Accountable MapReduce in Cloud Computing

Zhifeng Xiao and **Yang Xiao**
The University of Alabama
Tuscaloosa, AL 35487-0290 USA
Emails: zxiao1@ua.edu, yangxiao@ieee.org

***Abstract*—In this paper, we propose Accountable MapReduce, which forces each machine to be held responsible for its behavior. We set up a group of auditors to perform an Accountability Test (A-test) which will check all working machines and detect malicious nodes in real time.**

I. Introduction

MapReduce [1] has been widely used as a powerful parallel data processing model. It has efficiently solved a wide range of large-scale computing problems, including distributed grep, distributed sort, web-link graph reversal, web-access log stats, document clustering, machine learning, etc. Cloud Computing presents a unique opportunity for batch-processing and analyzing terabytes of data which would otherwise take hours to finish [19]. Most cloud providers (e.g., Google, Yahoo!, Facebook, etc.) adopt MapReduce to build multitenant computing environments. Usually, cloud customers have a large set of data to be processed under certain time constraints. They must provide a client MapReduce program and data that are ready to be processed. Cloud providers maintain thousands of working machines to fulfill the data processing jobs submitted by their customers. As an example [15], The New York Times used 100 Amazon EC2 instances and a Hadoop [2] application to process 4TB of raw image TIFF data (stored in S3) into 11 million finished PDFs in the space of 24 hours at a computation cost of about $240 (not including bandwidth).

In such a computing environment, the cloud customers outsource their data to the cloud, which performs the storing and computing operations required by the customers. Customers must therefore fully trust the cloud provider. However, a cloud provider cannot guarantee that its data center (which may have thousands of working machines) is 100 percent trustworthy. Some machines may become malicious if they are attacked and controlled by hackers; malicious machines will not faithfully carry out the tasks assigned to them. As a result, the processing result is no long correct and trustworthy. In the New York Times example, malicious nodes may mess up the image conversion process so that the PDFs do not match the original TIFF images. It is even hard for the New York Times to check if these PDFs are correctly converted because of the tremendous data size. In this paper, we explore the use of accountability to address this problem.

Accountability has been a longstanding concern of trustworthy computer systems [6], and it has recently been elevated to a first class design principle for dependable networked systems [4, 5]. Accountability implies that an entity should be held responsible for its own actions or behaviors [7]. In the MapReduce scenario, accountability means that all working machines (i.e., mappers and reducers) will be responsible for the tasks they have completed.

In this paper, we propose building an Accountable MapReduce to make the cloud computing platform trustworthy. We use an Accountability Test (A-test) which checks all working machines when a job is undertaken and detects malicious nodes in real time. The A-test is performed by a group of trusted machines, which are called the Auditor Group (AG). The AG takes advantage of the determinism of user MapReduce program to replay the tasks executed by working machines. The MapReduce framework makes it possible for an auditor to acquire the input data block and processing result without the knowledge of the working machine. Therefore, auditors are free to replay the tasks that have been finished. If the replay output does not match the original output, it means that the worker is returning bad results, as evidenced by the combination of the task, the input, the original output, and the replay output.

A challenge of Accountable MapReduce is to reduce the overhead introduced by the A-test. In theory, the A-test can guarantee detection of any misbehavior by fully duplicating each task, causing the processing time to at least double. To make the A-test more efficient, we abandon pursuing 100% accountability, which guarantees exposure of every malicious node but has a high cost. We adopt P-Accountability [14], which quantifies the degree of accountability. We use P-Accountability for system efficiency. Based on the batch-processing property of MapReduce, the performance of A-test with P-Accountability can be greatly improved with the tradeoff of decreasing the degree of accountability by less than 1%.

We summarize the contributions of this paper as follows:

1. We propose building an Accountable MapReduce to detect malicious nodes. Verifiable evidence will be generated to ensure that the malicious nodes cannot deny their behavior.
2. Instead of pursuing perfect accountability, A-test allows the system to achieve P-Accountability with less overhead and higher performance.

The rest of this paper is structured as follows: related work will be reviewed in Section II; we introduce MapReduce in Section III; we address the accountability issue in MapReduce and define the problem we focus on in Section IV; our solution, accountable MapReduce, is discussed in Section V; We conclude the paper in Section VI.

II. Related Work

Its ability to process data intensive tasks has made MapReduce increasingly important in distributed computing areas. Chu et al. [12] applied MapReduce to machine learning on multicore platforms. He et al. [16] implemented Mars, a MapReduce framework, in graphics processors. Papadimitriou et al. [18] applied MapReduce to the area of data mining; they designed Disco, which is a practical approach for distributed data pre-processing. Ekanayake et al. [17] adopted the MapReduce

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

technique for two scientific data analyses, high energy physics data analyses, and K-means clustering. Existing work focuses on utilizing MapReduce to solve different problems in various domains. However, few have considered accountability issues in MapReduce. Accountable MapReduce is an attempt to address the issue of untrustworthy nodes and their behavior in MapReduce.

