

An External Data Integrity Tracking and Verification System for Universal Stream Computing System Framework

Wang Hongyuan*

* China Academy of Electronics and Information Technology, LoongCloud Common Product Research Institute
wanghongyuan78@163.com

Abstract— With the development of the cloud computing and big data, massive data based computing and storage have emerged as core technologies. Moreover, a key problem to ensure data availability and analysis accuracy is the data integrity which has received extensive attention. Compared to the mature solution in the batch computing, the data integrity in the stream computing did not receive due attention. However, the characters of timeliness, volatility, suddenness and disorder of big data make the data integrity in the stream computing a very difficult problem. Based on the framework of current general stream computing system, in this paper, we construct an externally data integrity tracking and verification system for the real-time tracking and analysing of the data content and processing path of each message, and the discovering of the data integrity problem. We use the common cryptographic algorithm to construct a universal data integrity tracking and verification scheme outside the stream computing system to ensure the integrity and consistency of the data. Then we give the formal security analysis in standard model, and perform simulations in cloud environment. Results show that the proposed system can verify data integrity efficiently without compromising the efficiency.

Keywords— Stream computing system, data integrity verification, homomorphic message authentication code, pseudo-random function, security proof

I. INTRODUCTION

With the rapid development of information technologies and applications such as cloud computing, the internet of things, and social media, the human society is stepping into the age of big data. How to utilize the big data, and perform the mining of the important information is one of the most important issues in the age of big data. In general, the big data has three characteristics: low value density, irregular distribution, and deep information hiding. These characteristics lead to unprecedented challenges and opportunities for the data calculation and analysis. Moreover, the explosive growth of the big data also brings various types of data and management as well as data protection methods. Therefore, how to maintain and guarantee the integrity and consistency of the data throughout the processing of the data in a big data environment has become an urgent problem.

The rapid development of big data applications has brought the rapid evolution of data computing models. Currently,

batch offline computing and stream real-time computing are the main computing modes of big data: the batch computing (such as Hadoop) is mainly a computing model for static persistent data. Usually, the data is stored first, and then the data and computational logic are distributed to the distributed compute nodes for data calculation. The stream computing is mainly a data flow-oriented computing model. Compared to the former, the difference is that the stream computing does not store all the data, and calculate the data in transit. The stream computing is widely used in the field of the financial banking and Internet fields. Common stream computing architectures include Storm and Spark Streaming.

The current data integrity verification schemes are well studied in the batch computing mode, and the technology and methodology are relatively mature. Based on whether the fault-tolerant recovery measures of data in the mechanism, it can be divided into two categories: Provable Data Possession (PDP) [1, 2, 3, 4, 5] and Proofs Of Retrievability (POR) [6, 7].

However, the current research on data integrity and consistency in big data stream computing mode is deficient, due to the characteristics of real-time, volatile, disordered and infinite in the stream big data. These features make the data integrity and consistency very difficult to tackle, and lead the data in the stream computing system easy loss, repeatability and inconsistent, which resulting in that data integrity and consistency verification faces unprecedented challenges [8].

In summary, the research on data integrity and consistency in stream computing system has important practical significance and practical value.

Most current stream computing systems have a message acknowledgment mechanism (acker) [9, 10], which is only used to ensure that each message is completely processed, but the integrity of the message data is not guaranteed. Also, the acker module is inside the system, which means that if the complex verification calculation runs on the acker, it is easy to affect the efficiency of the real-time calculation of the message. Furthermore, since the stream calculation process is not persisted, the processing path of the history message cannot be viewed, so that the incomplete message is hard to be reproduced.

Our paper contributes as follows: Based on the current general stream computing system framework, an external

tracking and verification system is constructed to analyse and verify the data content and processing path of each message in real-time, proactively discover data or messages incomplete (missing or error), automatically alarm, and recover erroneous data. Then we gave the formal security analysis in standard model, and perform simulation experiments in cloud environment, the result shows that the external data integrity tracking and verification system can verify data integrity efficiently without compromising original computing efficiency.

II. MODEL CONSTRUCTION

A. Overall design

The overall architecture design of the external data integrity tracking verification system under the stream computing framework is shown in Fig. 1.

