IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 13, Issue 5 (Jul. - Aug. 2013), PP 14-20
www.iosrjournals.org

Cryptographic Cloud Storage with Hadoop Implementation

Sowmya. D, II-MCA,
IFET College of Engineering.

Abstract: *Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, it inevitably poses new security risks toward the correctness of the data in cloud. To address this problem, the proposed design allows users to audit the cloud storage with very lightweight communication and computation cost by utilizing the new cryptographic technique like homomorphic token and distributed erasure-coded data. By introducing the notion of parallel homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data via evaluation algorithms that can be efficiently executed in parallel. In addition, we can hide the function being evaluated with the MapReduce model runnable on Hadoop framework for element testing and keyword search. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks and attempting to strike a balance between security, efficiency and functionality using Kerberos protocol and HDFS system.*

I. Introduction

In the problem of private outsourced computation, a client wishes to delegate the evaluation of a function f on a private input x to an untrusted worker without the latter learning anything about x and f(x). This problem occurs in many applications and, most notably, in the setting of cloud computing, where a provider makes its computational resources available to clients as a service".

One approach to this problem is via the use of homomorphic encryption (HE). An encryption scheme is homomorphic if it supports computation on encrypted data, i.e., in addition to the standard encryption _Work done at Microsoft Research. The decryption algorithms has an evaluation algorithm that takes as input an encryption of some message x and a function f and returns an encryption of f(x). HE schemes can be roughly categorized into two types. The first is arithmetic HE schemes which, in addition to the standard encrypt and decrypt operations, have an add or multiply operation that take as inputs encryptions of messages x1 and x2 and returns encryptions of x1+x2. If an arithmetic HE scheme supports both addition and multiplication, then it can evaluate any arithmetic circuit over encrypted data and we say that it is a fully homomorphic encryption (FHE) scheme. We refer to the second type of HE schemes as non-arithmetic since they do not provide (at least explicitly) addition or multiplication operations.

The problem of outsourced computation occurs in various forms. For instance, in addition to the simple client/worker setting described above, clients often wish to outsource their computation to clusters of workers. This typically occurs when the computation is to be performed over massive datasets, e.g., to analyze large social networks or train machine learning algorithms on large corpora.

1.1 Existing system

In the existing system the cloud computing, the virtual infrastructure has no parallel computing environment. At such scales, computation is beyond the capabilities of any single machine so it is performed on clusters of machines, i.e., large-scale distributed systems often composed of low-cost unreliable commodity hardware. The problems such as data integrity, data corruption may occur. This leads to unreliability of the cloud computing technology.

1.2 Proposed system

In the proposed system, homomorphic encryption with MapReduce algorithm is used when the computation is performed over massive datasets which will not be possible in the existing system.

The Kerberos protocol and HDFS(Hadoop Distributed File System) are used to enhance high security. HDFS is made up of geographically distributed Data Nodes. Access to these Data Nodes is coordinated by a service called the Name Node. Data Nodes communicate over the network in order to rebalance data and ensure data is replicated throughout the cluster.

For my purposes, I will view such a cluster as a system composed of workers and one controller. Given some input, the controller generates n jobs to the workers. Each worker executes its job in parallel and returns some value to the controller who then decides whether to continue the computation or halt.

In this work, we consider the problem of privately outsourcing computation to a cluster of machines.

To address this, I introduce parallel homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. Using a PHE scheme, a client can outsource the evaluation of a function f on some private input x to a cluster of w machines as follows. The client encrypts x and sends the cipher text and f to the controller. Using the cipher text, the controller generates n jobs that it distributes to the workers and, as above, the workers execute their jobs in parallel. When the entire computation is finished, the client receives a cipher text which it decrypts to recover f(x).

Although the cloud infrastructures are much more powerful and reliable than personal computing devices, broad range of both internal and external threats for data integrity still exist. Examples of outages and data loss incidents of noteworthy cloud storage services appear from time to time [5], [6], [7], [8], [9].

