Survey on Ensuring Distributed Accountability for Data Sharing in the Cloud

Gayatri Karvande
ME, Compuer Engineering
JSPMNTC RSSOER Savitribai Phule Pune University
Pune, India
karvande.gayatri@gmail.com

Prof. Megha Borole
Assistant Professor, Computer Engineering
JSPMNTC RSSOER Savitribai Phule Pune University
Pune, India
megha.borole@gmail.com

***Abstract*—**Cloud computing is the use of computing of sources that are delivered as a service over a network for example on internet. It enables highly scalable services to be easily utilized over the Internet on an as needed basis. Important characteristic of the cloud services is that users' data are usually processed remotely in unknown machines that users do not operate. It can become a substantial barrier to the wide taking on cloud services. To address this problem highly decentralized responsibility framework to keep track of the actual usage of the user's data in the cloud. In this work has automated logging and distributed auditing mechanism. The Cloud Information Accountability framework proposed in this work conducts distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It conations two major elements: logger and log harmonizer. This methodology will also take concern of the JAR file by converting the JAR into obfuscated code which will adds an additional layer of security to the infrastructure. Rather than this here in this work, increase the security of user's data by provable data control for integrity verification.

Keywords-*Cloud computing, data sharing, information accountability framework, Provable data possession.*

*****

I. INTRODUCTION

Cloud services where services made available to users on demand via the internet, by providing for dynamically scalable and often virtualized resources as a service over the Internet. Now Days, there are a number of noticeable commercial and individual cloud computing services like Amazon, Google, Microsoft, Yahoo, and Salesforce [19]. Examples of cloud services include online backup solutions and data storage , Web-based e-mail services, hosted office suites and document collaboration services, database processing more. Furthermore, users may not know which machines actually process and host their data. While enjoying the advantage brought by this new technology, users also start concerned about losing control of their own data. The data processed on clouds are often deployed, causes to a number of issues related to accountability like the handling of personally identifiable information for cloud user. Such issues are becoming a noteworthy barrier to the wide adoption of cloud services [30]. It is more important for cloud users that monitor their usage of their data in the cloud. For example, users have to be able to ensure that their data are handled according to the service level agreements made at the time they sign on for services in the cloud. Traditional access control mechanisms uses centralized approach that is it uses centralized server in distributed environments which are not suitable, because of the following features characterizing cloud environments. First one is the, data handling can be outsourced/deployed by the direct cloud service provider (CSP) to other user in the cloud and these users can also instruct the tasks to others, and so on. Second, users are allowed to join and leave the cloud in a flexible manner. Because of this characteristic, data handling in the cloud goes through a complex and dynamic service chain which does not exist in traditional environments. To overcome the above problems, new novel approach, namely Cloud Information Accountability (CIA) framework, based on the concept of information accountability [44]. Unlike the privacy protection technologies which are built on the hide the information or lose it view, information accountability focuses on keeping the data usage transparent and trackable. CIA framework provides end-to end accountability in a highly distributed fashion. Most important feature of the CIA is that its ability of maintaining lightweight and powerful accountability that combines aspects of access control, usage control and authentication. In CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed. Associated with the accountability feature, also develop two distinct modes for auditing: push mode and pull mode. The push mode mentions to logs being periodically sent to the data owner or stakeholder while the pull mode mentions to an alternative approach whereby the user (or another authorized party) can retrieve the logs as needed. In CIA design presents substantial challenges, including uniquely identifying cloud service providers, ensuring the reliability of the log, adapting to a highly decentralized infrastructure, etc. In this existing approach toward addressing these issues is to hold and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. Users will send their data along with any policies according to their choice such as access control policies and logging policies that they want to enforce, enclosed in JAR files, to cloud service providers. Any access to the data will trigger an automated and authenticated logging mechanism local to the JARs. We refer to this type of enforcement as "strong binding" since the policies and the logging mechanism travel with the data. This strong binding exists even when copies of the JARs are created; thus, the user will have control over his data at any location. Such decentralized logging mechanism meets the dynamic nature of the cloud but also imposes challenges on ensuring the integrity of the logging. To cope with this issue, we provide the JARs with a central point of contact which forms a link between them and the user. It records the error correction information sent by the JARs, which allows it to monitor the loss of any logs from any of the JARs. Moreover, if a JAR is not able to contact its central point, any access to its enclosed data will be denied. Currently, we focus on image files since images represent a very common content type for end users and organizations (as is proven by the popularity of Flickr [14]) and are increasingly hosted in the cloud as part of the storage services offered by the utility computing paradigm featured by cloud computing. Further, images often reveal social and personal habits of users, or are used for archiving important files from organizations. In addition, our approach can handle personal identifiable information provided they are stored as image files (they contain an image of any textual content, for example, the SSN stored as a .jpg file). We tested our CIA framework in a cloud testbed, the Emulab testbed [42], with Eucalyptus as middleware [41].

