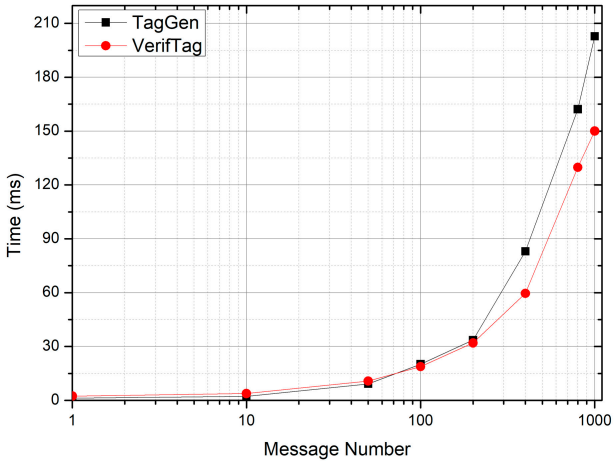


Figure 2. Overall performance of the RT-DIV scheme.

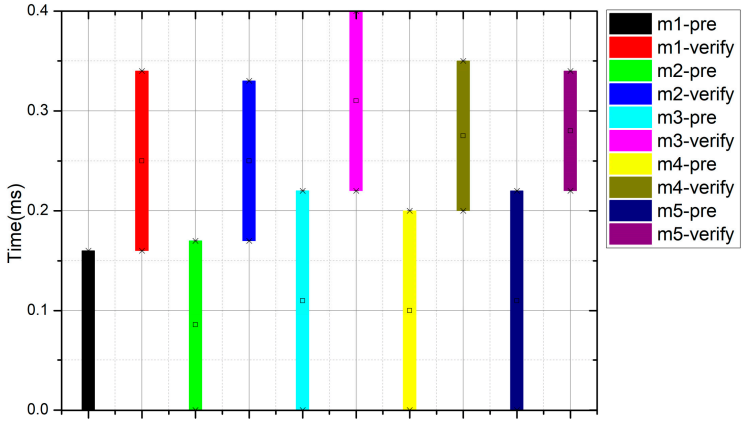


Figure 3. Timing diagram of multiple messages.

7.3. Comparison of Preprocessing Efficiency

In order to ensure the fairness of the experiments, this paper selects two classical data integrity verification schemes, respectively based on RSA (S-PDP [4]) and EC (E-PDP [21]), as the reference schemes. In the experiment, the RT-DIV scheme and the reference PDP schemes are used to preprocess the same number of messages, and the calculation time is recorded.

Figure 4 shows the comparison of preprocessing efficiency between RT-DIV, S-PDP and E-PDP under the condition of different message concurrency. In the experiment, only the calculation time of tag generation is recorded, while other I/O times are ignored.

As can be seen from Figure 4, with the increase of message concurrency, the preprocessing time increases linearly in all schemes, which is because these schemes calculate all messages in the preprocessing phase. However, the RT-DIV scheme has much higher computational efficiency than other schemes, with a difference of about tens to hundreds of times, and this advantage becomes more obvious as the message concurrency increases. This is because the RT-DIV scheme uses efficient XOR computation, while the S-PDP scheme uses exponentiation operation and the E-PDP scheme uses homomorphic addition operation on elliptic curves. Therefore, the computational efficiency of the RT-DIV scheme has been significantly improved.

