using Splitting and merging algorithm. Meanwhile a Meta table is created for currently uploading file and given to the client.

When client has to check data integrity for a particular file he/she should give the Meta data and private key for the file to a verifier (Third party auditor). The verifier distributes the challenge query to the corresponding cloud servers according to the storage metadata. The cloud servers respond the challenge. Based on the response the verifier checks whether the files are modified or not.

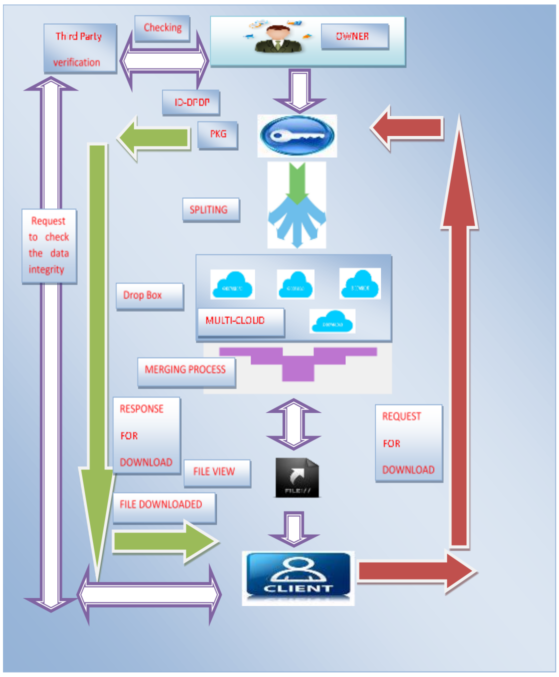

Figure 1: Architecture diagram (overall representation of)

C. Security Prospects By Multi-Cloud Architectures

The basic underlying idea is to use multiple distinct clouds at the same time to mitigate the risks of malicious data manipulation, disclosure, and process tampering. By integrating distinct clouds, the trust assumption can be lowered to an assumption of non-collaborating cloud service providers. Further, this setting makes it much harder for an external attacker to retrieve or tamper hosted data or applications of a specific cloud user. The idea of making use of multiple clouds has been proposed by Bernstein and Celesti. However, this previous work did not focus on security. Since then, other approaches considering the security effects have been proposed. These approaches are operating on different cloud service levels, are partly combined with cryptographic methods, and targeting different usage scenarios.

D. Security And Privacy-Enhancing Multicloud Architectures

In this paper, we introduce a model of different architectural patterns for distributing resources to multiple cloud providers. This model is used to discuss the security benefits and also to classify existing approaches.
In our model, we provide security in cloud by performing following actions:

- Partitioning the uploaded file (data) into fragments allows distributing fine-grained fragments of the data to distinct clouds. None of the involved cloud providers gains access to all the data, which safeguards the data's confidentiality.
- Encrypting each fragment using AES Encryption algorithm, which safeguards the data's integrity.
- Allowing only authorised person to download the complete data

Each of the above provides individual security merits, which map to different scenarios and their security needs. Obviously, the actions can be combined resulting in combined security merits, but also in higher deployment and runtime effort.

Partitioning the Uploaded File

This multi-cloud architecture specifies that the application data is partitioned and distributed to distinct clouds. The most common forms of data storage are files. Files typically contain unstructured data (e.g., pictures, text documents) and do not allow for easily splitting or exchanging parts of the data. To split this kind of data splitting and merging algorithm can be used.

File Splitter is a program which does not require installation and can be used to split files to multiple chunks as well as to merge multiple chunks into a single file. File Splitter is software which is used to split the user specifying file according to the user specifying size. The split portions of file may carry some temporary information to denote the number of split part and total number of parts etc. This idea is used to split big files to small pieces for transferring purpose, uploading etc. In the destination side, these parts of file can be jointed to form the original source file. Splitting process is mainly aiming in the area of file transferring from one end to another. There are three techniques to split the file.

i. Split in the size of 1.4MB (Floppy size)
ii. Split in the size of 650MB (CD size)
iii. User specifying size, here file will be split equally in the specified size.

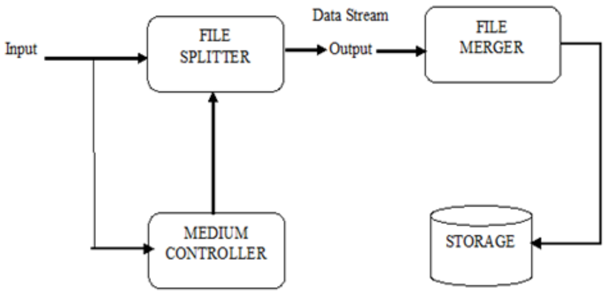

Figure 2: File Splitter and Merger model

Approach Used

File Splitter tool is very useful tool to split the file into chunks and we can use the same tool to join the already split files. A File Splitter is software that splits individual streams of a file, e.g., audio, video, or subtitles and sends them to their respective decoders for actual decoding. Usually a suffix is appended to the filename with a number, and at time of recombination, the files need to be combined in the order indicated by the suffix. So if you have largefile.zip, it might split it into largefile.zip.001, largefile.zip.002, largefile.zip.003, largefile.zip.004. That information needs to be preserved. You can change the prefix, but you must do it consistently. Sometimes splitting does follow a specific format and changes the underlying format a bit. This is common with the built-in splitting mechanisms of archive formats.