II. RELATED WORK

In this, we study on related works, which addresses the privacy and security issues in the cloud. Then, we briefly discuss works which adopt similar techniques as our approach but serve for different purposes.

A. Cloud Privacy and Security

Cloud computing has raised a range of important privacy and security issues [19], [25], [30]. Such issues are due to the fact that, in the cloud, users' data and applications reside—at least for a certain amount of time—on the cloud cluster which is owned and maintained by a third party.

Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. To date, little work has been done in this space, in particular with respect to accountability. Pearson et al. have proposed accountability mechanisms to address privacy concerns of end users [30] and then develop a privacy manager [31].

Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed. In [7], the authors present a layered architecture for addressing the end-to-end trust management and accountability problem in federated systems. The authors' focus is very different from ours, in that they mainly leverage trust relationships for accountability, along with authentication and anomaly detection.

Further, their solution requires third-party services to complete the monitoring and focuses on lower level monitoring of system resources. Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce [10], [21].

Arepresentative work in this area is given by [9]. The authors propose the usage of policies attached to the data and present a logic for accountability data in distributed settings. Similarly, Jagadeesan et al. recently proposed a logic for designing accountability-based distributed systems [20]. In [10], Crispo and Ruffo proposed an interesting approach related to accountability in case of delegation. Delegation is complementary to our work, in that we do not aim at controlling the information work-flow in the clouds. In a summary, all these works stay at a theoretical level and do not include any algorithm for tasks like mandatory logging.

To the best of our knowledge, the only work proposing a distributed approach to accountability is from Lee and colleagues [22]. The authors have proposed an agent-based system specific to grid computing. Distributed jobs, along with the resource consumption at local machines are tracked by static software agents. The notion of accountability policies in [22] is related to ours, but it is mainly focused on resource consumption and on tracking of sub jobs processed at multiple computing nodes, rather than access control.

B. Other Related Techniques

With respect to Java-based techniques for security, our methods are related to self-defending objects (SDO) [17]. Self-defending objects are an extension of the object-oriented programming paradigm, where software objects that offer sensitive functions or hold sensitive data are responsible for protecting those functions/data. Similarly, we also extend the concepts of object-oriented programming. The key difference in our implementations is that the authors still rely on a centralized database to maintain the access records, while the items being protected are held as separate files. In previous work, we provided a Java-based approach to prevent privacy leakage from indexing [39], which could be integrated with the CIA framework proposed in this work since they build on related architectures. In terms of authentication techniques, Appel and Felten [13] proposed the Proof-Carrying authentication (PCA) framework. The PCA includes a high order logic language that allows quantification over predicates, and focuses on access control for web services. While related to ours to the extent that it helps maintaining safe, high-performance, mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content. Another work is by Mont et al. who proposed an approach for strongly coupling content with access control, using Identity-Based Encryption (IBE) [26].

We also leverage IBE techniques, but in a very different way. We do not rely on IBE to bind the content with the rules. Instead, we use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and ciphertext attacks.

In addition, our work may look similar to works on secure data provenance [5], [6], [15], but in fact greatly differs from them in terms of goals, techniques, and application domains. Works on data provenance aim to guarantee data integrity by securing the data provenance.

They ensure that no one can add or remove entries in the middle of a provenance chain without detection, so that data are correctly delivered to the receiver. Differently, our work is to provide data accountability, to monitor the usage of the data and ensure that any access to the data is tracked.

Since it is in a distributed environment, we also log where the data go. However, this is not for verifying data integrity, but rather for auditing whether data receivers use the data following specified policies.