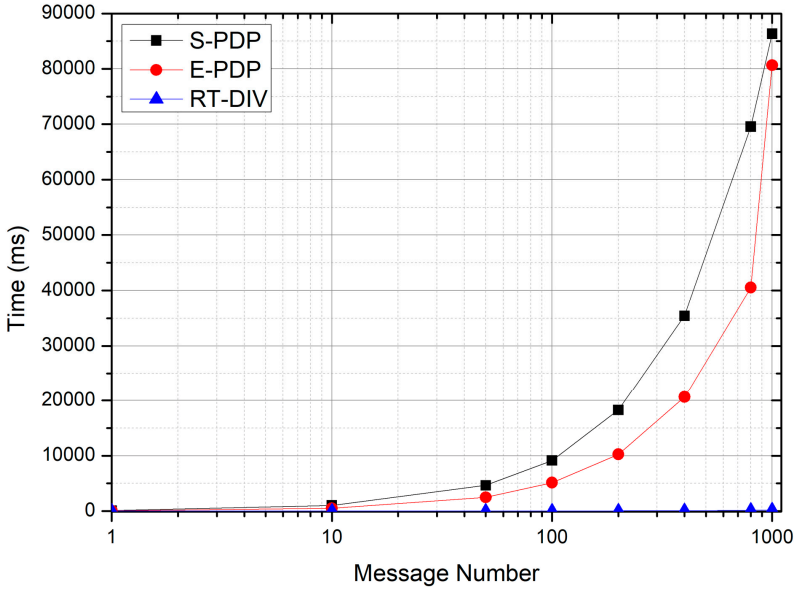
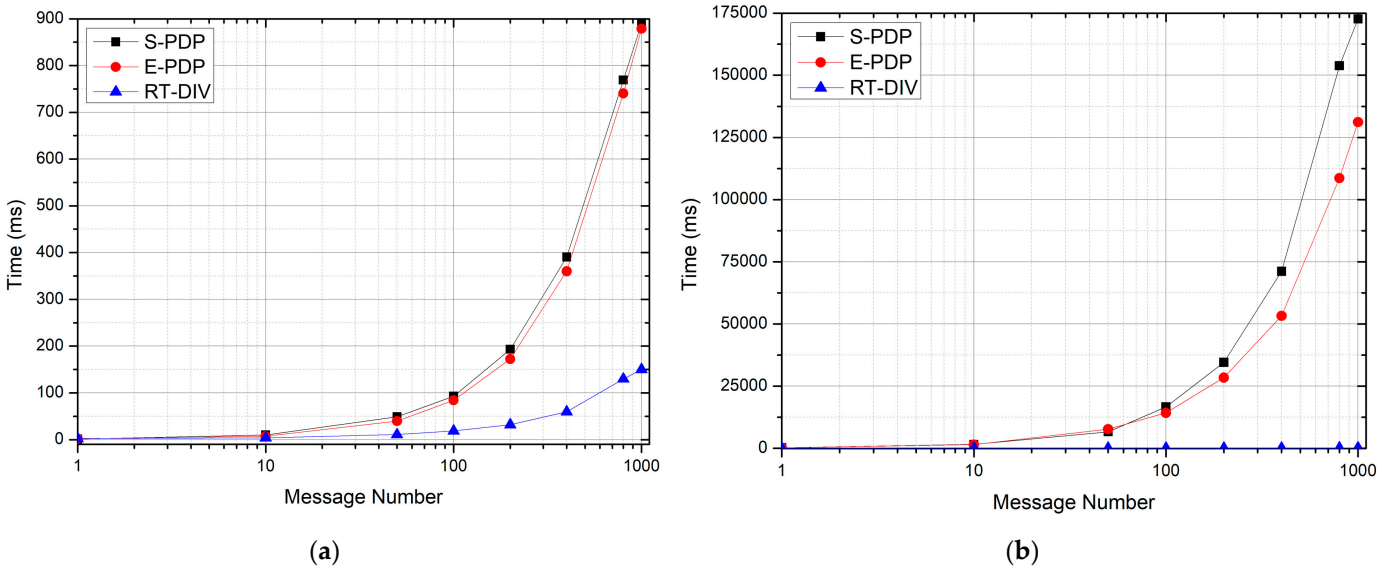


Figure 4. Comparison of preprocessing efficiency between RT-DIV and other schemes.

