INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Literature Survey

K. Vidhya
Assistant Professor, Department Of Computer Science And Engineering
Sri Shakthi Institute Of Engineering And Technology, Coimbatore, India
D. Bala Gayathri
PG Scholar, Department Of Computer Science And Engineering
Sri Shakthi Institute Of Engineering And Technology, Coimbatore, India

Abstract:
This paper mainly contains the detail description of the various ways of providing security for the data in the public cloud. The are many issues in preserving the security for the data and this paper gives some solution for establishing the security to the data or files which are stored in the cloud environment.

1.Short Signatures From The Weil Paring

A short Signatures scheme based on a computational Diffie Hellman assumption on elliptic curves and a hyper elliptic curve. The signature length is half size of a DSA signature provides the same level of security. These signatures are typed by human or sent over a low-bandwidth channel. The two most frequently used signature schemes are RSA and DSA. Both RSA and DSA are relatively long signature .Elliptic curve variants of DSA are 320 bits long.

The proposed system scheme uses the signature length of 160 bits and provide security as same as the security provided by DSA signature with 320 bits. It is secure against existential forgery under a chosen message attack.

The signature scheme can use the CDH (Computational Diffie-Hellman) and it is hard, but the Decision Diffie-Hellman problem (DDH) is easy and the Verification of the signature can be done with the help of bilinear pairing on the curve. Implementing short signatures had some old problems, shortening the DSA signature provide the same level of security. Gap Diffie Hellman groups lead to short signature.

The algorithms used are Key Generation, Signing, and Verification. GDH signature is good against forgery under chosen message attack. Using Weil pairing, certain elliptic curves use GDH groups for providing security, bilinear mapping uses Weil pairing. Weil pairing is a mapping which contains some properties like Identity, Bilinear, Non-degenerative, computable. GDH signature is used here and provides better security by using elliptic curve cryptography.

2.Multi-Signatures In The Plain Public-Key Model And A General Forking Lemma

A multi-signature scheme enables a group of singers to produce a compact, joint signature on a common document, and has many potential uses. The existing schemes impose key setup or PKI (Public Key Infrastructure) requirements that make them impractical, such as requiring a dedicated, distributed key generation protocol amongst potential signers. These requirements limit the use of the schemes for the signers.

The proposed scheme is proven secure using the plain public-key model, meaning requires nothing more than that each signer has a (certified) public key.

Furthermore, the important simplification in key management achieved is not at the cost of efficiency or assurance. The scheme matches or surpasses known ones in terms of signing time, verification time and signature size, and is proven secure in the random-oracle model under a standard (not bilinear map related) assumption. The proof is based on a simplified and general Forking Lemma that may be of independent interest.

The MS (Multi Signature) scheme , the most significant practical obstacle to multi signatures at present is the key-setup requirements. The contribution is to remove this obstacle by presenting a multi signature scheme in the plain public-key model. This means that, with regard to key setup, nothing more is required than in any usage of public-key cryptography, namely that any potential signer has a public key. There is no dedicated key generation protocol. A signer is not assumed to have proved knowledge of its secret key to the

CA (Certified Authority), but only to have a standard certificate. Yet, security against rogue-key attacks is proved without the KOSK (knowledge of secret key) .

To elaborate, in the setting, the group of potential signers is dynamic: anyone possessing a (certified) public key can join at any time. In our security model, the adversary can corrupt a signer and choose its public key as a function of those of other (honest) signers. It is not required to supply the challenger with a matching secret key, meaning we prove security even when the adversary does not know the secret key underlying a public key it makes for itself. The fact that we do not need to assume any kind of proof of-knowledge of the secret key performed to the CA at the time a public key is registered and a certificate is obtained reduce the demands on the PKI and allows the protocols to be implemented within the current PKI. CA's need not take any special actions or change their functioning or software. In particular, the efficient multi-signature schemes based on discrete Logarithms based on RSA, factoring and parings.