Along the lines of extended content protection, usage control [33] is being investigated as an extension of current access control mechanisms. Current efforts on usage control are primarily focused on conceptual analysis of usage control requirements and on languages to express constraints at various level of granularity [32], [34]. While some notable results have been achieved in this respect [12], [34], thus far, there is no concrete contribution addressing the problem of usage constraints enforcement, especially in distributed settings [32]. The few existing solutions are partial [12], [28], [29], restricted to a single domain, and often specialized [3], [24], [46]. Finally, general outsourcing techniques have been investigated over the past few years [2], [38]. Although only [43] is specific to the cloud, some of the outsourcing protocols may also be applied in this realm. In this work, we do not cover issues of data storage security which are a complementary aspect of the privacy issues.

III. CLOUD INFORMATION ACCOUNTABILITY

In this, we present an overview of the Cloud Information Accountability framework. The Cloud Information Accountability framework has automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two important parts: logger and log harmonizer.

A. Major Components

There are two important parts of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. Logger is responsible for logging access to that instance or copy. The log harmonizer forms the central part which allows the user access to the log files. The logger is strongly coupled with user's data. Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user X is only allowed to view but not to modify the data. The logger will control the data access even after it is downloaded by user X. The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and sends the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement. The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange. It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner.

The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions, which meets the second design requirement.

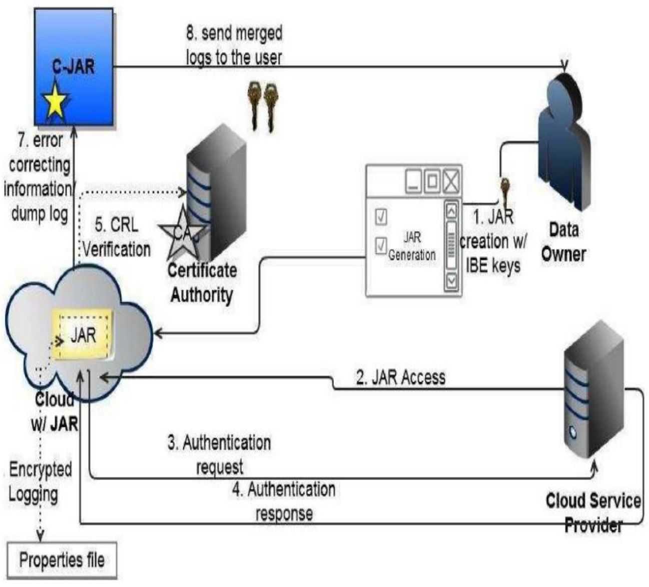

Figure 1. Overview of the cloud information accountability framework.

B. Data Flow

The CIA framework consist of combining data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture as described in Section 7. Using the generated key, the user will create a logger component which is a JAR file, to store its data items.

The JAR file includes a set of simple access control rules specifying whether and how the cloud servers, and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR (steps 3-5 in Fig. 1), we use OpenSSLbased certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication [8], wherein a trusted identity provider issues certificates verifying the user's identity based on his username. Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each .time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data (step 6 in Fig. 1). The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security (detailed discussion is in Section 7) without introducing any overhead except in the initialization phase. In

ISSN: 2321-8169
4189 - 4193

addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (step 7 in Fig. 1). To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer (step 8 in Fig. 1). As discussed in Section 7, our proposed framework prevents various attacks such as detecting illegal copies of users' data. Note that our work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud. Example 2. Considering Example 1, Alice can enclose her photographs and access control policies in a JAR file and send the JAR file to the cloud service provider. With the aid of control associated logging (called AccessLog in Section 5.2), Alice will be able to enforce the first four requirements and record the actual data access. On a regular basis, the push-mode auditing mechanism will inform Alice about the activity on each of her photographs as this allows her to keep track of her clients' demographics and the usage of her data by the cloud service provider. In the event of some dispute with her clients, Alice can rely on the pull-mode auditing mechanism to obtain log records.

IV. AUTOMATED LOGGING MECHANISM

In this section, we first elaborate on the automated logging mechanism and then present techniques to guarantee dependability.