7.4. Comparison of Verification Efficiency

In this experiment, we respectively compared the calculation time of single verification and the calculation time of proof generation add verification. The reason for designing the experiment in this way is that the RT-DIV scheme does not need the phase of proof generation and can directly verify the tags, while other reference schemes need to generate proof before verification. In the experiment, all schemes are used to verify the same number of messages, and the calculation time is recorded.

As can be seen from Figure 5, with the increase of message concurrency, the verification time increases linearly in all schemes, and the RT-DIV scheme has higher computational efficiency than other schemes, which is because the RT-DIV scheme uses efficient XOR operations for verification calculation. Comparing Figure 5a,b, it can be concluded that the proof generation process leads to lower efficiency of the reference schemes. Therefore, the RT-DIV scheme is more suitable for data integrity verification in real-time stream computing systems.



(**a**) (**b**)

Figure 5. Comparison of verification efficiency between RT-DIV and other schemes: (**a**) comparison of verification; (**b**) comparison of proof generation add verification.

7.5. Storage Cost

In the data integrity verification scheme of the traditional storage mode, the size of the tag corresponding to each data depends on the security parameter, and the traditional schemes generally store firstly and then calculate; thus, the verification tags and data need to be stored for a long time, so the required storage cost increases linearly with the amount of data.

The storage cost required by the RT-DIV scheme is relatively small. The temporary storage required of a tag is only related to the security parameter, which has nothing to do with the length of message itself. When the message stream is processed completely and the verification result is correct, the intermediate data of the message will be deleted in the cache and will not occupy additional storage cost.

7.6. Communication Cost

In the data collection phase, the communication cost is equal to the total message number, and the communication cost is $O(n)$. It is not related to the RT-DIV scheme, but rather to the overhead of the stream computing system itself. In preprocessing and verification phases, all operations are executed inside the RT-DIV subsystem; thus, there is no communication cost. In the exception handling phase, whether it is the case of message replay caused by verification failure or the case of message deletion caused by successful verification, only one message needs to be delivered, and the communication cost is $O(1)$.

8. Conclusions

Aiming at the difficult issue that real-time data integrity cannot be efficiently guaranteed in stream computing systems, this paper adopts the Carter–Wegman MAC method, pseudo-random function and symmetric cryptography mechanism to construct a Real-Time Data Integrity Verification scheme based on symmetric key in stream computing system (RT-DIV), which uses the XOR calculation to improve the computational efficiency. Then, a security analysis is given under the standard model, and experiments are conducted in a simulated environment; the experimental results show that the RT-DIV scheme can effectively guarantee the integrity of a real-time data stream. Furthermore, the RT-DIV scheme gives a new thought and new approach for the field of real-time data integrity verification and lays the foundation for the secure application of the stream computing system.

In future work, the main research direction that need to be considered is reasonable disposal of data: in some stream computing systems, deviated data or duplicate messages may be reasonably discarded; the data integrity verification scheme needs to be able to distinguish this situation and efficiently guarantee the integrity of valid data without generating a false alarm.

Author Contributions: H.W. conceived the idea, designed the experiments, and wrote the paper; W.Z., B.Z. and Y.L. provided the discussions and article correction. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the General Project of Science and Technology of Beijing Municipal Education Commission (KM202110005026, KM202210005023).

Data Availability Statement: The data for this study are available from the author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tan, S.; Jia, Y.; Han, W.H. Research and Development of Provable Data Integrity in Cloud Storage. *J. Comput.* **2015**, *38*, 164–177. (In Chinese)
2. Sun, D.W.; Zhang, G.Y.; Zheng, W.M. Big Data Stream Computing: Technologies and Instances. *J. Softw.* **2014**, *25*, 839–862. (In Chinese)
3. Storm Acker. Available online: https://www.cnblogs.com/DreamDrive/p/6671194.html (accessed on 16 June 2023).