Wei et al. [8] present SecureMR, a practical service integrity assurance framework for MapReduce. SecureMR provides a decentralized, replication-based integrity verification scheme for ensuring the integrity of MapReduce in open systems. SecureMR is intended to achieve 100 percent integrity of MapReduce, which can affect its performance. We believe that, for some applications, efficiency is prior to 100 percent integrity. Therefore, in this paper, instead of pursuing 100 percent accountability, we allow the customers choose the level of accountability that they need based on their applications. It turns out that decreasing the expectation of accountability slightly causes the system overhead to drop tremendously and greatly improves performance.

Accountability has been regarded as an important issue in cloud computing. Trustworthy relationships between cloud provider and cloud customers have been addressed in [20, 21]. The customer places his computation and data on machines that he cannot directly control; the provider agrees to run a service whose details that he does not know [20]. Therefore, accountability is employed to determine whether the Service Level Agreement (SLA) is fulfilled or not. If it is not, evidence should be provided in order to prove which unit is responsible. MapReduce is a popular computing framework in cloud platforms. In this paper, we build Accountable MapReduce, which solves the subset of problems addressed in [20]. Accountable MapReduce is able to detect malicious workers and provide verifiable evidence.

There are other related work in security [22-93].

III. MapReduce Background

A. Programming model

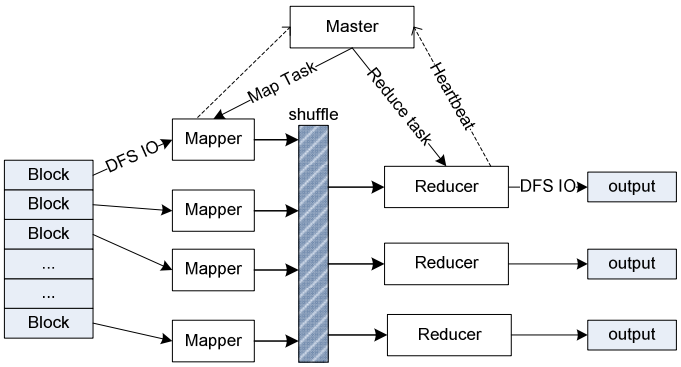

Fig. 1: MapReduce Working Flow

With the MapReduce programming model, programmers only need to specify two functions: *Map* and *Reduce*. The Map function receives a key/value pair as input and generates intermediate key/value pairs to be further processed. The Reduce function merges all the intermediate key/value pairs associated with the same (intermediate) key and then generates final output.

There are three main roles: the master, mappers, and reducers. The single master acts as the coordinator responsible for task scheduling, job management, etc. MapReduce is built upon a distributed file system (DFS) which provides distributed storage. Fig. 1 shows the execution process of MapReduce. The input data is split into a set of *M* blocks, which will be read by *M* mappers through DFS I/O. Each mapper will process the data by parsing the key/value pair and then generate the intermediate result that is stored in its local file system. The intermediate result will be sorted by the keys so that all pairs with the same key will be grouped together (the shuffle phase). If the memory size is limited, an external sort might be used to deal with large amounts of data at one time. The locations of the intermediate results will be sent to the master who notifies the reducers to prepare to receive the intermediate results as their input. Reducers then use Remote Procedure Call (RPC) to read data from mappers. The user defined reduce function is then applied to the sorted data; basically, key pairs with the same key will be reduced in some way, depending on the user defined reduce function. Finally the output will be written to DFS.

B. Fault tolerance

MapReduce is designed to be fault tolerant because failures are common phenomena in large scale distributed computing.

1) Worker failure

The master pings every *mapper* and *reducer* periodically. If no response is received for a certain amount of time, the machine is marked as failed. The ongoing task and any tasks completed by this mapper will be re-assigned to another mapper and executed from the very beginning. Completed *reduce* tasks do not need to be re-executed because their output is stored in the global file system.

2) Master failure

Since the master is a single machine, the probability of master failure is very small. MapReduce will re-start the entire job if the master fails.

IV. Problem Statement

A. Attack model

Fault-tolerance will address node failures, such as a worker not responding to the master, or a worker machine being totally crashed, etc. To address node failures, the master learns the task fail event and then takes further action (e.g., it re-executes the failed map/reduce task in another machine.). However, fault-tolerance is unable to detect a malicious node intending to alter the Map/Reduce function and return inaccurate results. We illustrate this type of attack with an example.

Wordcount is a typical MapReduce application. Its job is to count the occurrences of each word in large input text data. If there are malicious working machines in the system, the output file, which contains word counts of every word, is inaccurate.