3.Store, Forget, And Check: Using Algebraic Signatures To Check Remotely Administered Storage

As the Internet has increased in speed and bandwidth, remote storage of data over the network has become feasible. Peer-to-peer (P2P) storage systems, especially those based on the Distributed Object Location and Retrieval (DOLR) systems such as Oceanstore are an important class of such systems. Systems like these face a number of challenges such as data privacy, protection of the data against alteration, data loss due to node unavailability and the free rider problem.

We introduce new techniques based on algebraic signatures that allow a "data origination site" to verify that a remote site is storing data correctly, or whether a number of sites that collectively store a collection of objects is doing so correctly. Our scheme does not need the original data for its check, and only two small messages need be exchanged for each check. Both of these properties should be attractive to designers of remote storage schemes. As peer-to-peer technology has matured, a number of systems such as Oceanstore Intermemory have been built to utilize remote data storage. To protect against failure, this data is stored redundantly using either pure replication or m/n erasure coding.

Similarly, the proposed scheme where participants mutually store each other's backup data. All these schemes store data on sites that cannot be trusted. In addition to peer unavailability, they must face the problem of free riders. Free riders only pretend to store other data and thus enjoy the benefits of remote storage of their data without incurring any costs of their own. Our approach can be used to address the free rider problem as well as the more general problem of involuntary data loss or generic data corruption by using a system of challenges and responses. The naive algorithm requests random blocks of data from the storage site, verifying them against the locally-stored data. This is particularly easy in a remote back-up scheme, since the original of the data is still available, but becomes quite difficult in remote storage systems where the original is not retained. DOLR and other P2P storage systems that use redundancy in storage face an additional problem of assuring that all data reflects the same state. While stale data in a replicated system might still be useful, stale parity data in a scheme based on erasure coding usually prevents the reconstruction of the application data in case of need.

Algebraic signatures are unsuitable as cryptographically, secure hash functions such as MD5 or the SHA family of secure checksums; they are ideally suited for use in verifying remotely stored data in distributed systems.

4.Proving Retrievability And Possession Within The Cloud Storage Model

Recent trends within the Information Technology industry have seen a drastic growth of outsourced computing and infrastructure services to service providers, also known as public cloud computing.

One area of cloud computing that has gained significant ground is Storage as a Service, or SaaS, in which an entity stores data outside their local devices with an entity accessible via network. This storage can be a replacement for existing primary storage or used to augment existing storage infrastructure by providing off-site replication, data dispersion or tiered storage and is very attractive for storing large amounts of infrequently accessed data.

Although the main draw of the SaaS model is to save money by reducing infrastructure and expenses, it also introduces new complexities and variants to existing storage models. This paper focuses primarily on two additional complexities encountered when storing large amounts of data in the cloud. First, prove that data stored with a SaaS provider can actually be retrieved, which is referred to as Proof of Retrievability or POR.

Second, prove that a SaaS provider actually has and maintains the data that has been sent for storage. This is called Provable Data Possession, or PDP. At first these two complexities of cloud storage may seem trivial. However, they both become more complex when dealing with large amounts of data. Downloading the data in order to prove retrievability or possession defeats the purpose of storing data with a SaaS provider since it would require the same amount of storage locally.

Proofs of retrievability (PORs) schemes are one of the complexities that become evident when large files are stored in untrusted or semi-trusted cloud storage providers. SaaS has many use cases, including long term storage of large data sets in order to provide archiving of infrequently accessed data. The ability to prove that data can be retrieved and has not been modified becomes even more important as file size increases to a point where downloading a local copy becomes a time consuming task.

The goal of POR is to allow a client to determine that a SaaS provider actually possesses a data object without downloading the data. In this scheme the client is the verifier and the SaaS provider is the prover. This provides a mechanism for regular verification that data still exists in its original form which can be used to augment data archiving and dispersion architectures. Here, the POR is good for the data which are stored on static region but it is not good for the data which are stored on the dynamic location. Here, the future work concentrate on improving the POR.