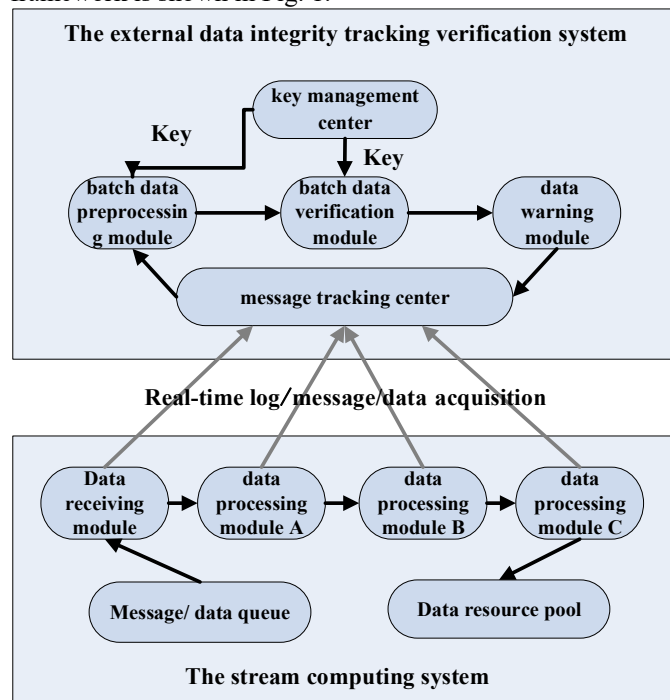


Figure 1. General architecture design of the external data integrity tracking and verification system under the stream computing framework

The external data integrity tracking and verification system is closely coupled with the stream computing system, and is mainly composed of a message tracking center, a key management center, batch data pre-processing module, batch data verification module and data warning module. The external side provides an interface coupled with the stream computing system for real-time data acquisition and reception; on the other hand, it is coupled with an external query module to facilitate manual query by the user or developer.

B. Work steps

The tracking and verification process of messages is divided into the following steps:

1) Real-time data acquisition: The externally coupled universal tracking and verification system adds a real-time

data acquisition interface to each module of the stream computing system, and records and sends the message data and status to the real-time tracking and verification system.

2) Message tracking: Each message in the tracking data center is classified and pre-processed with the message ID as a unique identifier, and all the trace data of the same message are merged together to form a batch of data of the message.

3) Key generation: The key management center distributes the pre-generated key to the data pre-processing module and the data verification module.

4) Batch data pre-processing: Generating a verification tag using the key and a cryptographic algorithm. The each output data is processed to generate a verification tag, and each module for receiving the message performs integrity verification on the received message data.

5) Batch data integrity verification: Integrity verification of the batch data of each message is performed using the cryptographic algorithm, verification tag and verification key corresponding to the phase of pre-process.

6) Active alarm and message recovery: If the verification is passed, it indicates that the message has been processed completely and correctly, then the related data of the message is deleted; if the verification fails, the data integrity is abnormal and the message is not processed correctly, then the alarm is performed. According to the record of the message and related data in the external tracking verification system, the message is resent to the message queue and recalculated and processed in the stream computing system.

III. DATA INTEGRITY VERIFICATION SCHEME IN STREAM COMPUTING SYSTEM

We use preliminaries such as Homomorphic message authentication code [11], Pseudo-Random Function [12] and Negligible function to construct the Data Integrity Verification Scheme in Stream Computing System (referred to S-DIV). and then we used the S-DIV scheme to construct the external data integrity tracking and verification system, which is closely coupled with the stream computing system.

First, the message format is introduced. Each message consists of four parts: the sending and receiving flag (*Flag*), the message ID (*Mid*), the session ID (*Sid*), and the message content (*M*).

The formal expression is $M_{sid} = (Flag \parallel Mid \parallel Sid \parallel M)$, where $\parallel$ is the splicing, as shown in Fig. 2.

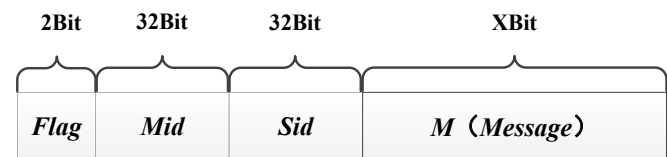


Figure 2. Message format

Flag is used to indicate whether the message is sent or received: if *Flag* = 00, the message is collected by the sending port, and the message is sent; if *Flag* = 01, the message is

collected by the receiving port, and the message is received; if it is other, the message is wrong, re-acquisition.

Mid is a unique identifier of the message, which is used to avoid message duplication. In actual system, the Mid is obtained by pseudo-random function calculation. When the message is verified, the message and the Mid are deleted together, so that 32-bit random numbers is able to keep the message unduplicated in the system.