4. Ateniese, G.; Burns, R.; Curtmola, R. Provable Data Possession at Untrusted Stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 31 October–2 November 2007; pp. 598–609.
5. Ateniese, G.; Di, P.R.; Mancini, L.V.; Tsudik, G. Scalable and Efficient Provable Data Possession. In Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, Istanbul, Turkey, 22–25 September 2008; ACM Press: New York, NY, USA, 2008.
6. Erway, C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic Provable Data Possession. *ACM Trans. Inf. Syst. Secur.* **2009**, *17*, 213–222.
7. Wang, H.; Zhu, L.; Xu, C.; Lilong, Y. A Universal Method for Realizing Non-Repudiable Provable Data Possession in Cloud Storage. *Secur. Commun. Netw.* **2016**, *9*, 2291–2301. [CrossRef]
8. Shacham, H.; Waters, B. Compact Proofs of Retrievability. In *Advances in Cryptology (ASIACRYPT)*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 90–107.
9. Juels, A.; Kaliski, B.S. Pors: Proofs of Retrievability for Large Files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 2 November–31 October 2007; ACM Press: New York, NY, USA, 2007; pp. 584–597.
10. Curtmola, R.; Khan, O.; Burns, R.; Ateniese, G. MR-PDP: Multiple-Replica Provable Data Possession. In Proceedings of the 2008 the 28th International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008; pp. 411–420.
11. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Ensuring Data Storage Security in Cloud Computing. In Proceedings of the 2009 17th International Workshop on Quality of Service, Charleston, SC, USA, 13–15 July 2009; pp. 1–9.
12. Wang, H. An External Data Integrity Tracking and Verification System for Universal Stream Computing System Framework. In Proceedings of the 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Republic of Korea, 17–20 February 2019; pp. 32–37. [CrossRef]
13. Wang, H.; Zu, B.; Zhu, W.; Li, Y.; Wu, J. On the Design and Implementation of the External Data Integrity Tracking and Verification System for Stream Computing System in IoT. *Sensors* **2022**, *22*, 6496. [CrossRef]
14. Zhu, Y.; Wang, H.; Hu, Z. Efficient Provable Data Possession for Hybrid Clouds. In Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 4–8 October 2010; pp. 756–758.
15. Zhu, Y.; Hu, H.; Ahn, G.; Yu, M. Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 2231–2244. [CrossRef]
16. Zhu, Y.; Ahn, G.J.; Hu, H. Dynamic Audit Services for Outsourced Storages in Clouds. *IEEE Trans. Services Comput.* **2013**, *6*, 227–238.
17. Wang, C.; Wang, Q.; Ren, K.; Lou, W.J. Privacy-Preserving public auditing for data storage security in cloud computing. In Proceedings of the 2010 IEEE INFOCOM, San Diego, CA, USA, 15–19 March 2010; IEEE: New York, NY, USA, 2010; pp. 1–9. [CrossRef]
18. Wang, C.; Chow, S.M.; Wang, Q.; Ren, K.; Lou, W.J. Privacy-Preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375. [CrossRef]
19. Armknecht, F.; Bohli, J.; Karame, G.; Liu, Z.; Reuter, C. Outsourced Proofs of Retrievability. *IEEE Trans. Cloud Comput.* **2014**, *9*, 286–301. [CrossRef]
20. Hanser, C.; Slamanig, D. Efficient Simultaneous Privately and Publicly Verifiable Robust Provable Data Possession from Elliptic Curves. In Proceedings of the 2013 International Conference on Security and Cryptography (SECRYPT), Reykjavik, Iceland, 29–31 July 2013; pp. 1–12.
21. Wang, H.; Zhu, L.; Wang, F.; Lilong, Y.; Chen, Y.; Liu, C. An Efficient Provable Data Possession based on Elliptic Curves in Cloud Storage. *Int. J. Secur. Its Appl.* **2014**, *8*, 97–108. [CrossRef]
22. Tate, S.R.; Vishwanathan, R.; Everhart, L. Multi-user Dynamic Proofs of Data Possession Using Trusted Hardware. In Proceedings of the Third ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 18–20 February 2013; pp. 353–364.
23. Wang, B.; Li, B.; Li, H. Oruta: Privacy-Preserving Public Auditing for Shared Data in The Cloud. *IEEE Trans. Cloud Comput.* **2014**, *2*, 43–56. [CrossRef]
24. Wang, B.; Li, B.; Li, H. Panda: Public Auditing for Shared Data with Efficient User Revocation in The Cloud. *IEEE Trans. Serv. Comput.* **2015**, *8*, 92–106. [CrossRef]
25. Zhu, L.; Wang, H.; Xu, C.; Sharif, K.; Lu, R. Efficient Group Proof of Storage with Malicious-Member Distinction and Revocation. *IEEE Access* **2019**, *7*, 75476–75489. [CrossRef]
26. Wang, H.Y.; Zhu, L.H.; Lilong, Y.J. Group Provable Data Possession with Deduplication in Cloud Storage. *J. Softw.* **2016**, *27*, 1417–1431. (In Chinese)
27. Li, Y.; Yu, Y.; Min, G.; Susilo, W.; Ni, J.; Choo, K.R. Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems. *IEEE Trans. Dependable Secur. Comput.* **2017**, *16*, 72–83. [CrossRef]
28. Xu, Y.; Ren, J.; Zhang, Y.; Zhang, C.; Shen, B.; Zhang, Y. Blockchain Empowered Arbitrable Data Auditing Scheme for Network Storage as A Service. *IEEE Trans. Serv. Comput.* **2019**, *13*, 289–300. [CrossRef]
29. Guo, W.; Zhang, H.; Qin, S.; Gao, F.; Jin, Z.; Li, W.; Wen, Q. Outsourced Dynamic Provable Data Possession with Batch Update for Secure Cloud Storage. *Future Gener. Comput. Syst.* **2019**, *95*, 309–322. [CrossRef]