On the other hand, since users may not retain a local copy of outsourced data, there exist various incentives for CSP to behave unfaithfully toward the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion [10]. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation [11], [12], [13].

In this paper, we propose an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability of users' data in the cloud. We rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability against Byzantine servers [26], where a storage server may fail in arbitrary ways. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: The identification of the misbehaving server(s). In order to strike a good balance between error resilience and data dynamics, we further explore the algebraic property of our token computation and erasure-coded data, and demonstrate how to efficiently support dynamic operation on data blocks, while maintaining the same level of storage correctness assurance. In order to save the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors (TPA) and be worry-free to use the cloud storage services. The work is among the first few ones in this field to consider distributed data storage security in cloud computing.

The contribution can be summarized as the following three aspects:

1) Compared to many of its predecessors, which only provide binary results about the storage status across the distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s).

2) Unlike most prior works for ensuring remote data integrity, the new scheme further supports secure and efficient dynamic operations on data blocks, including: update, delete, and append.

3) The experiment results demonstrate the proposed scheme is highly efficient. Extensive security analysis shows our scheme is resilient against byzantine failure, malicious data modification attack, and even server colluding attacks.

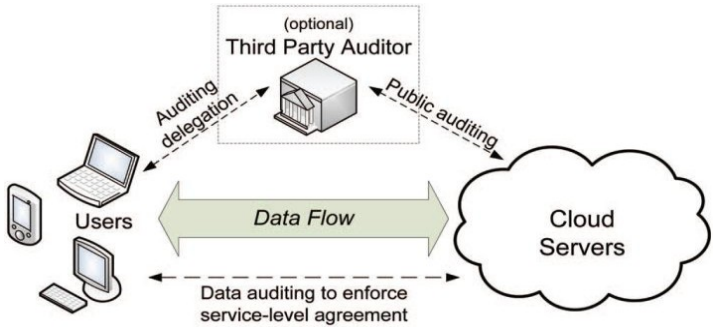


Fig. 1. Cloud storage service architecture.

II. Parallel Homomorphic Encryption (PHE)

2.1 Hadoop framework to cloud-based cluster-computing.

Due to its simplicity and generality, the MapReduce model has quickly become the standard for working with massive datasets. In fact, it is arguably the most successful and widely used model of parallel computation. In 2008 it was reported that Google processed over 20 petabytes of data a day using MapReduce

[24] and that Yahoo! deployed a 10,000 core MapReduce cluster [4]. A MapReduce algorithm is run by an execution framework that handles the details of distributing work among the machines in the cluster, balancing the workload so as to optimize performance and recovering from failures. The most popular framework is Hadoop [3] which is open source and used by hundreds of large organizations including Amazon, Ebay, Facebook, Yahoo!, Twitter and IBM. Building and maintaining large-scale clusters requires a considerable amount of effort and resources, so a recent trend in cluster-computing has been to make use of cloud infrastructures. Examples include Amazon's Elastic MapReduce [1], Cloudera's Hadoop distribution [2] (which can run over several cloud infrastructures) and the recently announced Microsoft Azure Hadoop service. With such services, a client can run a MapReduce algorithm on massive datasets in the cloud". While these services allow clients to take advantage of all the benefits of cloud computing, they require the client to trust the provider with its data. For many potential clients (e.g., hospitals or government agencies) this presents an unacceptable risk. Using an MR-parallel HE scheme a client can maintain the confidentiality of its data while utilizing the processing power of a third-party MapReduce cluster such as Amazon's Elastic MapReduce service. Of course, the client must bear the costs of encryption and decryption which, for massive datasets, can represent a non-trivial amount of work.

2.2 Current level of security in Hadoop.

Current version of hadoop has very basic rudimentary implementation of security which is advisory access control mechanical. Hadoop doesn't strongly authenticate the client, it simply asks the underlying Unix system by executing `whoami` command Any one can communicate directly with a Data node (without the need for communicating with the Namenode) and ask for blocks if you have the block location details.