5.Provable Data Possesssion At Entrustedstores

Ateniese,G et al proposed provable Data Possession at Entrusted Stores Introduce a model for provable data possession (PDP) that allows a client that has stored data at an entrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. Archival network storage presents unique performance demands. Given that file data are large and are stored at remote sites, accessing an entire file is expensive in I/O costs to the storage server and in transmitting the file across a network. Reading an entire archive, even periodically, greatly limits the scalability of network stores.(The growth in storage capacity has far outstripped the growth in storage access times and bandwidth

Furthermore, I/O incurred to establish data possession interferes with on-demand bandwidth to store and retrieve data. We conclude that clients need to be able to verify that a server has retained file data without retrieving the data from the server and without having the server access the entire file. Some schemes provide a weaker guarantee by enforcing storage complexity: The server has to store an amount of data at least as large as the client's data, but not necessarily the same exact data. The concept of a Homomorphic verifiable tag is used as a building block for the PDP schemes. Given a message m (corresponding to a file block), Tm is its Homomorphic verifiable tag. The tags will be stored on the server together with the file F. Homomorphic verifiable tags act as verification metadata for the file blocks and, besides being unforgeable, they also have the following properties: Block less verification- Using HVTs the server can construct a proof that allows the client to verify if the server possesses certain file blocks, even when the client does not have access to the actual file blocks. Homomorphic tags-Given two values T_{mi} and T_{mj}, anyone can combine them into a value T_{mi} + T_{mj} corresponding to the sum of the messages mi + mj.

Verifying the authenticity of data has emerged as a critical issue in storing data on untrusted servers. It arises in peer-to-peer storage systems, network file systems, long-term archives, web-service object stores, and database systems. Such systems prevent storage servers from misrepresenting or modifying data by providing authenticity checks when accessing data. However, archival storage requires guarantees about the authenticity of data on storage, namely that storage servers possess data. It is insufficient to detect that data have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Archival storage servers retain tremendous amounts of data, little of which are accessed.

They also hold data for long periods of time during which there may be exposure to data loss from administration errors as the physical implementation of storage evolves for example backup and restore, data migration to new systems, and changing memberships in peer-to-peer systems. Provide a weaker guarantee by enforcing storage complexity. The server has to store an amount of data at least as large as the client's data, but not necessarily the same exact data.

6.Enabling Public Auditability And Data Dynamics For Storage Security In Cloud Computing

The work of enabling public auditability and data dynamics for storage security (Wang 2009) studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, the task of allowing a third party auditor (TPA) is considered, on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only.

The network architecture for cloud data storage has three different network entities can be identified as follows: i)Client- an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations; ii) Cloud Storage Server (CSS)- an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data; iii)Third Party Auditor- an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request. In the cloud paradigm, by putting the large data files on the remote servers, the clients can be relieved of the burden of storage and computation. As clients no longer possess their data locally, it is of critical importance for the clients to ensure that their data being correctly stored and maintained. That is, clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies. In case the clients do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the monitoring task to a trusted TPA.

Verification schemes with public auditability are considered and any TPA in possession of the public key can act as a verifier. We assume that TPA is unbiased while the server is untrusted. For application purposes, the clients may interact with the cloud servers via CSP to access or retrieve their prestored data. More importantly, in practical scenarios, the client may frequently perform block-level operations on the data files. The most general forms of these operations are modification, insertion, and deletion.

The design goals can be summarized as the following: (i) Public auditability for storage correctness assurance- to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand. (ii) Dynamic data operation support: to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support. (iii) Blockless verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern. Here Third party Auditor can able to see the local copy of the data. So, some security problems may arise like integrity of the data may be get affected The Third Party auditor have chance to leak client's private information. So, some security issues got arised here.

7.Scalable And Efficient Provable Data Possession

Storage outsourcing is a raising trend which prompts a number of interesting security issues, many of which have been extensively investigated in the past. However, Provable Data Possession (PDP) is a topic that has only recently appeared in the research literature. The main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing its client's (potentially very large) outsourced data. The storage server is assumed to be entrusted in terms of both security and reliability. In other words, it might maliciously or accidentally erase hosted data; it might also relegate it to slow or off-line storage.