Consider the wordcount example in Fig. 2. Assume that the system is free of malicious nodes. There are three mappers, each of which maps one line of the file. After the mapping function, we have the map output as the intermediate result. Then the intermediate results will be shuffled (sorted by key) and read by reducers (five, in this case), which reduce the intermediate results and generate the final output.

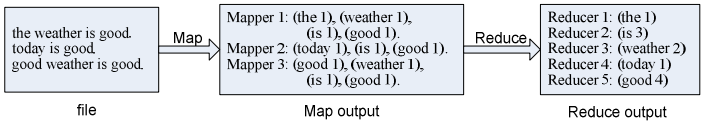

Fig. 2: a wordcount example of MapReduce

If all units faithfully execute their tasks, the final output will be accurate. Otherwise, we cannot trust the results because the malicious units may alter part of the results. For example, if a mapper is malicious, it has multiple ways to alter the output: 1)

filter some keys, 2) create keys that do not exist in the input file, 3) modify the value intentionally, etc. A malicious reducer is able to cause similar errors.

To solve the above problem, we propose Accountable MapReduce, which ensures that

1) Malicious nodes intending to alter the processing result will be exposed; additionally, Accountable MapReduce is able to provide verifiable evidence to ensure that the detection is repudiable.
2) The failed jobs will be re-directed to another working node until it is verified as correct.

V. ACCOUNTABLE MAPREDUCE

A. Design Principles

A key function of Accountable MapReduce is detecting malicious nodes which generate inaccurate results. We now present the principles that guided our design:

1. Accountability mechanism is should be concealed so that malicious nodes are unaware of what is happening. Since we assume that machines may be fully controlled by attackers and they may be smart enough to discover this; if a machine is aware of anything abnormal, it takes countermeasures to cover itself. It follows that we leave any machine alone when an A-test is ongoing.
2. The overhead brought by the accountability mechanism should be minimized to reduce processing time.
3. When a malicious node is caught, the system should be able to provide verifiable evidence to show that the node is indeed being malicious.

B. Assumptions

The design of Accountable MapReduce is based on the following assumptions:

1. We assume that the data set provided by cloud customers can be processed by MapReduce.
2. We assume that the Auditor Group (AG) is a trustworthy domain, meaning that the machines of the AG are free of any malicious action.
3. A worker cannot be reclaimed until the entire job is completed. When the customer confirms the job is done, all machines will be released back to the cloud.
4. A malicious node randomly performs bad actions. This means that the faulty parts of processing result also distribute randomly throughout the entire result. In addition, there may be multiple faulty parts in the processing result, once a fault area is found; the test will stop, because we already have evidence to expose the bad node.
5. All input data, intermediate results, and output data will not be removed until the entire job is finished.

C. Accountable MapReduce design

1) Correctness checking scheme

PeerReview [9] provides accountability for distributed systems. It assumes that every node in the system is a deterministic state machine, i.e., for some certain input, the output will be the same. Two critical technologies employed by PeerReview are tamper-evident logging and witnessing. A tamper-evident log is implemented by a hash chain, which guarantees that any modification to the log will be detected so that a node has to record its behavior faithfully. A witness (which is also a regular node) is able to check the correctness of other nodes that it is witnessing by replaying the log files kept in each node. As a result, malicious nodes will eventually be detected and exposed to all other correct nodes. PeerReview is applicable to most distributed applications. However, it is not applicable to MapReduce. The major concern is overhead. First, the input of a large task might be at the TB or even PB level (even though there are 1000s workers, the split task also has large workload), and the output depends on the input, meaning that all input and output events will have to be logged so that the witness is capable of replaying log files and checking their correctness. Second, for witness checking, a node has to upload its log segment to multiple witnesses, which is extremely bandwidth-consuming.

The idea of correctness checking is simple. As shown in Fig. 2, assume that the auditor is a trustworthy node; both the worker and the auditor are regarded as deterministic state machines, and the protocol is running on them. If the input data is the same (adopting tamper-evidence logs to ensure it), the output should be the same as well. After comparing output (from the worker) and output' (from the auditor), the system is able to determine whether the worker is good or not. The evidence is the combination of input, output', and output; additionally, it is verifiable to any other auditors.

