ZERO DATA REMNANCE PROOF IN CLOUD STORAGE

Mithun Paul and Ashutosh Saxena[1]

[1] Security and Privacy Group, SETLabs
Infosys Technologies Ltd, Hyderabad
(Mithun_Paul, Ashutosh_Saxena01)@infosys.com

ABSTRACT

In a cloud environment where Storage is offered as a service, a client stores his data with a provider and pays as per the usage. Once the contract ends, the client, as the data owner, may like to see, due to privacy reasons and otherwise that the data is properly shredded in the provider storage. In this paper we propose a scheme for Zero Data Remnance Proof (ZDRP) – a comprehensive proof given by the cloud data storage provider as regards to zero data remnance post the SLA period. In absence of such shredding the provider can consume the data to his benefit without coming in legal framework. The proof of data destruction can be achieved together by clauses in the SLA and a comprehensive destruction-verifier algorithm. The implementation of this can be achieved by appropriate modification of the data updation mechanisms provided by open source cloud providers.

KEYWORDS

Data Shredding, Cloud, Privacy

1. INTRODUCTION

Clouds are a large pool of easily accessible and usable virtualized resources (such as development platforms, hardware, and/or services). Such resources can be re- configured dynamically to adjust a variable load (scale), allowing also for a resource utilization which is optimum. This is typically used in a pay-per-use model where customized SLAs by the Infrastructure Provider are offered as guarantees [1].Cloud is fast being adapted as the de-facto standard by many industries and academic organizations.

The core idea in cloud storage is that a user who stores his data in a cloud can recover them irrespective of geographical and time barriers. In this case, the user's data will be kept in the infrastructure provided by a third party where he has virtually no access once the data is stored and where the SLA provided by the cloud storage facility provider is the only safety assurance he has.

For small or mid-sized businesses, which generally lack the capability for a massive initial investment necessary for an elaborate infrastructure set up, Storage as a Service (SaaS) is considered a boon. Also when it comes to long-term storage of records, disaster recovery for business continuity and availability post geographic calamities etc, SaaS is considered as a very credible option [2]. An obvious concern of data privacy forebodes such a setup inevitably which generally can be overcome by various methodologies like mandatorily storing the data in an encrypted form. But this might not suffice for a post SLA scenario, where the remnants of the stored data will be at the mercy of the Storage Provider, albeit in an encrypted form.

Let's consider the following example: A User/Corporate 'U' wants to store sensitive data in the cloud with a storage provider/vendor 'P' for a particular period of time. Now at the end of that particular time period, 'U' wants to withdraw the data and dissociate with 'P'. The provider P

must in exemplary congruence with the SLA, return the data to 'U' or delete it as U's requirement be. Now the difficulty here is how the provider can claim and give a tangible proof to client that the data has completely left his storage area.

The Provider 'P' will have to ensure that the data stored by the client is comprehensively destructed from the storage, especially sensitive data that may be under regulatory compliance requirements. But even after assurances from the provider that the data has been completely rendered useless or erased from their storage media, there is no methodology currently to ratify it. It will create unfathomed issues if U's data storage provider's (P) data infrastructure destruction requirements isn't compatible with U's destruction requirements (e.g., the provider erases the data from the disks but inadvertently forgets to do so from storage archives) and some form of data remnance succumbs at last to data leakages. One example of such exploitation of vulnerability is when the allegedly destroyed data is recovered from magnetic media or the random-access memory.

This paper involves development of a prototype model for data shredding in distributed environment. The stress is on a destructor or a probing engine which will probe the environment and as per the the rules on the data store, will shred them partially or fully. This rendering itself can further be extended as a service on Cloud.

2. RELATED WORK

This paper is an extension of our previous work on Proof of Erasability [3]. The work done on 'Proof of Retrievability', from which the name 'Proof of Erasability' is inspired, by Jules and Kaliski [6] Shacham and Waters [7] has been also referred for arriving at this. Two methods of comprehensive data shredding have been consulted for this work, viz Kishi's methodology by which data can be shredded comprehensively within a storage subsystem and the Gutmann algorithm which provides a similar methodology for secure deletion of data [4] ,[5].

3. OUR SCHEME

We are proposing a methodology/protocol which will tweak the storage infrastructure in such a manner that it necessitates the capability of making a hard erasure with proof at the end of a storage requirement. Our main interest in Proof Of Zero Data Remnance Scheme is related to the destructor based approach mentioned in the introduction. Before giving details, we outline the general protocol structure.