Sid is the session identifier for messages transmission in different modules. When the message is in a computing module A, the module A performs calculation processing on the message. When the processed message is ready to be sent, the Sid is modified and added to the corresponding position of the message, and sending the new message to the next module, so repeated that the message Sid transmitted between each module is different, and the message Sid of the sender of the previous module and the receiver of the next module is the same, to verify that the message has not been tampered or lost during transmission.

A. Formal definition of S-DIV

Definition 1 The Data Integrity Verification Scheme in Stream Computing System is composed of four polynomial time algorithms (**KeyGen**, **PreGen**, **ProofGen**, **VerifProof**) as shown below:

- **KeyGen**(1^λ) $\rightarrow$ Key is a probabilistic algorithm used to generate keys, which enter security parameters λ and output key.
- **PreGen**($Key, M_{(Mid, Sid)} \rightarrow T_{(Mid, Sid)}$) is an algorithm for generating a verification tag that inputs a key and message data $M_{(Mid, Sid)}$ to produce a tag value for the message data $T_{(Mid, Sid)}$.
- **ProofGen**($Key, \{M\}_{Mid}, \{T\}_{Mid} \rightarrow \rho$) is used to generate a proof of data integrity, which inputs a key, a set of all session message $\{M\}_{Mid} = \{M_{(Mid, sid_1)}, \dots, M_{(Mid, sid_n)}\}$, and corresponding set of tags $\{T\}_{Mid} = \{T_{(Mid, sid_1)}, \dots, T_{(Mid, sid_n)}\}$, resulting in a proof of data possession ρ , which can be considered as data aggregation of $\{M\}_{Mid}$ and $\{T\}_{Mid}$.
- **VerifProof**($Key, \rho \rightarrow \{1, 0\}$) is a deterministic algorithm used to verify the proof of data integrity, it inputs the key and the proof ρ , and outputs whether ρ is a correct proof.

According to an S-DIV scheme, an external data integrity tracking and verification system is constructed, which is divided into the following four stages:

- Key generation phase: The key management center runs an algorithm **KeyGen** to generate a key, and sends the key to the batch data pre-processing module and the batch data verification module respectively.
- Batch data pre-processing phase: The pre-processing module runs an algorithm **PreGen**, calculates each

received message $M_{(Mid, Sid)}$, generates a verification label $T_{(Mid, Sid)}$, and sends it to the batch data verification module.

- Batch data challenge and verification phase: the batch data verification module receives all the tag data of the message $\{T\}_{Mid}$ from the pre-processing module, and collects all session messages of the message $\{M\}_{Mid}$ from the receiving module. Then according to the message ID, all the message data $\{M\}_{Mid}$ with the same Mid and the corresponding tag data $\{T\}_{Mid}$ are aggregated, and the algorithm **ProofGen** is executed to generate the integrity proof ρ of the message $\{M\}_{Mid}$, then running algorithm **VerifProof** to verify the integrity proof ρ , and if the verification passes, the tracking data center and the stream computing system deletes the message data in each caches, otherwise performing the message alarm and recovery phase.
- Message alarm and recovery phase: When the data alarm recovery module receives the error message, the error message is picked out from the batch data verification module. According to Mid and Sid , the original message data is found and sent to the stream computing system, and the stream computing system replays the message.

B. The concrete structure of S-DIV

Let $f : \kappa \times I \rightarrow F_q$ be a pseudo-random function, q is the order of finite field F_q , and its size depends on the security parameter λ (for example: $q = 2^\lambda$). Then the key space of the S-DIV is $\kappa = F_q^3$ and the label space is $F_q \cup \{\perp\}$.

- **KeyGen**(1^λ) $\rightarrow (a, b, c)$:

Input the security parameters λ , output key $(a, b, c) \in \kappa$, where a, b, c is the value randomly selected from F_q .

- **PreGen**($(a, b, c), M_{(Mid, Sid)} \rightarrow T_{(Mid, Sid)}$):

Calculate the tag value as follows:

$$a \cdot M_{(Mid, Sid)} + f_b(Mid \parallel Sid) = c \cdot T_{(Mid, Sid)}$$

Where $\parallel$ stands for splicing, and the label value $T_{(Mid, Sid)}$ can be efficiently calculated by solving the one-dimensional equation.

Finally, output the label $T_{(Mid, Sid)}$.

- **ProofGen**($Key, \{M\}_{Mid}, \{T\}_{Mid} \rightarrow \rho$)

The messages and tags are aggregated based on all messages ID (i.e., the same total number of messages and tags), resulting in a proof of integrity.