2) Auditor Group

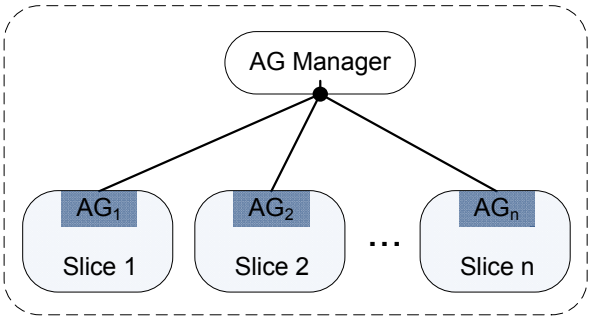

Fig. 3: Auditor Group in cloud platform

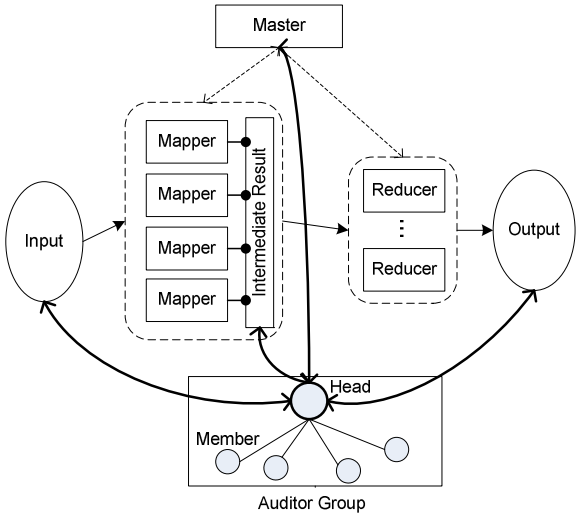

Fig. 4: Accountability Test

The ***Auditor Group (AG)*** carries out Accountability Test (i.e., A-test, which will be introduced next) to detect malicious nodes. Normally, as shown in Fig. 3, cloud resource will be divided into multiple slices, each of which is rented by a customer. A slice is a group of working machines assigned to a customer. We maintain an AG manager for the entire cloud, and one AG for each slice that runs MapReduce. The reason of associating each slice with one AG is to conserve the privacy and independence of customers.

The AG Manager is a coordinator that conducts AG creation, management, and disposal. After the AG manager becomes aware of the customer's data size, timing, and other requirements, it will determine the AG size and then create an AG for the slice.

Each AG is internally structured as a cluster. The head node is the Group Head (GH), and the member node is the Group Member (GM). The GH picks up workers as test targets randomly. The master has a protocol with the GH to provide all information needed for an A-test. The GH assigns A-test tasks to the available GMs, which are the actual machines that accomplish the tasks and report their status.

3) Accountability Test

A-Test is built upon the correctness checking scheme we adopt in this paper. The ***Auditor Group (AG)*** is the entity fulfilling the A-test. The AG consists of a set of trustworthy workers assigned by the AG manager; these are machines dedicated to performing the A-test as shown in Fig. 4. The working flow of A-test is as follows:

1. The A-test is started when map/reduce starts.
2. A group of idle auditors will be chosen as the Auditor Group of a certain slice. The AG forms a cluster and only the GH interacts with the master. The GH is thus able to request the job information from the master. Therefore, it knows 1) the input data and output (i.e., the intermediate data before it is shuffled and sorted) of each mapper; 2) the input (i.e., the intermediate data after it is shuffled and sorted) and output (i.e., the final result) of each reducer. This information is essential for the auditors to check the correctness of each worker.
3. After the job begins, the GM will receive test tasks from the master, which will be notified once a worker finishes its task. Based on the processing sequence of map/reduce, the mappers will finish first and then the reduce process is started. Therefore, in the initial period of time, mappers will be tested and then reducers will be tested after the mappers.
4. After the GH receives a test task of checking a worker, it finds an available GM to carry out the test. Each check is executed as follows:
   a) The GM will find corresponding input and output based on the task type (i.e., map/reduce).
   b) The GM will process the input data again and compare its output with the original one to check for inconsistency. If there is, it indicates that the worker being tested is malicious.
   c) The GM reports the test result to the GH, which will further report back to the master.
5. If a worker is detected to be malicious, the task it was assigned will be resubmitted to another worker so that the job continues.

4) A-test with P-Accountability

If the auditor is trustworthy and it processes all the input of the worker, then the system can definitely determine whether the worker is malicious or not. This means that the task assigned to the worker is fully replicated. In the system view, the entire job will be executed twice, once by regular workers and once by the auditors. However, the high overhead of processing the job one time shows that it will take even longer and bring more overhead to process it twice. Therefore, instead of pursuing perfect accountability, the A-test provides P-Accountability [14], which gives the customers options. P-Accountability trades the degree of accountability for efficiency.

Definition **P-Accountability**: we define P-Accountability as the probability that a malicious worker will be detected when it tampers with the processing result.