3.1. Setup Phase

Assume User U who has a huge quantity of sensitive data with him doesn't have the necessary storage facilities or the infrastructure for procuring it. As a solution to this U approaches one of the vendors, V, in the cloud who provides U the required storage space for a pre determined amount of time at cheap rates. Examples of such vendors include Amazon's EC2 [8], or any of the other popular vendors who offer Storage as a Service [9], [10].

3.2. SLA Details

The user U or his company would sign a service level agreement (SLA) with the Vendor V whereby the SaaS provider agrees to rent storage space on a cost-per-gigabyte-stored and cost-per-data-transfer basis and the company's data would be automatically transferred at the specified time over the storage provider's proprietary wide area network (WAN) or the Internet [11]. Under the same preamble, the user may or may not be periodically accessing/updating the data through the multitude of application interfaces that he uses. E.g.: web services. Though it is out of the scope of this discussion, it is understood that the SLA must be scrutinized for regulatory checks like back up services, encryption and compression algorithms involved for data protection during

the storage time [2]. In preamble to our current invention at this stage we are more concerned about the node and IP level details and the number of simultaneous storages/Nodes in the cluster that will be used for storing the client's data as per the SLA. The user must ensure that in the SLA the provider gives the details of the servers, starting from respective geographic locations delving into minute details like the I.P Addresses, nodes and memory blocks, on which their data will be simultaneously distributed

Generally in a cloud certain number of copies of the same data is created, regularly updated and stored at various geographical locations to ensure that a mishap in one of the geographical location doesn't affect the continuous data access process any of the client's applications might be doing.

Also a check must be done on the exposure window time mentioned in the SLA. Other Typically need to have clauses in the SLA include details about Archival frequency, Archival locations and the Exposure window , Δ T in which a change made to one node is updated in the rest of the nodes in the cluster etc.

3.3. Exposure Window

An exposure window is the time that is needed to complete an updation cycle across various copies of the data stored in various nodes in the clusters. For example : the user application which is accessing the data through an interface, say a web service, needs to update one row in one table in the database. So the application sends an update command, SQL or otherwise, to one of the databases to which it is connected. Suppose the cloud vendor is simultaneously storing the client's data in 4 different servers, viz D1, D2, D3 and D4. Assume that at the give time the data being shown to the user application is being retrieved from D1. Now the Exposure Window (Δt) is the time, as per the SLA, that is required for this data to be updated in all the other 3 databases, viz D2, D3, and D4 across the world i.e. the time during which the data is exposed/not updated. Similarly, post SLA, when the provider does a data erasure, an Exposure Window is the time required for the erasure done on one node to be reflected in all the respective copies.

3.4. End of SLA period

Suppose that at the end of a particular time period the user U is done with the necessity of storing the data in the cloud. Or maybe the user has found a better vendor who is providing the storage facility in a much cheaper rate. At this stage the user wants to pull out and ensure that the data is no longer with the vendor. Or even if the data remains at the already existing cloud storage vendor, even if it is in the encrypted state, it is rendered useless so that no privacy leakage occurs even if the data goes into wrong hands.

3.5. Destructor

So at this stage the Provider U needs to make use of a DESTROYER application. At the end of SLA period, provider stores all the modified bits and its corresponding address values as key-value pairs before the destructor program run.EG: [OFFBC, 0].

3.5.1. Probable Methodologies for Data Destruction

3.5.1.1. Systematically modifying/updating the Most Significant Bit of every data chunk, Bit flipping

Given a data set this application takes the Most Significant Bit of all the bytes, and replaces it with the opposite Boolean value; i.e. a zero will be replaced with one and one will be replaced with zero. This essentially will render the byte useless. Fig1 and Fig 2 show how the Most Significant Bits have been manipulated before and after the destruction algorithm works on it.

