**Towards Shared Ownership in the Cloud**

**Abstract**

Cloud storage platforms promise a convenient way for users to share files and engage in collaborations, yet they require all files to have a single owner who unilaterally makes access control decisions. Existing clouds are, thus, agnostic to the notion of shared ownership. This can be a significant limitation in many collaborations because, for example, one owner can delete files and revoke access without consulting the other collaborators. In this paper, we first formally define a notion of shared ownership within a file access control model. We then propose two possible instantiations of our proposed shared ownership model. Our first solution, called Commune, relies on secure file dispersal and collusion resistant secret sharing to ensure that all access grants in the cloud require the support of an agreed threshold of owners. As such, Commune can be used in existing clouds without modifications to the platforms. Our second solution, dubbed Comrade, leverages the blockchain technology in order to reach consensus on access control decision. Unlike Commune, Comrade requires that the cloud is able to translate access control decisions that reach consensus in the blockchain into storage access control rules, thus requiring minor modifications to existing clouds.

**Existing System**

Even though the cloud promises a convenient way for users to share files and effortlessly engage in collaborations, it still retains the notion of individual file ownership. That is, each file stored in the cloud is owned by a single user, who can unilaterally decide whether to grant or deny any access request to that file. However, the individual ownership is not suitable for numerous cloud-based applications and collaborations. Consider a scenario where a number of research organizations and industrial partners want to set up a shared cloud repository to collaborate on a joint research project. If all participants contribute their research efforts to the project, then they may want to share the ownership over the collaboration files so that all access decisions are agreed upon among the owners. There are two main arguments why this may be preferred to individual ownership. First, a sole owner can abuse his rights by unilaterally making access control decisions. The community features a number of anecdotes where users revoke access to shared files from other collaborators. Second, even if owners are willing to elect and trust one of them to make access control decisions, the elected owner may not want to be held accountable for collecting and correctly evaluating other owners’ policies. For example, incorrect evaluations may incur negative reputation or financial penalties.

**Proposed System**

In proposed system we implement the shared ownership in the cloud by leveraging functionality from the blockchain. Our solution, dubbed Comrade, enables a distributed blockchain-based enforcement of the SOM access control policy in a cooperative cloud. Unlike Commune, Comrade does not assume an agnostic cloud, and requires the cloud operator to cooperate and to interface with the blockchain. Since SOM does not specify concrete file access operations, we instantiate Comrade with write and read actions. We formalize the notion of shared ownership within a file access control model named SOM, and use it to define a novel access control problem of distributed enforcement of shared ownership in existing clouds. We propose a first solution, called Commune, which distributively enforces SOM and can be deployed in an agnostic cloud platform. Commune ensures that (i) a user cannot read a file from a shared repository unless that user is granted read access by at least t of the owners, and (ii) a user cannot write a file to a shared repository unless that user is granted write access by at least t of the owners. We propose a second solution, dubbed Comrade, which leverages functionality from the blockchain technology in order to reach consensus on access control decision. Comrade improves the performance of Commune, but requires that the cloud is able to translate access control decisions that reached consensus in the blockchain into storage access control rules, thus requiring minor modifications of existing clouds.

**Implementation**

**Module Description**

The Modules are

 1.Blockchain and Smart Contracts Module

 2.Comrade Module

 3. Collusion Resistant Secret Sharing Module

 4. Comrade security Module

**1.Blockchain and Smart Contracts Module**

The notion of blockchain was originally introduced by the wellknown proof-of-work hash-based mechanism that confirms cryptocurrency payments in Bitcoin. The PoW-based blockchain ensures that all transactions and their order of execution are available to all blockchain nodes, can be verified by all involved entities and aids the consensus between the parties. Bitcoin’s blockchain fueled innovation, and a number of innovative applications have already been devised by exploiting the secure and distributed provisions of the underlying blockchain. Prominent applications include secure timestamping and smart contracts . Smart contracts refer to binding contracts between two or more parties that are executed by all blockchain nodes. Namely, smart contracts implement state machine replication. Smart contracts typically consist of a self-contained code and persistent storage available to all blockchain nodes. For example, Ethereum is a decentralized platform that enables the execution of arbitrary applications (or contracts) on its blockchain. Owing to its support for a Turing-complete language, Ethereum (which currently also relies on PoW-based consensus) offers an easy means for developers to deploy their distributed applications in the form of smart contracts.

**2.Comrade Module**

The main idea behind Comrade is that a smart contract can instantiate a trusted third party that can evaluate user credentials against owners access policies in a trustworthy manner. This is a basic provision of the blockchain technology that holds as long as the security assumptions on the blockchain hold. (We will argue on those assumption in the security analysis). Hence, in Comrade, a smart contract assists the cloud’s PDP ensuring trustworthy handling of policies and credentials. Differently from Commune, however, Comrade needs the cloud to be “shared-ownership aware” and enforce the policies defined by the smart contract. (Recall that we assume the cloud to correctly enforce access policies.)