Let P_A denote P-Accountability, and let w denote the number of records in an input file, which can be either a raw data block for a map operation or a partition of intermediate results for a reduce operation. Assume that for any one record, a node has probability p_m of being malicious (i.e., tampering with the result); this will cause the corresponding output to be inaccurate. Variable x means that, if we want to achieve P_A, we need to check at least x records. If $P_A = 1$, x is equal to w, meaning that the entire input file is checked.

$$1-(1-p_m)^x \geq P_A \qquad (1)$$

We have

$$x \geq \left\lceil \log_{(1-p_m)}(1-P_A) \right\rceil \qquad (2)$$

If $P_A = 0.9999$ and $p_m = 0.01$, we have x = 917, which means that only 917 records need to be checked. Under the assumption that a malicious node randomly (with probability p_m) tampers with the map/reduce result, we observe that x will not be affected by input data size and only p_m and P_A will be related to x. Fig. 5 shows how x changes when P_A increases from 0 to 1. When P_A increases, the tester needs to check more records to achieve a certain degree of P_A. We also observe that a smaller p_m indicates that there are more records a tester needs to check because the malicious node has less of a chance to tamper with the Map/Reduce operation.

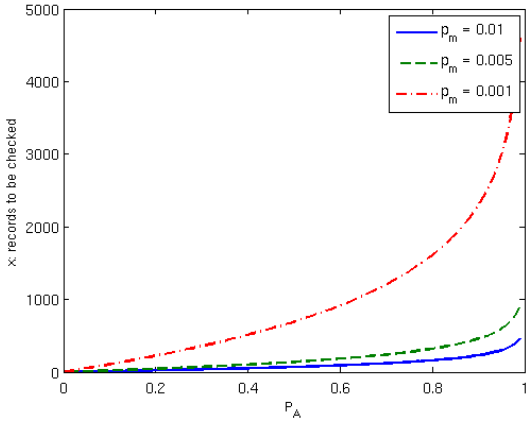

Fig. 5: P_A vs. x

Some features of A-test are as follows:

1. It is practical to implement the A-test, which makes the most of the existing properties of MapReduce. One important task of A-test is to acquire the input and output data of mappers/reducers, and the master has already kept this information.
2. It is an online test, meaning that the malicious nodes will be detected as early as possible. The flow of A-test ensures that a worker will be tested once it finishes.
3. Workers do not know they are being tested. Therefore, it is hard to take countermeasures to hide bad behavior.
4. With P-Accountability, A-test will be very efficient since a

lower P-Accountability will greatly cut down the records that need to be checked.

One limitation is that false positives may occur if P-Accountability is less than 1. In real world MapReduce applications, we can adjust the parameters so that the probability of a false positive is close to zero.

VI. Conclusion

In this paper, we proposed Accountable MapReduce as an additional component for the current MapReduce model. Accountable MapReduce employs an Auditor Group to conduct an A-test on every worker in the system. If malicious behavior occurs, the AG is able to detect it and provide verifiable evidence. To improve the performance, we introduce P-Accountability in A-test to trade the degree of accountability with efficiency.

Acknowledgement

This work was supported in part by the US National Science Foundation (NSF) under grants CNS-0737325, CNS-0716211, CCF-0829827, and CNS- 1059265.

References

[1] J. Dean and S.Ghemawat. 2004. *Mapreduce: simplified data processing on large clusters*. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation. USENIX Association, Berkeley, CA, USA.

[2] http://hadoop.apache.org/

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, 2008, pp. 107-113.

[4] A.R. Yumerefendi and J.S. Chase, "The role of accountability in dependable distributed systems," Proc. of HotDep, 2005.

[5] A.R. Yumerefendi and J.S. Chase, "Trust but verify: accountability for network services," Proc. of 11th workshop on ACM SIGOPS 2004, p. 37.

[6] Department of Defense. Trusted Computer System Evaluation Criteria. Technical Report 5200.28-STD, Department of Defense, 1985.

[7] Y. Xiao, "Flow-Net Methodology for Accountability in Wireless Networks," IEEE Network, Vol. 23, No. 5, Sept./Oct. 2009, pp. 30-37.

[8] W. Wei, J. Du, T. Yu, and X. Gu, "SecureMR: A Service Integrity Assurance Framework for MapReduce," Proceedings of the 2009 Annual Computer Security Applications Conference, 2009, pp. 73–82.

[9] A. Haeberlen, P. Kouznetsov, and P. Druschel, "PeerReview: Practical accountability for distributed systems," Proc. of ACM SIGOPS 2007.

[10] I. Roy, S. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and privacy for MapReduce," Proc. USENIX Symposium on Networked Systems Design and Implementation, 2010, pp. 297–312.

[11] J. Schlesinger, "Cloud Security in Map/Reduce," 2009. http://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-jason_schlesinger-cloud_security.pdf

[12] C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference, 2007, p. 281 - 288.

[13] A. Wieder, P. Bhatotia, A. Post, and R. Rodrigues, "Brief announcement: modelling MapReduce for optimal execution in the cloud," Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing, 2010, pp. 408–409.

[14] Z. Xiao and Y. Xiao, "P-Accountable Networked Systems," INFOCOM IEEE Conference on Computer Communications Workshops, 2010, pp. 1–5.