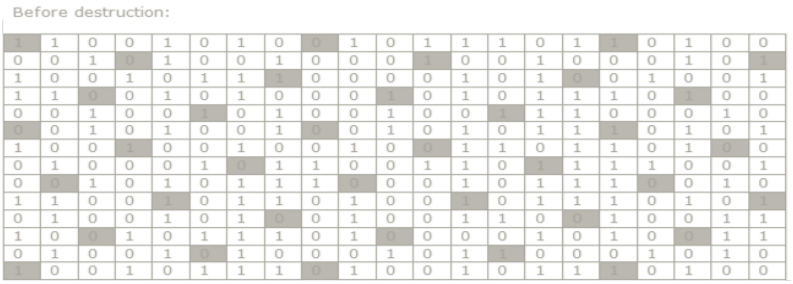

Figure 1. How the typical bits look before destruction

Fig. 1. How the typical bits look before destruction

During this destru static storage media before every destruction operation. This will include a log of the current values of the bit being modified, the relative address of the bit and the time during which the modification is happening. This destroyer application is invoked on the database to which the user application is currently connected to, viz D1 until each and every tuple in the data is systematically and comprehensively rendered useless.

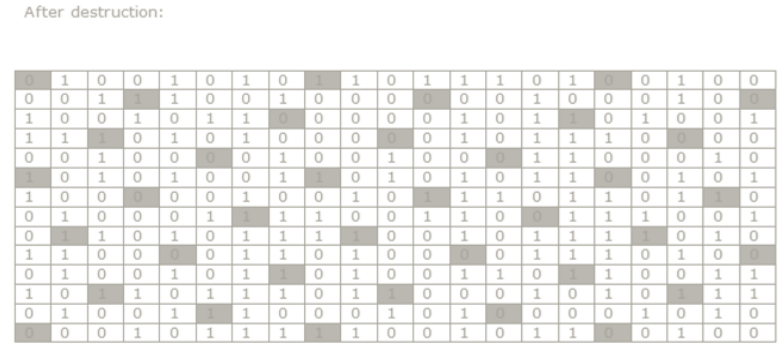

Figure 2. How the typical bits look after destruction

A typical destructor algorithm can be represented by a flowchart as below:

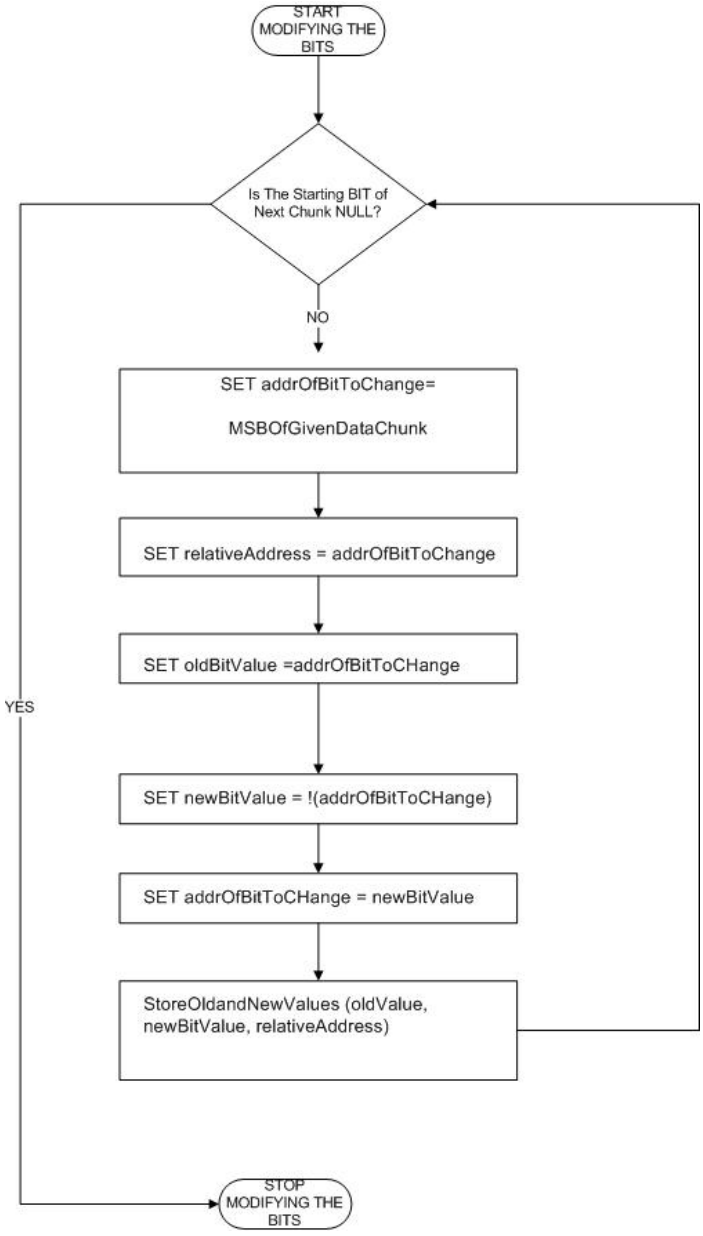

Figure 3. Flow Chart for a typical destuctor algorithm

3.5.1.2. Assumptions:

a) Assuming the data is not being stored contiguously and hence there is a necessity for an algorithm which would respawn the pointer to the address where the next sequential data resides.

b) While modifying a bit value, store the bit values, previous and new, the current time and also the relative address at which the change was made, in a stable storage like database, so that this can be used for data verification process.