The problem is exacerbated by the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form. A highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption.

Also, in contrast with its predecessors, our PDP technique allows outsourcing of dynamic data, it efficiently supports operations, such as block modification, deletion and append. The project construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption.

The proposed scheme consists of two phases: setup and verification (also called challenge) owner and server. Our scheme is based entirely on symmetric-key cryptography. The main idea is that, before outsourcing, OWN pre-computes a certain number of short possession verification tokens, each token covering some set of data blocks. The actual data is then handed over to SRV. Subsequently, when OWN wants to obtain a proof of data possession, it challenges SRV with a set of random-looking block indices. In turn, SRV must compute a short integrity check over the specified blocks (corresponding to the indices) and return it to OWN. For the proof to hold, the returned integrity check must match the corresponding value pre-computed by OWN. However, in our scheme OWN has the choice of either keeping the pre-computed tokens locally or outsourcing them – in encrypted form – to SRV. Notably, in the latter case, OWN's storage overhead is constant regardless of the size of the outsourced data.

The Project developed and presented a step-by-step design of a very light-weight and provably secure PDP scheme. It surpasses prior work on several counts, including storage, bandwidth and computation overheads as well as the support for dynamic operations. However, since it is based upon symmetric key cryptography, it is unsuitable for public (third party) verification.

8.Privacy-Preserving Audit And Extraction Of Digital Contents

A growing number of online services, such as Amazon, Yahoo!, Google, Snapfish, and Mozy.com aim to profit by storing and maintaining lots of valuable user data. Example uses of this storage include online backup, email, photo sharing, and video hosting. Many of this service offer a small amount of "teaser" storage for free, and charge for larger, upgraded versions of the service.

Studies of deployed large-scale storage systems show that no storage service can be completely reliable; all have the potential to lose or corrupt customer data. Today, a customer that wants to rely on these services must make an uneducated choice. He has only negative newsworthy anecdotes on which to base his decision, and service popularity or "brand name" is not a positive indicator of reliability. To know if his data is safe, he must either blindly trust the service or laboriously retrieve the hosted data every time he wants to verify its integrity, neither of which is satisfactory.

Unfortunately, to date, there are no fair and explicit mechanisms for making these services accountable for data los our proposed solution to provide storage service accountability is through independent, third party auditing and arbitration. The customer and service enter into an agreement or contract for storing data in which the service provides some type of payment for data loss or failing to return the data intact, e.g. free prints, refunds, or insurance. In such an agreement, the two parties have conflicting incentives.

The service provider, whose goal is to make a profit and maintain a reputation, has an incentive to hide data loss. On the other hand, customers are terribly unreliable, e.g. casual home users. Customers can innocently (but incorrectly) or fraudulently claim loss to get paid. Thus, we involve an independent, third party to arbitrate and confirm whether stored and retrieved data is intact.

Our protocols have three important operations, initialization, audit, and extraction, and we primarily focus on the latter two. For audits, the auditor interacts with the service to check that the stored data is intact. For extraction, the auditor interacts with the service and customer to check that the data is intact and return it to the customer.

These protocols have the following salient features: Audit: With minimal long-term state, an auditor can efficiently and repeatedly check stored contents on behalf of the customer. In these audits, the service must prove that contents are completely unchanged. Extraction: Upon retrieval, if the customer doubts the integrity of the data, the customer can use the extraction protocol which routes the data through the auditor to the customer. During extraction, the auditor can determine which party is at fault: whether the service lost data or which party is cheating by not obeying the protocol. Thus, the auditor can arbitrate a data retention contract.

All our protocols do not reveal the data contents to the auditor. Our auditing protocols are zero-knowledge, providing no added information to the auditor. Our extraction protocols prevent an adversarial auditor from recovering the data contents. Yet, they still allow the auditor to check the integrity of retrieved data and forward it so that a customer can efficiently recover the contents. A customer does not need to maintain any long-term state. For example, he does not need to keep "fingerprints" or hashes to audit the stored data, or keep secret keys to decrypt the stored data upon retrieval.