[15] D. Gottfrid, "Self-service, Prorated Super Computing Fun!". The New York Times. http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/?scp=1&sq=self%20service%20prorated&st=cse.

[16] B. He, W. Fang, Q. Luo, N.K. Govindaraju, and T. Wang, "Mars: a MapReduce framework on graphics processors," Proceedings of the 17th international conference on Parallel architectures and compilation techniques, 2008, pp. 260–269.

[17] J. Ekanayake, S. Pallickara, and G. Fox, "Mapreduce for data intensive scientific analyses," IEEE Fourth International Conference on eScience, 2008. eScience'08, 2008, pp. 277–284.

[18] S. Papadimitriou and J. Sun, "Disco: Distributed co-clustering with Map-Reduce: A case study towards petabyte-scale end-to-end mining," Eighth IEEE International Conference on Data Mining, 2008. ICDM'08, 2008, pp. 512–521.

[19] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and others, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[20] A. Haeberlen, "A case for the accountable cloud," ACM SIGOPS Operating Systems Review, vol. 44, 2010, pp. 52–57.

[21] C. Wang and Y. Zhou, "A Collaborative Monitoring Mechanism for Making a Multitenant Platform Accountable." Hotcloud 2010.

[22] Z. Zhuang, Y. Li, and Z. Chen, "Enhancing Intrusion Detection System with proximity information," International Journal of Security and Networks, Vol. 5, No.4 pp. 207 - 219, 2010.

[23] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Efficient decision tree for protocol analysis in intrusion detection," International Journal of Security and Networks, Vol. 5, No.4 pp. 220 - 235, 2010.

[24] K. R. Schrader, B. E. Mullins, G. L. Peterson, and R. F. Mills, "An FPGA-based system for tracking digital information transmitted via Peer-to-Peer protocols," International Journal of Security and Networks, Vol. 5, No.4 pp. 236 - 247, 2010.

[25] Z. Chen, C. Chen, and Q. Wang, "On the scalability of Delay-Tolerant Botnets," International Journal of Security and Networks, Vol. 5, No.4 pp. 248 - 258, 2010.

[26] Y. Guo and S. Perreau, "Detect DDoS flooding attacks in mobile ad hoc networks," International Journal of Security and Networks, Vol. 5, No.4 pp. 259 - 269, 2010.

[27] H. Guo, Y. Mu, X.Y. Zhang, and Z.J. Li, "Enhanced McCullagh-Barreto identity-based key exchange protocols with master key forward security," International Journal of Security and Networks, Vol. 5, No.2/3 pp. 173 - 187, 2010.

[28] A. O. Richard, A. Ahmad, and K. Kiseon, " Security assessments of IEEE 802.15.4 standard based on X.805 framework," International Journal of Security and Networks, Vol. 5, No.2/3 pp. 188 - 197, 2010.

[29] Y. Dong, S. Hsu, S. Rajput, and B. Wu, "Experimental analysis of application-level intrusion detection algorithms," International Journal of Security and Networks, Vol. 5, No.2/3 pp. 198 - 205, 2010.

[30] M. Yang, J. C.L. Liu, and Y. Tseng, "Editorial," International Journal of Security and Networks, Vol. 5, No.1 pp. 1 - 3, 2010,

[31] S. Malliga and A. Tamilarasi, "A backpressure technique for filtering spoofed traffic at upstream routers," International Journal of Security and Networks, Vol. 5, No.1 pp. 3 - 14, 2010.

[32] S. Huang and S. Shieh, "Authentication and secret search mechanisms for RFID-aware wireless sensor networks," International Journal of Security and Networks, Vol. 5, No.1 pp. 15 - 25 , 2010.

[33] Y. Hsiao and R. Hwang, "An efficient secure data dissemination scheme for grid structure Wireless Sensor Networks," International Journal of Security and Networks, Vol. 5, No.1 pp. 26 - 34 , 2010.

[34] L. Xu, S. Chen, X. Huang, and Y. Mu, "Bloom filter based secure and anonymous DSR protocol in wireless ad hoc networks," International Journal of Security and Networks, Vol. 5, No.1 pp. 35 - 44, 2010.

[35] K. Tsai, C. Hsu, and T. Wu, "Mutual anonymity protocol with integrity protection for mobile peer-to-peer networks," International Journal of Security and Networks, Vol. 5, No.1 pp. 45 - 52, 2010.

[36] M. Yang, "Lightweight authentication protocol for mobile RFID networks," International Journal of Security and Networks, Vol. 5, No.1 pp. 53 - 62, 2010.

[37] J. Wang and G.L. Smith, "A cross-layer authentication design for secure video transportation in wireless sensor network," International Journal of Security and Networks, Vol. 5, No.1 pp. 63 - 76, 2010.

[38] Y. Xiao, "Editorial," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 1 - 1, 2006.