Algorithm

STEP 1. If file is to be split go to step 2 else merge the fragments of the file and goto step 10.
STEP 2. Input srcpath, destnpath, sof
STEP 3. Size = size of source file
STEP 4. If size>sof goto step 6 else print file cannot be split and goto step 10
STEP 5. Split into fragment=sof
STEP 6. Size = size-sof
STEP 7. If size>sof goto step 6
STEP 8. We get fragments with merge option
STEP 9. End

Encryption using AES Algorithm

Advanced Encryption Standard is designed by Rijmen-Daemen in Belgium. It has 128/192/256 bit keys, 128 bit data. It is an iterative rather than Feistel cipher. It processes data as block of 4 columns of 4 bytes. It operates on entire data block in every round. It is designed to have:

- Resistance against known attacks
- Speed and code compactness on many CPUs
- Design simplicity

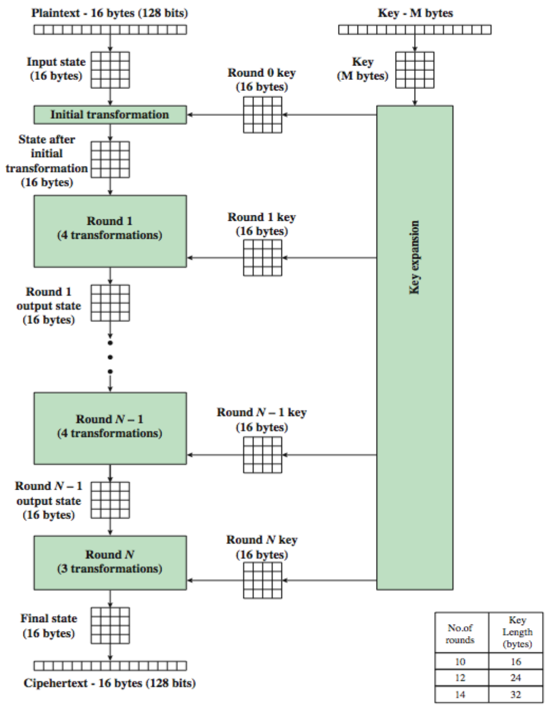

Figure 3: AES Encryption Process

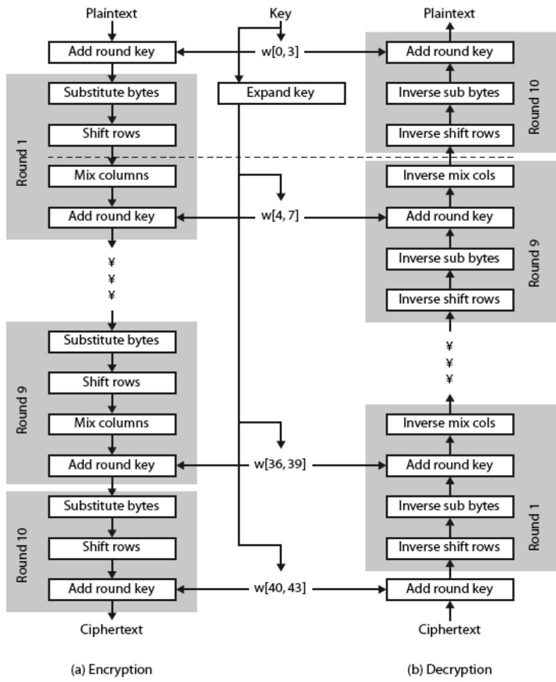

Figure 4: AES Structure

Authentication

Only authorised person can download the complete data from the cloud. This is made sure by performing following procedures

- While registering each and every client will be given a private key which is generated based in ID-DPDP protocol
- While uploading a file client has to provide a unique 8-bit key as public key.
- If a client wishes to download a file he/she has to login and provide both the private key and public key.

If and only if all these are matched then the file will be downloaded otherwise file will not get downloaded.

E. System Model And Security Model of ID-DPDP

The ID-DPDP system model and security definition are presented in this section. An ID-DPDP protocol comprises four different entities which are illustrated in Figure 4. We describe them below:

1) Client: An entity, which has massive data to be stored on the multi-cloud for maintenance and computation, can be either individual consumer or corporation.
2) CS (Cloud Server): An entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.
3) Combiner: An entity, which receives the storage request and distributes the block-tag pairs to the corresponding cloud servers. When receiving the challenge, it splits the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier.
4) PKG (Private Key Generator): an entity, when receiving the identity, it outputs the corresponding private key. First, we give the definition of interactive proof system. It will be used in the definition of ID-DPDP.