Set the number of session messages z , select a random value $k \in F_q$ for $1 \leq j \leq z$:

Calculate the pseudorandom parameters of each session message: $v_j = f_k(j)$.

Then calculate two sets of data separately:

$$\omega_{Mid} = v_1 M_{Mid, Sid_1} + v_2 M_{Mid, Sid_2} + \dots + v_z M_{Mid, Sid_z}$$

$$\tau_{Mid} = v_1 T_{Mid, Sid_1} + v_2 T_{Mid, Sid_2} + \dots + v_z T_{Mid, Sid_z}$$

Finally, outputs the proof ω_{Mid} and τ_{Mid} .

- **VerifProof** (Key, ρ) $\rightarrow \{1, 0\}$

Analogously for $1 \leq j \leq z$:

Calculate the pseudorandom parameters of each session message: $v_j = f_k(j)$.

The verification process is as follows:

$$\omega = \omega_{Mid}, \tau = \tau_{Mid}$$

Calculation:

$$\sigma = v_1 f_b(Mid \parallel Sid_1) + v_2 f_b(Mid \parallel Sid_2) + \dots + v_z f_b(Mid \parallel Sid_z)$$

Verify that the equation $a \cdot \omega + \sigma = c \cdot \tau$ is true. If it is true, the verification passes and outputs 1 otherwise, otherwise it outputs 0.

Notices:

1. The identifier Mid is randomly generated when the message is generated. The unique identifier of the message is used to distinguish the actual message of the different content data. Mid remains unchanged throughout the entire life of the message, and can effectively resist collision in the actual system. After the message is completely processed and the integrity verification is correct in the entire cycle, all message data is deleted along with Mid .

2. The identifier Sid is generated when the message is transmitted between modules, and each session (transferred from one module to another is called a session) is used to distinguish from the same message with different content or the different processing phase. After the message is completely processed and the integrity verification is correct in the entire cycle, all message data is deleted along with Sid .

3. The algorithm **ProofGen** can be regarded as an aggregate function, which is used to aggregate all the different status messages containing the same Mid message and their corresponding verification tags to perform batch verification, thereby improving the calculation efficiency.

IV. SECURITY ANALYSIS

A. Security model

Firstly, define the data integrity game to analyse the security of the S-DIV scheme.

Definition 2 Data Integrity Game

Setup: The challenger runs the algorithm to generate the key and keeps the key private.

Ask: The adversary adaptively selects a series of messages $M_{(Mid, Sid)}$ to ask the challenger: the adversary selects a message data block $M_{(Mid, Sid)}$ and sends it to the challenger, then the

challenger runs **PreGen**($Key, M_{(Mid, Sid)}$) to calculate the tag value of the message $T_{(Mid, Sid)}$ and sends for the adversary, this query process can be executed multiple times. The adversary stores all data sets $\{M\}_{Mid} = \{M_{(Mid, Sid_1)}, M_{(Mid, Sid_2)}, \dots, M_{(Mid, Sid_z)}\}$ and tag value sets $\{T\}_{Mid} = \{T_{(Mid, Sid_1)}, T_{(Mid, Sid_2)}, \dots, T_{(Mid, Sid_z)}\}$.

Challenge: The challenger requests the adversary to prove the integrity of all blocks of this message $M_{(Mid, Sid_1)}, \dots, M_{(Mid, Sid_z)}$.

Forgery: The adversary runs the algorithm **ProofGen**, calculates an integrity proof ρ and sends it to the challenger.

If **VerifProof** (Key, ρ) $\rightarrow 1$, then the adversary wins the game.

Intuitively, the probability that an adversary wins a game is negligible close to the probability that the decimator ϵ extracts all of the message data blocks $M_{(Mid, Sid_1)}, \dots, M_{(Mid, Sid_z)}$ corresponding to the message.

B. Security proof

Theorem 1 Assume that f is a safe pseudo-random function, then S-DIV scheme is a secure data integrity verification scheme under the standard model.

Proof:

(1) Correctness and reliability

In an S-DIV scheme, correctness is the verification of legal proof, and reliability is the verification of illegal proof.

(2) Data integrity:

Simplify the parameters when proving data integrity, set them to 1, omit all subscripts, and perform data integrity games in real-world and ideal environments using pseudo-random functions and random numbers respectively.

In an real-world environment, we performed experiment according to Data Integrity Game normally.