[39] M. Shehab, E. Bertino, and A. Ghafoor, "Workflow authorisation in mediator-free environments," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 2 - 12, 2006.

[40] E. Jung and M. G. Gouda, "Vulnerability analysis of certificate graphs," International Journal of Security and Networks, Vol. 1, No.1/2 pp. 13 - 23, 2006.

[41] A. Kiayias and M. Yung, "Secure scalable group signature with dynamic joins and separable authorities," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 24 - 45, 2006.

[42] M. Franklin, "A survey of key evolving cryptosystems," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 46 - 53, 2006.

[43] I. Hamadeh and G. Kesidis, "A taxonomy of internet traceback," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 54 - 61, 2006.

[44] A. Jhumka, F. Freiling, C. Fetzer, and N. Suri, "An approach to synthesise safe systems," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 62 - 74.

[45] J. B. Evans, W. Wang, and B. J. Ewy, "Wireless networking security: open issues in trust, management, interoperation and measurement," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 84 - 94, 2006.
[46] H. Englund and T. Johansson, "Three ways to mount distinguishing attacks on irregularly clocked stream ciphers," International Journal of Security and Networks, Vol. 1, No.1/2 pp. 95 - 102, 2006.
[47] B. Zhu, S. Jajodia, and M. S. Kankanhalli, "Building trust in peer-to-peer systems: a review," International Journal of Security and Networks, Vol. 1, No.1/2, pp. 103 - 112, 2006.
[48] M. Ramkumar and N. Memon, "Secure collaborations over message boards," International Journal of Security and Networks,Vol. 1, No.1/2, pp. 113 - 124, 2006.
[49] Y. Xiao, X. Jia, B. Sun, and X. Du, "Editorial: security issues on sensor networks," International Journal of Security and Networks, Vol. 1, Nos.3/4, pp.125-126, 2006.
[50] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography-based access control," International Journal of Security and Networks, Vol. 1, No.3/4 pp. 127 - 137, 2006.
[51] J. Zheng, J. Li, M. J. Lee, and M. Anshel, "A lightweight encryption and authentication scheme for wireless sensor networks," International Journal of Security and Networks 2006 - Vol. 1, No.3/4 pp. 138 - 146, 2006.
[52] J. N. Al-Karaki, "Analysis of routing security-energy trade-offs in wireless sensor networks," International Journal of Security and Networks 2006 - Vol. 1, No.3/4 pp. 147 - 157, 2006.
[53] O. Araz and H. Qi, "Load-balanced key establishment methodologies in wireless sensor networks," International Journal of Security and Networks 2006 - Vol. 1, No.3/4 pp. 158 - 166, 2006.
[54] J. Deng, R. Han, and S. Mishra, "Limiting DoS attacks during multihop data delivery in wireless sensor networks," International Journal of Security and Networks 2006 - Vol. 1, No.3/4 pp. 167 - 178, 2006.
[55] J. Hwu, S. Hsu, Y.-B. Lin, and R. Chen, "End-to-end security mechanisms for SMS," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 177 - 183, 2006.
[56] X. Wang, "The loop fallacy and deterministic serialisation in tracing intrusion connections through stepping stones," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 184 - 197, 2006.
[57] Y. Jiang, C. Lin, M. Shi, and X. Shen, "A self-encryption authentication protocol for teleconference services," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 198 - 205, 2006.
[58] S. F. Owens and R. R. Levary, "An adaptive expert system approach for intrusion detection," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 206 - 217, 2006.
[59] Y. Chen, W. Susilo, and Y. Mu, "Convertible identity-based anonymous designated ring signatures," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 218 - 225, 2006.
[60] J. Teo, C. Tan, and J. Ng, "Low-power authenticated group key agreement for heterogeneous wireless networks," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 226 - 236, 2006.
[61] C. Tan, "A new signature scheme without random oracles," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 237 - 242, 2006.
[62] Y. Liu, C. Comaniciu, and H. Man, "Modelling misbehaviour in ad hoc networks: a game theoretic approach for intrusion detection," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 243 - 254, 2006.
[63] V. Karyotis, S. Papavassiliou, M. Grammatikou, and V. Maglaris, "A novel framework for mobile attack strategy modelling and vulnerability analysis in wireless ad hoc networks," International Journal of Security and Networks, Vol. 1, Nos.3/4 pp. 255 - 265, 2006.
[64] H. Chen and M.Guizani, "Editorial," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 1 - 2, 2007.
[65] R. Li, J. Li, and H. Chen, "DKMS: distributed hierarchical access control for multimedia networks," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 3 - 10, 2007.
[66] P. Sakarindr and N. Ansari, "Adaptive trust-based anonymous network," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 11 - 26, 2007.
[67] R. A. Malaney, "Securing Wi-Fi networks with position verification: extended version," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 27 - 36, 2007.