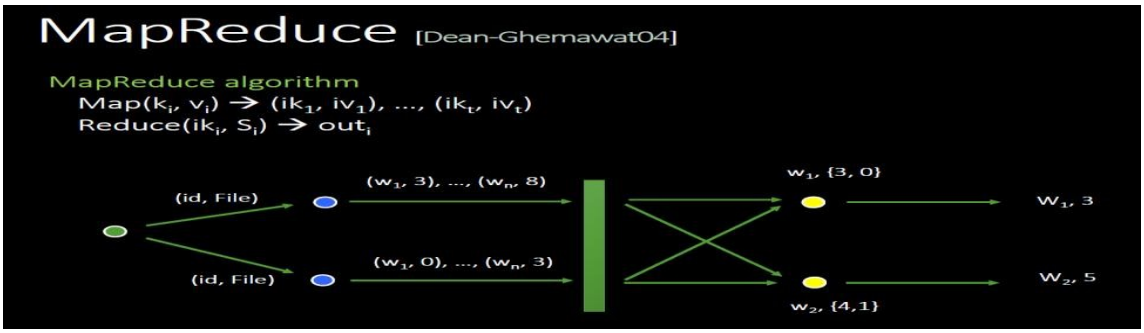
Hadoop cluster may be prone to following attacks unauthorized client scan impersonate authorized users and access the cluster. One can get the blocks directly from the Datanodes by bypassing the Namenode. Eaves dropping / sniffing of data packets being sent by Datanodes to client.

2.3 Homomorphic encryption.

As mentioned in section 1, the problem of private outsourced computation can be addressed non-interactively using HE. Of course, several semantically secure arithmetic HE schemes are known [27, 28, 29], including Gentry's breakthrough FHE scheme [34] and its variants [31,32]. Several non-arithmetic HE schemes are also known including.

2.4 MapReduce.

MapReduce was introduced by Dean and Ghemawat in [33]. In this work, they describe the design and implementation of the MapReduce framework for processing massive datasets on large-scale systems of loosely coupled machines.



MapReduce is a simple interface to design and implement parallel algorithms and an execution framework that handles the details of distributing work among the machines in the cluster, balancing the workload so as to optimize performance and recovering from failures. Due to the simplicity and generality of the MapReduce model, it has quickly become the standard for working with massive datasets

The MapReduce model of computation has been formalized and studied in recent work by Karlo_, Suri and Vassilvitskii [35]. This work introduces a new complexity class that captures the power of the model and relates it to known models of parallel computation, including PRAMs and NC circuits.

Perhaps the simplest MapReduce algorithm is to determine frequency counts, i.e., the number times a keyword occurs in a document collection. The parse algorithm takes the document collection (D1,…,Dn) as input and outputs a set of input pairs $(I,D_i)_i$. Each mapper receives an input pair (i;Di) and outputs a set of intermediate pairs (word count for each document) wj found in Di. All the intermediate pairs are then partitioned by the partition operation into sets.

For example, consider Pl consists of all the intermediate pairs with label wl. The reducers receive a set Pl of intermediate pairs and sum the values of each pair. The result is a count of the number of times the word wl occurs in the document collection. Then all the intermediate pairs are concatenated to reveal all these counts and returns the result.

III. Security Implementation

How does Kerberos work?

A network authentication protocol is used as private-key cryptography for providing authentication across open the cloud computing environment. It mediates authentication through a trusted 3rd party. It provides authentication for the confirmation that a user who is requesting services is a valid user of the network services requested. The next is Authorization, the granting of specific types of service to a user, based on their authentication, what services they are requesting, and the current system state. At last accounting, the tracking of the consumption of network resources by users.

3.1 Capabilities

To read data from the HDFS the client has to obtain block locations and capabilities from Namenode before it goes to Datanodes.