A straightforward solution for maintaining privacy during audits is for the customer to encrypt his contents using symmetric-key encryption and keep those keys intact and secret from uninvited parties. Then, the auditor can use existing provably secure, challenge-response schemes on the encrypted contents. This solution is unsatisfactory because an unsophisticated customer is increasingly likely over time either to lose the keys and be unable to recover the contents, or to leak the keys. Finally, our current schemes require the auditor to be trusted and not collude with either party. In real world auditing situations in other contexts (e.g. financial auditing), such

collusions can occur and have occurred in the past. In this case, a scheme that allows auditing by multiple auditors without the traditional overheads of Byzantine fault-tolerance techniques would be useful.

9.Dynamic Provable Data Possession

As storage-outsourcing services and resource-sharing networks have become popular, the problem of efficiently proving the integrity of data stored at untrusted servers has received increased attention.

In the provable data possession (PDP) model, the client preprocesses the data and then sends it to an untrusted server for storage, while keeping a small amount of meta-data. The client later asks the server to prove that the stored data has not been tampered with or deleted (without downloading the actual data). However, the original PDP scheme applies only to static (or append-only) files.

The client should not download all stored data in order to validate it since this may be prohibitive in terms of bandwidth and time, especially if the client performs this check frequently (therefore authenticated data structure solutions cannot be directly applied in this scenario).

Ateniese et al. have formalized a model called provable data possession (PDP). In this project, data (often represented as a file F) is preprocessed by the client, and metadata used for verification purposes is produced. The file is then sent to an untrusted server for storage, and the client may delete the local copy of the file. The client keeps some (possibly secret) information to check server's responses later. The server proves the data has not been tampered with by responding to challenges sent by the client. The authors present several variations of their scheme under different cryptographic assumptions. These schemes provide probabilistic guarantees of possession, where the client checks a random subset of stored blocks with each challenge.

A definitional framework and efficient constructions for dynamic provable data possession (DPDP), which extends the PDP model to support provable updates to stored data is presented. A new version of authenticated dictionaries based on rank information is used. The price of dynamic updates is a performance change from O(1) to O(log n), for a file consisting of n blocks, while maintaining the same (or better, respectively) probability of misbehavior detection.

- Definition 1 (DPDP Scheme):In a DPDP scheme, there are two parties. The client wants to off-load her files to the untrusted server. A complete definition of a DPDP scheme should describe the following (possibly randomized) efficient procedures.
- Definition 2 (Security of DPDP):That a DPDP scheme is secure if for any probabilistic polynomial time (PPT) adversary who can win the following data possession game with non-negligible probability, there exists an extractor that can extract (at least) the challenged parts of the file by resetting and challenging the adversary polynomially many times.

10.Toward Publicly Audi-Table Secure Cloud Data Storage Services

Cloud computing is the long dreamed vision of computing as a utility, where data owners can remotely store their data in the cloud to enjoy on-demand high-quality applications and services from a shared pool of configurable computing resources.

Here, we propose that publicly auditable cloud data storage is able to help the cloud become fully established. The reason that linear combination of sampled blocks may potentially reveal owner data information is due to the following fact about basic linear algebra theory: if enough linear combinations of the same blocks are collected, the TPA can simply derive the sampled data content by solving a system of linear equations.

By using homomorphic-authenticator-based techniques in a publicly auditable cloud data storage system. From the perspective of protecting data privacy, the owners, who own the data and rely on the TPA just for the storage security of their data, do not want this auditing process introducing new vulnerabilities of unauthorized information leakage into their data security . Without a properly designed auditing protocol, encryption itself cannot prevent data from flowing away toward external parties during the auditing process. Thus, it does not completely solve the problem of protecting data privacy but just reduces it to the one of managing the encryption keys. Unauthorized data leakage still remains a problem due to the potential exposure of encryption keys.