**3. Collusion Resistant Secret Sharing**

Collusion Resistant Secret Sharing (CRSS) Similar to threshold secret-sharing schemes, CRSS allows one party to distribute a secret among a set of designated shareholders, so that any subset of shareholders of size equal to or greater than the threshold can reconstruct the secret. Furthermore, CRSS allows shareholders to issue to other users delegation to reconstruct the secret. If a user collects enough (i.e., above the threshold) delegations, he can rightfully reconstruct the secret. However, users cannot pool their delegations to reconstruct the secret, unless one of them has collected enough delegations. In Commune, CRSS is used to secret-share the key K used in SFD, in order to achieve collusion resistance.

**4. Comrade security Module**

In Comrade user authentication at the server and at the owner contract rules out external adversaries. Further, assuming that the aforementioned security provisions of the blockchain are met, the owner contract essentially instantiates a trusted third party that handles access policies and credentials. Moreover, since an untampered log of the system is available from the blockchain, computation of the contract can be publicly audited. Each credential issued by an owner is instantiated by a blockchain transaction that is confirmed within the blockchain by the validator/miners. As such, the vote of an owner on an access control decision is eventually taken into account by the smart contract and persisted to the blockchain. Similarly, access requests for a file by owners are processed, saved to the blockchain, and evaluated by the smart contract against the policy define by the owners of that file.

Algorithm

**Information dispersal algorithms**

Information dispersal algorithms encode a file in n chunks so that any t chunks (where t≤ n) are sufficient to decode it. However, information dispersal algorithms do not provide any security guarantees if the number of available chunks is smaller than t: any party with fewer than t chunks may still recover meaningful information about the original file’s content.

Previous work on securing information dispersal algorithms combines erasure codes with All-Or-Nothing Transforms (AONT) . The latter is an efficient block-wise transformation that maps an n-block bitstring in input to an n0-block bitstring in output (with n,≥n). AONTs are designed in such a way that, unless all the n0 output blocks are available, it is hard to recover any of the input blocks.

Existing AONTs leverage block ciphers and rely on the secrecy of a cryptographic key that is embedded within the output blocks. Given all AONT output blocks, the key can be recovered; once the key is known, individual blocks can be reverted, independently of other blocks. Current AONTs, therefore, preserve their all-or-nothing property only for one time: knowledge of the cryptographic key allows to revert single output blocks and to recover parts of the original data. This is at odds with our security requirements. As argued before, we cannot prevent users from caching a local copy of the file and reading it at later time when their read rights may have been revoked. However, we still want to provide revocation of a user who only stored the encryption key at the time when he had read access to the file.

We therefore introduce a new scheme, called Secure File Dispersal (SFD), that combines information dispersal algorithms with an AONT that preserves its all-or-nothing property even if the adversary has the encryption key.

Encryption

**Attribute-based encryption**

Attribute-based encryption (ABE) is as public-key cryptography, where a message is encrypted for a specific receiver using the receiver’s public-key. ABE defines the identity not atomic but as a set of attributes, e.g., roles, and messages can be encrypted with respect to subsets of attributes (key-policy ABE - KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE - CP-ABE). The key issue is, that someone should only be able to decrypt a ciphertext if the person holds a key for "matching attributes" (more below) where user keys are always issued by some trusted party.

**Architecture**

****

**SYSTEM REQUIREMENTS**

➢ **H/W System Configuration:-**

➢ Processor - Pentium –IV or Later Version

➢ RAM - 4 GB (min)

➢ Hard Disk - 40 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**Software Requirements:**

* Operating System - Windows XP or Later Version
* Coding Language - Java/J2EE(JSP,Servlet)
* Front End - J2EE
* Back End - MySQL

**Conclusion**

Even though existing cloud platforms are used as shared repositories, they do not support any notion of shared ownership. We consider this a severe limitation because contributing parties cannot jointly decide how their resources are used. In this paper, we introduced a novel concept of shared ownership and we described it through a formal access control model, called SOM. We then propose two possible instantiations of our proposed shared ownership model. Our first solution, called Commune, relies on secure file dispersal and collusion-resistant secret sharing to ensure that all access grants in the cloud require the support of an agreed threshold of owners. As such, Commune can be used in existing agnostic clouds without modifications to the platforms. Our second solution, dubbed Comrade, leverages the blockchain technology in order to reach consensus on access control decision. Unlike Commune, Comrade requires that the cloud is able to translate access control decisions that achieved consensus in the blockchain into storage access control rules. Comrade, however, shows better performance than Commune.

**Future Enhancement**

The future enhancement of the project is given the rise of personal clouds we argue that Commune and Comrade find direct applicability in setting up shared repositories that are distributively managed atop of the various personal clouds owned by users. We therefore hope that our findings motivate further research in this area.