3.5.1.3. Other data erasure techniques

Other data erasure/destruction techniques that can be used to achieve the same end can be Replacement with Zeroes, Data masking algorithms etc [12].

3.6. Verifier

Once the destruction cycle is completed it is necessary that the provider P does a verification, preferably in presence of the user U, to ensure that the bits have been modified/data has been destructed in the other copies of the database, viz the database servers provided by the cloud data storage provider, where all the data is copied, on which any data updation made on the first database, D1, will be updated after the exposure time mentioned in the database. For this purpose the verifier application will next connect to the other databases, D2, D3 etc. Here it will pick the bits of data from the same address location as that was modified in D1. A comparison of this bit value in the secondary databases with the equivalent bit value in D1, on which the destructor application had worked is done next.

A typical verifier algorithm can be represented by a flowchart as below.

3.5.1. Assumptions:

a) Assuming all the modified bits and its corresponding address values were stored as key-value pairs during the destructor program run.EG: [OFFBC, 0].

b) DeltaT, is the exposure time mentioned in SLA which is the time taken by the provider to update other copies of the update that is made in one database.

c) This is the address of the modified bit stored during the destructor process.

d) For getting right results the verification/comparison is done only after the exposure time elapses.

e) We are comparing the value in the auxiliary/copies of the database.