30. Guo, W.; Qin, S.; Gao, F.; Zhang, H.; Li, W.; Jin, Z.; Wen, Q. Dynamic Proof of Data Possession and Replication with Tree Sharing and Batch Verification in the Cloud. *IEEE Trans. Serv. Comput.* **2020**, *15*, 1813–1824. [CrossRef]
31. Li, T.; Ren, S.; Wang, G.; Meng, Q. Cloud-edge-device Collaborative Integrity Verification Scheme Based on Chameleon Authentication Tree for Streaming Data. *Netinfo. Secur.* **2022**, *1*, 37–45. (In Chinese)
32. Zhou, C.; Wang, L.; Wang, L. Lattice-based provable data possession in the standard model for cloud-based smart grid data management systems. *Int. J. Distrib. Sens. Netw.* **2022**, *18*, 137–147. [CrossRef]
33. Wang, Q.; Cheng, C.; Xu, R.; Ding, J.; Liu, Z. Analysis and Enhancement of a Lattice-Based Data Outsourcing Scheme with Public Integrity Verification. *IEEE Trans. Serv. Comput.* **2022**, *15*, 2226–2231. [CrossRef]
34. Qi, Y.; Yang, Z.; Luo, Y.; Huang, Y.; Li, X. Blockchain-Based Light-Weighted Provable Data Possession for Low Performance Devices. *Comput. Mater. Contin.* **2022**, *73*, 17. [CrossRef]
35. Carter, L.; Wegman, M.N. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.* **1979**, *18*, 143–154. [CrossRef]
36. Boyle, E.; Goldwasser, S.; Ivan, I. Functional Signatures and Pseudorandom Functions. In *PKC 2014: Public-Key Cryptography*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8383, pp. 501–519.
37. Stinson, D.R. *Cryptography Theory and Practice*; CRC Press: Boca Raton, FL, USA, 1995.
38. Kafka. Available online: https://kafka.apache.org/ (accessed on 16 June 2023).
39. Storm. Available online: https://storm.apache.org/ (accessed on 16 June 2023).
40. Openssl. Available online: https://www.openssl.org/ (accessed on 16 June 2023).