- C->N:read(path),TS,hash<read(path),TS,KUCN>
- N->C:block_locations, hash<block_locations>
- The capabilities are embedded into block location information and signed by the Namenode. The Datanode verifies the capabilities and accordingly allows to read or doesn't.
- C->D:read(block)
- Description of Capability information embedded into the block location information. The sign (With Namenode's private key) of the capability and block id is also embedded.
- C=ID, permissions, path.
- Sign=<c, block_id>KRN.

3.2 Revocation of capabilities

Capabilities can potentially used by clients to read the data from HDFS at any time after when they were issued. However the file permissions change over period of time. Revocation of capabilities needs to be done, in order to prevent replay attacks. Capabilities issued by Namenode will have an expiry period (say1hr) and this can be configured in hadoop-site.xml. The client has to get a renewal ticket issued by the Namenode and has to present it to the Datanode for every request after expiry of the capabilities.If the renewal ticket is not presented, the Datanode will deny the request. Revocation of capabilities is done actively by Namenode.This is done by sending the message to the Datanodes to deny the particular capabilities.

IV. Ensuring Cloud Data Storage

In cloud data storage system, users store their data in the cloud and no longer possess the data locally. Thus, the correctness and availability of the data files being stored on the distributed cloud servers must be guaranteed. One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures.

In order to achieve assurance of data storage correctness and data error localization simultaneously, our schemeentirely relies on the precomputed verification tokens. The main idea is as follows: before file distribution the user precomputes a certain number of short verification tokens on individual vector $G^{(j)}$ $(j \in \{1, \ldots, n\})$.

Each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge ,each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens precomputed by the user. Meanwhile, as all servers operate over the same subset of the indices, the requested response values for integrity check must also be a valid codeword determined by the secret matrix P.

Algorithm 1. Token Precomputation.

procedure
  Choose parameters l, n and function f, ϕ;
  Choose the number t of tokens;
  Choose the number r of indices per verification;
  Generate master key K_{PRP} and challenge key k_{cha}
  for vector $G^{(j)}, j \leftarrow 1, n$ **do**
    for round $i \leftarrow 1, t$ **do**
      Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
      Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
    end for
  end for
  Store all the v_i's locally.
end procedure

Once all tokens are computed, the final step before t
distribution is to blind each parity block $g_i^{(j)}$ in $(G^{(m+1)}, .$
$G^{(n)})$ by

$$g_i^{(j)} \leftarrow g_i^{(j)} + f_{k_j}(s_{ij}), i \in \{1, \ldots, l\},$$

where k_j is the secret key for parity vector $G^{(j)}(j \in \{m+1, \ldots, n\})$. After blinding the parity information, the user disperses all the n encoded vectors $G^{(j)}\ (j \in \{1, \ldots, n\})$ across the cloud servers S1, S2, . . . , Sn.

4.1 Correctness Verification and Error Localization

Error localization is a key prerequisite for eliminating errors in storage systems. It is also of critical importance to identify potential threats from external attacks. However, many previous schemes [23], [24] do not explicitly consider the problem of data error localization, thus only providing binary results for the storage verification. Our scheme outperforms those by integrating the correctness verification and error localization (misbehaving server identification).

In our challenge-response protocol:

The response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s).

Algorithm 2. Correctness Verification and Error Localization.

procedure CHALLENGE(i)
  Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
  Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
  Receive from servers:
  $\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
  for $(j \leftarrow m+1, n)$ **do**
    $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q$, $I_q = \phi_{k_{prp}^{(i)}}(q)$
  end for
  if $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **than**
    Accept and ready for the next challenge.
  else
    for $(j \leftarrow 1, n)$ **do**
      if $(R_i^{(j)} != v_i^{(j)})$ **than**
        return server j is misbehaving.
      end if
    end for
  end if
end procedure

V. Conclusion

The overhead with the introduction of capabilities is low but secures the data access for only clients which have been issued the capabilities by Namenode. Although the performance overhead at the Datanode isn't significant for 64MB or larger blocksize, it can be reduced further by caching the capabilities for each block.