11.MR-PDP: Multiple-Replica Provable Data Possession

(MR-PDP): A provably-secure scheme that allows a client that stores t replicas of a file in a storage system to verify through a challenge-response protocol that (1) each unique replica can be produced at the time of the challenge and that (2) the storage system uses t times the storage required to store a single replica. MR-PDP extends previous work on data possession proofs for a single copy of a file in a client/server storage system . Using MR-PDP to store t replicas is computationally much more efficient than using a single-replica PDP scheme to store t separate, unrelated files.

Multiple-replica PDP system uses replication in order to improve the data availability and reliability of a single-replica PDP system. The basic principle of data Replication is well understood. By storing redundant copies of the data, one can ensure that if some of the copies are destroyed, the data can still be recovered from the remaining Copies.

We address the problem of creating multiple unique replicas of a file in a distributed storage system. This allows a client to query the distributed system to ensure that there are multiple unique copies of its file stored in the network even when storage sites collude. Our original motivation was to give a data owner that archives data with third-party storage services, such as Amazon S3 or the Storage Request Broker, the ability to perform introspection and maintenance on its data. However, these techniques apply to all replication-based, distributed, and untrusted storage systems, including peer-to-peer storage systems .

The natural solutions to this problem result in significant time, space, and management overheads. A simple way to make replicas unique and differentiable is by encryption. If the client were to generate each replica by encrypting the data under different keys that are kept secret from the servers, then the servers could not compare the replicas, use one replica to answer challenges for another, or

compress replicas with respect to one another. Each replica is a separate file to be created and checked individually, using a protocol for checking data possession. (Cross-checking protocols would not work as encrypted replicas cannot be compared.) In this simple approach, the computation time for both replica creation and checking grow linearly in the number of replicas.
Here, we describe cryptographic constructs that allow for a client to securely establish that the network stores multiple unique replicas. The scheme uses a constant amount of metadata for any number of replicas and new replicas may be created dynamically without preprocessing the file again. Also, multiple replicas may be checked concurrently, so that checking t replicas is less expensive than t times the cost of checking a single replica. Thus, our solution overcomes the time, space, and management overheads associated with the natural solution mentioned previously.
Replication systems that rely on untrusted servers have another generic limitation. To prove data availability, the servers can produce replicas on demand upon a client's challenge; however, this does not prove that the actual replicas are stored at all times. For example, malicious servers may chose to introduce dependencies among replicas, by encrypting the replicas before storing them. Replicas can then be decrypted and served on demand whenever they are requested by clients. By storing the encryption key in a single location, the malicious servers can effectively negate any reliability improvements achieved by storing the replicas at different locations. Loss of the encryption key means loss of all the replicas. MR-PDP possesses many integrity problems to the data due to replication of data. So, the data should be stored in multiple locations and they should be encrypted with the help of private or public keys.

12.Towards A Theory Of Accountability And Audit

We make two contributions toward bringing such formal foundations to the study of accountability. First, we describe an operational model of accountability based systems. Honest and dishonest principals are described as agents in a distributed system where the communication model guarantees point-to-point integrity and authenticity.
Auditors and other trusted agents (such as trusted third parties) are also modeled internally as agents. Behaviours of all agents are described as processes in process algebra with discrete time. Auditor implement ability is ensured by forcing auditor behaviour to be completely determined by the messages that it receives.
Second, we describe analyses to support the design of accountability systems and the validation of auditors for finitely systems (those with finitely many principals running finite state processes with finitely many message kinds). We compile finitely systems to (turn-based) games and use alternating temporal logic to specify the properties of interest. This permits us to adapt existing model-checking algorithms for verification.
Our results provide the foundations necessary to explore tradeoffs in the design of mechanisms that ensure accountability. The potentially conflicting design parameters include the efficiency of the audit, the amount of logging, and the required use of message signing, watermarking, or trusted third parties. Design choices place constraints on the auditor, the agents of the system and the underlying communication infrastructure.
We now describe our model and its relation to the following properties. The discussions intended to establish intuitions, with formalities deferred to later sections.