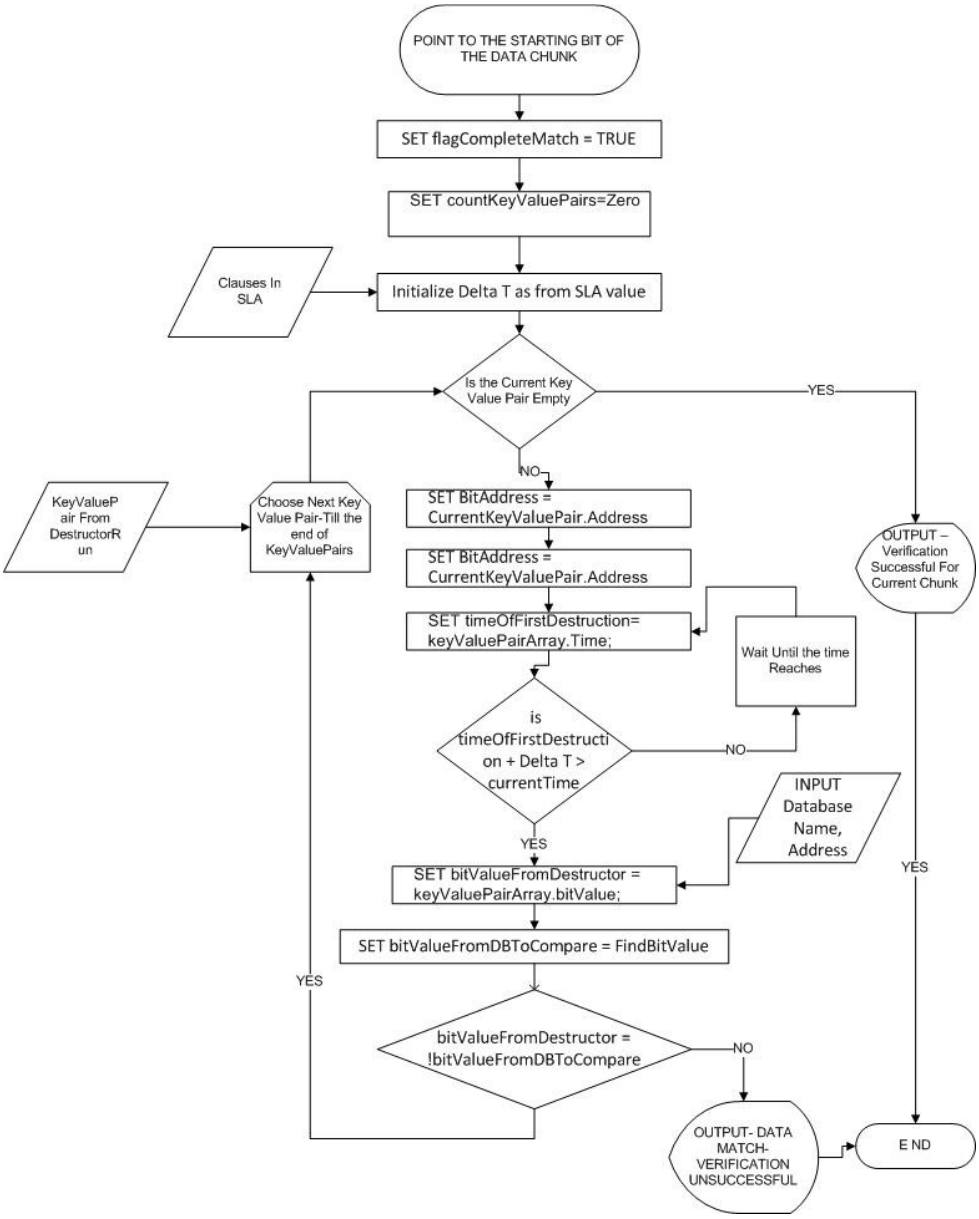

Figure 4. Flow Chart for a typical verifier algorithm

4. PROOF OF ZERO DATA REMNANCE

The Provider now needs to ensure that a similar data updation, which inherently destructs the data, has happened to the rest of the other database copies, viz D2, D3 and D4. As mentioned earlier, as per the SLA any updation done to D1 is reflected in D2, D3 and D4 with a time difference of (Δt).

For this verification the cloud storage provider gets relative bit location of certain bits in the D2, D3 and D4. If we are using the MSB based methodology, these will be the relative MSBs noted by the destroyer application before rendering the database D1 useless. Once the bits are retrieved

the Provider runs those bits through the hash algorithm chosen before the destruction run, with the aforementioned key and the output is compared with the earlier noted values.

This output is compared with the value recovered from the process one on database D1. If both these values match, we can ensure that the data with the provider is now rendered useless.

As per the SLA provided by 'P' this updation will be reflected in the rest of the copies of the database within the given exposure window. At the end of this exposure window, 'P' can retrieve certain bits of the values from one of these secondary databases, and compare them with the equivalent hashed value which was generated during the destructor run. UN SUCCESSFUL A match of these two would mean that all the data stored with the provider has thus been rendered useless. This can be called a Proof Of Zero Data Remnance, viz a tangible proof given by the 'P' to 'U' that the data that is with 'P' is no more useful to anyone and hence proven destructed.

5. ANALYSIS OF OUR SCHEME

As mentioned, the scenario is that the cloud provider is providing a storage space, along with certain permissions for the applications from the client end to access and modify the data. It is on extension of the same postulation that a destructor algorithm is being proposed. At the end of SLA, as the cloud provider is accessing the data through legitimate and well defined means, albeit for destructive purposes, the proposition can be considered meticulous and proper.

Considering security aspects, there are some possible vulnerabilities that can be raised. One is that during the exposure time, viz the time it takes to make the updation to one node reflected in various other nodes/copies in the same cluster, the data is vulnerable. A probing, comparison and verification of pre and post data during this time will yield unwarranted results thus leading to the nullification of the zero data remnance proof.

Also it can be argued that the provider can anytime make an extra copy of the data without informing the client. A legally binding Service Level Agreement will be the best solution for this. Within this SLA the provider will be mentioning the various archival methodologies he will be using to store client's data and also the copies and locations of the data which he will be creating to prevent local mishaps. A breach of such legal agreements will possibly be contested on the courts and any provider would want to avoid such a scenario and the loss of credibility that might arise along with.

Performance wise, it must be mentioned that there is a scope of improvement with the given pseudo code. Optimizations can be incorporated with regard to the destructor and verifier processes.

6. BUSINESS SCENARIOS

The Zero Data Remnance Proof can be provided as an addendum by the cloud data provider along with his storage services. This will give him a business edge over his competitors when other providers will be providing data privacy during storage period using various techniques like encryption or data masking, he will be the only provider who offers privacy of the data erasure even after SLA is completed. Alternately this can be interlaced/masqueraded as the regular update calls that the client application is permitted to make towards its data This also can be offered as an independent Service in itself which can be made use of either by the Client to ensure privacy of his data or by any cloud data provider for ensuring his capacity as a safe storage provider. PZDR does not by itself, however, protect against remnance of file contents. The robustness must be enhanced by proper Service Level Agreement (SLA) clauses, like legal disclosure of file storage redundancy/archival details.