In an ideal environment, we use a random number instead of a pseudo-random function, then $B = r$. All random numbers are recorded, and calculated during the verification process.

Suppose the adversary can complete the verification in the case that M has changed, which means in an ideal environment, the adversary can successfully find the changed T' , makes $T' = c^{-1}aM' + c^{-1}B$, where M' is changed from M .

Since a and c are randomly generated keys and B is a pure random number, the inequation can be obtained as follows:

$$A_{s-DIV}^{ideal} = \Pr[T' \mid T' = c^{-1}aM' + c^{-1}B] = \Pr[T' \mid T' = r_1M' + r_2] \leq \frac{1}{2^i}$$

Where r_1, r_2 are random numbers.

According to the assumption: f is a safe pseudo-random function, then the adversary cannot distinguish whether the scheme is executed in the real environment or in the ideal environment.

Therefore, in the real environment, the probability of the forgery is: $A_{S-DIV}^{real} \cong A_{S-DIV}^{ideal} \leq \frac{1}{2^i}$, the probability is negligible.

The proof is completed.

V. EXPERIMENTAL ANALYSIS

Experimental environment: All data is calculated in real time using the simulated stream computing system on the cloud server. All experiments are run on the Red Hat Enterprise Linux AS release 4 operating system with 8G RAM. The cryptographic library used by the algorithm is OpenSSL with version 0.9.7a. The experiment uses a speed of 5400, and the interface type is a SATA 6gb/s hard disk. Since the test results of each experiment will be slightly different (caused by environmental errors), all data results are obtained after multiple tests and comparisons.

Different from the integrity verification method in the storage mode (the sampling method is mostly used in the storage mode), this test uses all the message data for calculation, and the verification accuracy rate can reach 100%.

A. Overall performance test of the S-DIV scheme

In the S-DIV scheme, each message data corresponds to a verification tag. In the message collection phase, the data center determines a storage queue according to the value of the flag, and then the pre-processing module processes the message in the storage pool $Flag = 00$, according to each message identifier Mid and Sid , calculates the corresponding message tag value. Finally, the verification module selects the corresponding message data from the storage pool $Flag = 01$, according to the two identifiers Mid and Sid of the message, verifies the integrity according to the verification tags.

Fig. 3 shows the performance test results of the S-DIV scheme under different message concurrency.

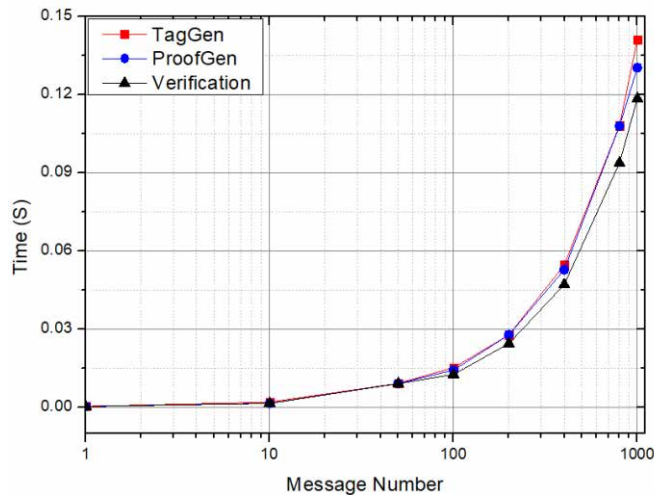


Figure 3. Overall performance test of S-DIV

The time of the data acquisition phase depends entirely on the time of different message data generation in the original stream computing system, and has no reference value for the performance of the S-DIV scheme, so the data acquisition

time is ignored. The pre-processing time only records the time that verification tags are calculated, and ignores the time that the message data is retrieved, looked up, read, and transmitted (hereinafter referred to as I/O time). Challenge and verification time refers to the time for calculating the proof and verifying tags of all messages, while ignores the I/O time. The message alarm and recovery time depends on the efficiency of the original stream computing system and is therefore not considered in the performance testing experiments of the S-DIV solution.

As can be seen from the figure, as the number of messages increases, the pre-processing time (tag generation time), challenge time (proof generation time), and verification time (verification calculation time) all increase linearly because the S-DIV scheme pre-processes and challenges all the messages, therefore, as the amount of message data increased, the calculation amount of all stages increased linearly. However, when the message concurrency is small (below 100), the tag generation calculation time, the proof calculation time, and the verification calculation time are all in the millisecond level; when the message concurrency is large (1000 or more), the tag generation calculation time, proof calculation time and verification calculation time are both around 0.1s, which is suitable for the stream computing system in real time, and since the S-DIV scheme is the external module of the system, the internal performance of the stream computing system has little effect.