- Upper bound: Every agent guilty of a dishonest action is blamed by the auditor.
- Lower bound: Everyone blamed by the auditor is guilty.
- Overlap: At least one of the agents blamed by the auditor is guilty.
- Liveness: The auditor is always successful in blaming a non-empty subset of agents.
- Blamelessness: Honest agents have a strategy to avoid being pronounced a possible offender by an auditor.

Agents, we model the behaviour of principals (both honest and dishonest) as agents in a distributed system. Auditors are also modeled internally as honest agents. We use processes to specify an upper bound on honest behaviour: a principal is behaving honestly in a run whenever their contribution to the run is a trace of an honest process. And is honest agent is unconstrained. A run of an agent reveals its dishonesty if it is not a permitted trace for an honest agent.
The communication model captures point-to-point communication over an underlying secure communication mechanism which provides integrity and authenticity guarantees, but provides no additional mechanisms for non-repudiation or end-to-end security. This model is realizable using transport mechanisms such as TLS. Dishonest agents may collaborate arbitrarily. This means that the auditor has to achieve its objectives independent of potential cartels of dishonest agents.
Auditors cannot detect cartels of dishonest agents who conduct dishonest exchanges amongst themselves. Thus, our auditors cannot in general satisfy Upper bound. To see this, consider the leakage of patient records to a dishonest non-health professional by a dishonest health professional via out-of-band mechanisms .Such leakage of records by dishonest agents solely to dishonest agents will not be detected at all by an auditor in our framework .Mandatory logging and responsiveness. Even in the case that the auditor has become aware of dishonest behaviour and initiates an audit, the auditor is powerless unless there are statutory and enforceable reporting requirements on the honest agents.

13.Towards Usage Control Models: Beyond Traditional Access Control

With today's revolutionary innovations in information technology and their impact on our society, we are encountering a series of new problems on security and privacy issues. Scientists have tried to resolve these problems by focusing on their own target issues. These efforts have produced significant results in their own areas and the area of information security overall. Access control is one of these areas and has been considered as a major issue in information security community since the beginning of the information security discipline.

Access control literature has traditionally focused on the protection of data in a closed environment. The enforcement of control has been primarily based on identity and attributes of a known user or a process by using a reference monitor and specified authorization rules. Trust management relates authorization to a user's capability and properties.
While both traditional access control and trust management have focused on the control of access to server-side objects, there have been other studies to control access to and usage of digital objects even after the objects are disseminated [4,6,7]. This area of study has come to be called digital rights management (DRM). Because of DRM's potential opportunity for commercial sector, current DRM solutions are largely focused on payment-based dissemination controls though its underlying technologies can be also used for controls of non-payment based dissemination.
Because each of access control, trust management, and DRM has focused on its own target problems and detailed solutions for these problems, we lack a comprehensive, systematic approach for controls on usage of digital objects regardless of specific circumstances. In this paper, we introduce a consolidated view of all these three areas and define a model called usage control that unifies all three areas. The term "usage" means usage of rights on digital objects. And the term "rights" includes rights for use of digital object and rights for delegation of the rights.