7. CONCLUSION

We have presented here a scheme for Proof Of Zero Data Remnance (PZDR) in a distributed environment. The focus of our work is on a probing engine/destructor which will probe the environment and based on the rules on the data store, will shred them partially or fully. This prototype can further be extended as a service on Cloud. This destructor will systematically modify/update the most significant bit of every data chunk, thus ensuring that the data is so corrupted/ impaired that no part of the data will make any sense to whomever gets hold of it later. On analyzing the scheme we found that our proposition can be considered meticulous and proper as we are accessing the data through legitimate and well defined means, albeit for destructive purposes. Our scheme enjoys full faith of the provider as it provides him with the shredding capabilities. With the given pseudo code there is a scope of improvement and further optimizations can be incorporated with regard to the destructor and verifier processes. Implementation of our scheme is underway and we expect a good performance of the protocol.

REFERENCES

[1] Vaquero, Luis M., Rodero-Merino ,Luis., Caceres ,Juan., & Lindner, Maik (January 2009) "A break in the clouds: towards a cloud definition", ACM SIGCOMM Computer Communication Review, Vol. 39, No. 1, pp50-56.

[2] "'What is Storage as a Service (SaaS)? - Definition from Whatis.com" Feb., 2010. [Online]. Available: http://searchstorage.techtarget.com/sDefinition/0,,sid5_gci1264119,00.html. [Accessed: June, 2010].

[3] Paul ,M. & Saxena, A, (July 2010) "Proof Of Erasability For Ensuring Comprehensive Data Deletion In Cloud Computing", CNSA '10:Proceedings of the Third International Conference, Communications in Computer and Information Science, Vol. 89, pp 340-348.

[4] Kishi,Gregory, "Method and system for shredding data within a data storage subsystem, " U. S. Patent 7308543B2, March, 2005.

[5] Peter Gutmann, "Secure deletion of data from magnetic and solid-state memory," in Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography, 1996, pp. 8-8.

[6] Ari. Juels and Burton. Kaliski, "Proofs of retrievability for large files," in CCS '07:Proceedings of the 14th ACM conference on Computer and communications security, 2007 pp. 584-597.

[7] H. Shacham and W Brent.(2008): Compact proofs of Retrievability. Cryptology ePrint Archive, Report 2008/073 .Available: http://eprint.iacr.org/2008/073.pdf

[8] "Amazon Elastic Compute Cloud (EC2)" Feb., 2010 [Online]. Available: http://www.amazon.com/ec2/ [Accessed: June, 2010].

[9] "3X Systems private cloud" Mar., 2010 [Online]. Available: http://www.3x.com/ [Accessed: June, 2010].

[10] "The 100 Coolest Cloud Computing Vendors" Feb., 2010. [Online]. Available: http://www.crn.com/storage/222600510 [Accessed: June, 2010].

[11] "Service level agreement" Feb., 2010. [Online]. Available: http://en.wikipedia.org/wiki/Service_level_agreement [Accessed: June, 2010].

[12] "Data Erasure" Feb., 2010. [Online]. Available: http://en.wikipedia.org/wiki/Data_erasure [Accessed: Aug, 2010].

Authors

Mithun Paul is a Technology Analyst at SETLabs, Infosys Technologies Ltd., Hyderabad, India, and received his BTech (2005) from BITS, Pilani.He has been active in research with publications in the likes of Springer etc. His research interests are in the areas of cryptographic protocols, authentication technologies, data and security assurance.

Ashutosh Saxena is a Principal Research Scientist at SETLabs, Infosys Technologies Ltd., Hyderabad, India, and received his MSc (1990), MTech (1992) and PhD in Computer Science (1999). The Indian government awarded him the post-doctorate BOYSCAST Fellowship in 2002 to research on 'Security Framework for E-Commerce' at ISRC, QUT, Brisbane, Australia. He is on the Reviewing Committees of various international journals and conferences. He has authored the book entitled PKI – Concepts, Design and Deployment, published by Tata McGraw-Hill, and also co-authored more than 70 research papers. His research interests are in the areas of authentication technologies, data privacy, key management and security assurance.