B. Storage cost and communication cost

(1) Storage cost

In the PDP scheme^[1] of the traditional storage mode, the tag length of each data block is same, which only depending on the security parameters, regardless of the data block size. If the file block size is 4 KB and the modulus value is 1024 bits, the additional storage space is 3.125% of the original storage, and the operation mode in the traditional storage mode is stored firstly and then calculated, so the data needs to be stored for a long time, and the verification tag is also stored according to the corresponding data block. Therefore when the amount of data is larger, the extra storage is increased, the total storage is also greater.

The storage cost required for the S-DIV scheme is relatively small. When the message flow is calculated in the stream computing system, the storage space required for each tag is only related to the security parameters, regardless of the length of the message. After the verification result is correct in the external verification system, the message flow is deleted in the cache of the stream computing system, and no additional storage space is occupied.

(2) Communication cost

In the data acquisition phase, the communication bandwidth is equal to the total data volume of the message flow, and the communication cost is $O(n)$. In the pre-processing and challenge verification phase, all calculation operations are performed inside the verification system, so there is no consideration of the data transmission cost. After the error is found, the message recovery phase only needs to send the

original message to the message buffer pool of the stream computing system, and the communication cost is $O(1)$.

In summary, the S-DIV scheme can guarantee data integrity efficiently without affecting the efficiency of the original stream computing system, and can be widely used in the reality system.

VI. CONCLUSIONS

In this paper, we proposed an external tracking and verification system based on the current general stream computing system framework, to analyse and verify the data integrity of each message in real-time. Then we gave the formal security analysis in standard model, and perform simulation experiments in cloud environment. The result shows that S-DIV scheme can verify data integrity efficiently without compromising original computing efficiency.

In the future research, the research of the universal data integrity tracking and verification system can be applied to several mainstream stream computing systems. The system promotion and application scenarios need further analysis.

REFERENCES

- [1] Ateniese G, Burns R, Curtmola R. Provable data possession at untrusted stores. Proceedings of the 14th ACM conference on Computer and communications security (CCS 2007). Alexandria USA, 2007: 598–609.
- [2] Zhu Y, Hu H, Ahn G, Yu M. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. on Parallel Distributed Systems, 2012: 2231–2244.
- [3] Wang C, Chow SM, Wang Q, Ren K, Lou WJ. Privacy-Preserving public auditing for secure cloud storage. IEEE Trans. on Computers, 2013, 62(2): 362–375.
- [4] Wang H, Zhu L, Wang F, et al. An Efficient Provable Data Possession based on Elliptic Curves in Cloud Storage [J]. International Journal of Security and Its Applications, 2014, 8(5): 97–108.
- [5] Wang H, Zhu L, Xu C, et al. A Universal Method for Realizing Non-Repudiable Provable Data Possession in Cloud Storage [J]. Security & Communication Networks, 2016, 9(14): 2291–2301.
- [6] Juels A, Kaliski BS. Pors: Proofs of retrievability for large files. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. ACM Press, 2007: 584–597.
- [7] Shacham H, Waters B. Compact proofs of retrievability. In: Proc. of the Advances in Cryptology (ASIACRYPT 2008). Berlin, Heidelberg: Springer-Verlag, 2008: 90–107.
- [8] Sun Dawei, Zhang Guangyan, Zheng Weimin. Big Data Streaming Computation: Key Technologies and System Examples[J]. Journal of Software, 2014, 25(4): 839–862.
- [9] <https://www.csdn.net/article/2015-03-09/2824135>. 《Three frameworks for streaming big data processing: Storm, Spark, and Samza》
- [10] <https://www.cnblogs.com/DreamDrive/p/6671194.html>. 《 Storm's acker message confirmation mechanism》
- [11] Agrawal S, Dan B. Homomorphic MACs: MAC-Based Integrity for Network Coding [J]. In: Proceedings of ACNS, 2009: 292–305.
- [12] Dan B, Waters B. Constrained Pseudorandom Functions and Their Applications [J]. Lecture Notes in Computer Science, 2013: 280–300.



Wang Hongyuan (M), born in 1988-07, received a PhD. of computer science from Beijing Institute of Technology in 2017, now working in China Academy of Electronics and Information Technology, her research area is Cloud Computing Security and Big Data Security.