III. RESULTS AND DISCUSSION

Proposed ID-DPDP Protocol

An ID-DPDP protocol is a collection of three algorithms (Setup, Extract, TagGen) and an interactive proof system (Proof). They are described in detail below.

1) Setup: Input the security parameter k, it outputs the system public parameters params, the master public key mpk and the master secret key msk.
2) Extract: Input the public parameters params, the master public key mpk, the master secret key msk, and the identity ID of a client, it outputs the private key skID that corresponds to the client with the identity ID.

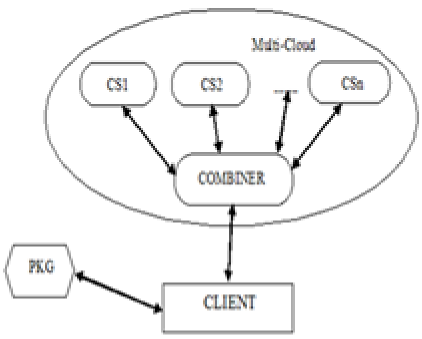

Figure 5: The System Model of ID-DPDP

3) TagGen: Split the whole file F into n blocks, i.e., F = (F1, F2, ···, Fn). The client prepares to store the block Fi in the cloud server CSi. Input the private key skID, it outputs the tuple {φi, (Fi)}, where φi denotes the i-th record of metadata, (Fi) denotes the i-th block. Denote all the metadata {φi} as φ.
4) Proof (P, C (Combiner), V (Verifier)): is a protocol among P, C and V. If the client delegates the verification task to some verifier, it sends the metadata table to the verifier. Of course, the verifier may be the third auditor or the client's proxy. At the end of the interactive protocol, V outputs a bit {0|1} denoting false or true. Besides of the high efficiency based on the communication and computation overheads, a practical ID-DPDP protocol must satisfy the following security requirements:
   - The verifier can perform the ID-DPDP protocol without the local copy of the file(s) to be checked.
   - If some challenged block are modified or lost it will be notified to the clients.

IV. CONCLUSION

In multi-cloud storage, this paper formalizes the ID-DPDP system model and security model. Besides of the elimination of certificate management, our ID-DPDP protocol has also flexibility and high efficiency. At the same time, the proposed ID-DPDP protocol can realize private verification, delegated verification and public verification based on the client's authorization.

V. REFERENCES

[1] Gartner, "Gartner Says Cloud Adoption in Europe Will Trail U.S. by at Least Two Years," http://www.gartner.com/it/page.jsp?id=2032215, May 2012.

[2] J.D.J. Wisner, G.K.G. Leong, and K.-C. Tan, Principles of Supply Chain Management: A Balanced Approach. South-Western, 2011.

[3] Jens-Matthias Bohli, Nils Gruschka, Meiko Jensen, Member, Ieee, Luigi Lo Iacono, And Ninja Marnau At Ieee Transactions On Dependable And Secure Computing, Vol. 10, No. 4, July/August 2013

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, "Provable Data Possession at Untrusted Stores", CCS'07, pp. 598-609, 2007.

[5] G. Ateniese, R. DiPietro, L. V. Mancini, G. Tsudik, "Scalable and Efficient Provable Data Possession", SecureComm 2008, 2008.

[6] C. C. Erway, A. Kupcu, C. Papamanthou, R. Tamassia, "Dynamic Provable Data Possession", CCS'09, pp. 213-222, 2009.

[7] F. Sebe´, J. Domingo-Ferrer, A. Mart´ınez-Balleste´, Y. Deswarte, J. Quisquater, "Efficient Remote Data Integrity checking in Critical Information Infrastructures", IEEE Transactions on Knowledge and Data Engineering, 20(8), pp. 1-6, 2008.

[8] H.Q. Wang, "Proxy Provable Data Possession in Public Clouds," IEEE Transactions on Services Computing,2012.http://doi.ieeecomputersociety.org/10.1109/TSC.2012.35

[9] Y. Zhu, H. Hu, G.J. Ahn, M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", IEEE Transactions on Parallel and Distributed Systems, 23(12), pp. 2231-2244, 2012.

[10] A. F. Barsoum, M. A. Hasan, "Provable Possession and Replication of Data over Cloud Servers", CACR, University of Waterloo, Report2010/32, 2010. Available at http://www.cacr.math.uwaterloo.ca/techreports /2010/cacr2010-32.pdf.

[11] An Approach for File Splitting and Merging by Shristi Sharma, Shreya Jaiswal, Priyanka Sharma, Prof. Deepshikha Patel, Prof. Sweta Gupta