Moreover, this paper relies on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasurecoded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s).I also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third party auditors and be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

References

[1] Amazon elastic mapreduce. See http://aws.amazon.com/elasticmapreduce.
[2] Cloudera. See http://cloudera.com.
[3] Powered by hadoop. See http://wiki.apache.org/hadoop/PoweredBy.
[4] Yahoo! launches world's largest hadoop production application. See http://developer.yahoo. net/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html, 2008.
[5] M. Abadi, J. Feingenbaum, and J. Killian. On hiding information from an oracle. In ACM Symposium on Theory of Computing (STOC 1987), pages 195{203, 1987.
[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. Journal of Computational Complexity, 15(2), 2006.
[7] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC0. SIAM Journal of Computation, 36(4):845{888, December 2006.
[8] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography with constant input locality. In Advances in Cryptology - CRYPTO '07, Lecture Notes in Computer Science, pages 92{110. Springer-Verlag, 2007. 19 (CCS '07), pp. 584-597, Oct. 2007.
[9] B. Krebs, "Payment Processor Breach May Be Largest Ever," http://voices.washingtonpost.com/securityfix/2009/ 01/ payment_processor_breach_may_b.html, Jan. 2009.
[10] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.
[11] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, Oct. 2007.
[12] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.
[13] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-PreservingAudit and Extraction of Digital Contents," Cryptology ePrintArchive, Report 2008/186, http://eprint.iacr.org, 2008.
[14] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf.Security and Privacy in Comm. Netowrks (SecureComm '08), pp. 1-10,2008.
[15] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.
[16] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Sixth Theory of Cryptography Conf.(TCC '09), Mar. 2009.
[17] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22,no. 5, pp. 847-859, 2011.
[18] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, preprint, 2012, doi:10.1109/TC.2011.245.
[19] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 187-198, 2009.
[20] T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06), pp. 12-12, 2006.
[21] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," Proc. USENIX Ann. Technical Conf. (General Track), pp. 29-41, 2003.
[22] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," ACM Trans. Computer Systems, vol. 20, no. 4, pp. 398-461, 2002.
[23] L. Carter and M. Wegman, "Universal Hash Functions," J.Computer and System Sciences, vol. 18, no. 2, pp. 143-154, 1979.
[24] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure-Coded Data," Proc. 26th ACM Symp. Principles of Distributed Computing, pp. 139-146, 2007.
[25] J.S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial on Reed-Solomon Coding," Technical Report CS-03-504, Univ. of Tennessee, Apr. 2003.
[26] Hadoop Security Design. See http://media.blackhat.com/bh-us-10/whitepapers/Becherer/BlackHat-USA-2010-Becherer-Andrew-Hadoop-Security-wp.pdf
[27] S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28(2):270{299, April 1984.
[28] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in Cryptology – CRYPTO 1984, Lecture Notes in Computer Science, pages 10{18. Springer, 1985.
[29] J. Benaloh. Veri_able secret-ballot elections. PhD thesis, Yale University, 1987.

[30] N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In International Conference on Practice and Theory in Public Key Cryptography (PKC '10), volume 6056 of Lecture Notes in Computer Science, pages 420{443. Springer, 2010.
[31] L. Valiant. A bridging model for parallel computation. Communications of the ACM, 33(8):103-111,1990.
[32] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Advances in Cryptology - EUROCRYPT 2010, volume 6110 of Lecture Notes in Computer Science, pages 24{43. Springer, 2010.
[33] J. Dean and S. Ghemawat. Mapreduce: simpli_ed data processing on large clusters. In Symposium on Opearting Systems Design & Implementation, pages 10{10, Berkeley, CA, USA, 2004. USENIX Association.
[34] C. Gentry. Fully homomorphic encryption using ideal lattices. In ACM Symposium on Theory of Computing (STOC '09), pages 169{178. ACM Press, 2009.
[35] H. Karlo_, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In Symposium on Discrete Algorithms (SODA '10), pages 938{948. SIAM, 2010.