14.Logic For Auditing Accountability In Decentralized Systems

The problem here is how we can make sure that the data is being used only according to Alice's wishes. Notice that in the above scenario Big Brother might sell Alice's data to Big Sister, who in turn might sell part of it to Small Nephew, and so on. This problem is not only that of privacy protection in a distributed setting. In fact, modern scenarios of digital asset delivery (where a digital asset can be anything ranging from a piece of private information to a movie or a character in a multiplayer game) are departing from the usual schemas in which the assets are equipped with an immutable usage policy that applies to the whole distribution chain. Instead, we are moving towards a situation in which information brokers collect, combine and redistribute digital assets.
We present a logic data access and agent accountability in a setting in which data can be created, distributed and re-distributed. Using this logic, the owner of the data attaches a usage policy to the data, which contains a logical specification of what actions are allowed with the data, and under which conditions. This logic allows for different kind of accountability and it is shown to be sound. Therefore, we consider an alternative to policy enforcement, based on an analogy with the real world, where people are not always controlled for correct behaviour. Instead, eventually an agent (say Alice) might be suspected of incorrect behaviour; in that case, an authority would query Alice for a justification of her actions. This justification can be supported by evidence that the authority can check and validate.
The problem of policy enforcement—guaranteeing that data is used according to established policies and is not disclosed to or corrupted by unauthorized users—is paramount to our IT dominated world. In particular, in inter organizational cooperation, collaborating organizations often need to protect their intellectual property by enforcing policies that capture objectives such as "this document may be seen and modified only by senior engineers working on project X"
The approaches to enforcement of such policies most commonly used are preventative, in the sense that unauthorized actions are prevented from occurring. By far the most widely used method is access control (AC), wherein unauthorized access is prevented.

- Policy enforcement must be end-to-end. This means that information in documents is protected from the time it is entered into the system until the point at which it is deleted, no matter where the information may flow in the mean time. So when a document is released to a remote host or domain, the policy that governs it remains associated with it and enforced.
- Users should be able to create, modify and merge documents using the programs they are accustomed to use for these purposes.
- In dynamic cooperation's, where documents are owned by different users or organizations, the owner of a document should be free to determine the (initial) policy that applies to it.
- Policies should be able to refer to dynamic groups, and should govern both the use and release of the document(s) they refer to as well as the administration of the policy itself (e.g. "any manager may modify this document as well as restrict this policy").
- The environment is highly decentralized and does not offer compatible role or level assignment (as in RBAC or MAC): in dynamic cooperation's we cannot expect participating organizations to find a way of merging their MAC or RBAC systems for the time required by the cooperation.
- When they believe circumstances justify it, it should be possible for a user to elect to perform actions that are not normally permitted. In this case, the decision should be subject to inquiry and review. Only users that can be held accountable should have this option.

15.A Privacy Preserving System For Cloud Computing

While the storage of corporate data on remote servers is not a new development, current expansion of cloud computing justifies a more careful look at its actual consequences involving privacy and confidentiality issues. We focus on the Software as a Service kind of utility computing model. The proposed architecture addresses the important security question: 'To what extent, an organization has to trust client users, system administrators and service providers?'
Remote access to corporate networks is already an essential aspect of the corporate work environment. Client systems are used by employees both inside the corporate Intranet, and remotely from home. In both scenarios, applications require effective requests and use of Cloud-based computing resources on the fly.

The system administrator A has the role of setting up an environment, and a base system for the encryption proxy. At this stage, the system administrators have privileges of doing anything they like in the system. The system afterwards is not anymore modifiable except for those unmeasured configurations. This is the post-system setup stage, in which system-administrators A have the tasks of cloning new encryption proxies (same hardware, same software and same configuration) and managing the configuration files. The system administrators have for each proxy an admin-panel, so he can submit new (valid-signed) XACML-files, change the administration credentials (for the admin-panel), and to edit lists of used cloud database.

16.References

1. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010.
2. 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, May 2011.
3. 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, 2007.
4. M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, 2008.
5. C. Wang, K. Ren, W. Lou, and J. Li, "Towards Publicly Auditable Secure Cloud Data Storage Services," IEEE Network Magazine, vol. 24, no. 4, pp. 19-24, July/Aug. 2010.
6. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," J. Cryptology, vol. 17, no. 4, pp. 297-319, 2004.
7. G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik,"Scalable and Efficient Provable Data Possession," Proc. Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '08), pp. 1-10, 2008.
8. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. ACM Conf. Computer and Comm. Security (CCS '09), pp. 213-222, 2009
9. M. Bellare and G. Neven, "Multi-Signatures in the Plain Public- Key Model and a General Forking Lemma," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 390-399, 2006.
10. 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), 2006.
11. R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '08), pp. 411-420, 2008.