[68] F. Sun and M. A. Shayman, " On pairwise connectivity of wireless multihop networks," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 37 - 49, 2007.
[69] Oz. Erdogan and P. Cao, " Hash-AV: fast virus signature scanning by cache-resident filters," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 50 - 59, 2007.
[70] P. Rabinovich and R. Simon, "Secure message delivery in publish/subscribe networks using overlay multicast," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 60 - 70, 2007.
[71] Z. Chen and C. Ji, " Optimal worm-scanning method using vulnerable-host distributions," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 71 - 80, 2007.
[72] J. Pan, L. Cai, and X. Shen, " Vulnerabilities in distance-indexed IP traceback schemes," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 81 - 94, 2007.
[73] T. Korkmaz, C. Gong, K. Sarac, and S. G. Dykes, "8 Single packet IP traceback in AS-level partial deployment scenario," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 95 - 10, 2007.
[74] H. Ling and T. Znati, "End-to-end pairwise key establishment using node disjoint secure paths in wireless sensor networks," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 109 - 121, 2007.
[75] N. S. Artan and H. J. Chao, "Design and analysis of a multipacket signature detection system," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 122 - 136, 2007.
[76] Y. Zhu, X. Fu, R. Bettati, and W. Zhao, " Analysis of flow-correlation attacks in anonymity network," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 137 - 153, 2007.
[77] Q. Gu, P. Liu, C. Chu, and S. Zhu, "Defence against packet injection in ad hoc networks," International Journal of Security and Networks, Vol. 2, Nos. 1/2, pp. 154 - 169, 2007.
[78] Y. Mu, L. Chen, X. Chen, G. Gong, P. Lee, A. Miyaji, J. Pieprzyk, D. Pointcheval, T. Takagi, J. Traore, J. Seberry, W. Susilo, H. Wang, and F. Zhang, "Editorial," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 171 - 174, 2007.
[79] C. Tartary and H. Wang, "Efficient multicast stream authentication for the fully adversarial network model," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 175 - 191, 2007.
[80] R. Bhaskar, J. Herranz, and F. Laguillaumie, "Aggregate designated verifier signatures and application to secure routing," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 192 - 201, 2007.
[81] H. Hsu, S. Zhu, and A. R. Hurson, "LIP: a lightweight interlayer protocol for preventing packet injection attacks in mobile ad hoc network," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 202 - 215, 2007.
[82] L. B. Oliveira, H. Wong, A. A.F. Loureiro, and R. Dahab, "On the design of secure protocols for hierarchical sensor networks," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 216 - 227, 2007.
[83] H. E. Michail, G. A. Panagiotakopoulos, V. N. Thanasoulis, A. P. Kakarountas, and C. E. Goutis, " Server side hashing core exceeding 3 Gbps of throughput," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 228 - 238, 2007.
[84] K. Hoeper and G. Gong, "Preventing or utilising key escrow in identity-based schemes employed in mobile ad hoc networks," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 239 - 250, 2007.
[85] Z. Cheng and L. Chen, "On security proof of McCullagh–Barreto's key agreement protocol and its variants," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 251 - 259, 2007.
[86] K. M. Finnigin, B. E. Mullins, R. A. Raines, and H. B. Potoczny, "Cryptanalysis of an elliptic curve cryptosystem for wireless sensor networks," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 260 - 271, 2007.
[87] D. Huang, "Pseudonym-based cryptography for anonymous communications in mobile ad hoc networks," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 272 - 283, 2007.
[88] M. Abdalla, E. Bresson, O. Chevassut, B. Moller, and D. Pointcheval, "Strong password-based authentication in TLS using the three-party group Diffie–Hellman protocol," International Journal of Security and Networks, Vol. 2, Nos. 3/4, pp. 284 - 296, 2007.
[89] N. Memon and R. Goel, "Editorial," International Journal of Security and Networks, Vol. 3, No. 2, pp. 79, 2008.
[90] I. Ray and N. Poolsappasit, "Using mobile ad hoc networks to acquire digital evidence from remote autonomous agents," International Journal of Security and Networks, Vol. 3, No. 2, pp. 80 - 94, 2008.
[91] T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa, and S. Shenoi, "Forensic analysis of SCADA systems and networks," International Journal of Security and Networks, Vol. 3, No. 2, pp. 95 - 102, 2008.
[92] E, Cronin, M, Sherr, and M, Blaze, "On the (un)reliability of eavesdropping," International Journal of Security and Networks, Vol. 3, No. 2, pp. 103 - 113, 2008.
[93] J. S. Okolica, G. L. Peterson, and R. F. Mills, "Using PLSI-U to detect insider threats by datamining e-mail," International Journal of Security and Networks, Vol. 3, No. 2, pp. 114 - 121, 2008.