5.1 Logger Structure

The CIA framework consist a logger component which is basically a nested Java JAR file that stores a user"s data items and corresponding log files. JAR file consists of one outer JAR enclosing one or more inner JARs. Each inner JAR consists of encrypted data, class files to assist retrieval of log files and a log file for each encrypted item. The encryption is done using Weil-pairing-based IBE scheme. Outer JAR consist multiple inner JARs, access policy and class file that authenticates the server or the users and one more class file for finding the correct inner JAR. This enables logger component to handle authentication of entities which want to access the data stored in the JAR file. Logging occurs at any access to the data in the JAR, and new log entries are appended sequentially, in order of creation LR= (r1, r2 . . . rk). Each record ri is encrypted individually and appended to the log file. In particular, a log record takes the following form:
rk = (id, action, T, loc, h((id, action, T, loc)ri-1…r1), sig)
Where,
rk = log record
id = user identification
action = perform on user's data
T = Time at location loc
loc = Location
h((id, action, T, loc)ri-1…r1) = checksum component
sig = Signature of record by server
Checksum of each record is calculated and it is stored with data. Checksum is computed using hash function [37].

B. Log Harmoniser

A log harmonizer has two main responsibilities: first is to deal with copies of JARs and second is to recover corrupted logs. The harmonizer is implemented as a JAR file. It does not contain the user"s data items being audited, but it has class files for both a server and a client processes to allow it to communicate with its logger components. The harmonzer stores error correction information sent from its logger components, as well as the user"s IBE decryption key, to decrypt the log records and handle any duplicate records. Duplicate records result from copies of the user"s data Jars. Here, user"s data are strongly coupled with the logger component in a data JAR file, the logger will be copies together with the user"s data.

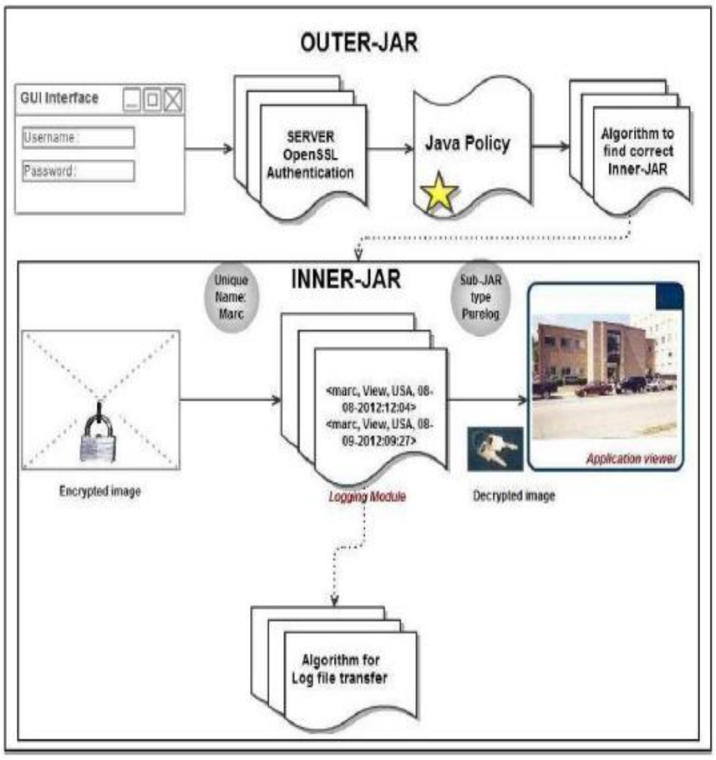
OUTER-JAR
GUI Interface
Username:
Password:
SERVER OpenSSL Authentication
Java Policy
Algorithm to find correct Inner-JAR
INNER-JAR
Unique Name: Marc
Sub-JAR type Purelog
<marc, View, USA, 08-08-2012:12:04>
<marc, View, USA, 08-09-2012:09:27>
Encrypted image
Logging Module
Decrypted image
Application viewer
Algorithm for Log file transfer

Figure. 2. The structure of the JAR file.

C. Generation of Encryption Keys

The traditional method to protect sensitive data from being outsourced to third parties is to store encrypted data on servers, while the decryption keys are disclosed to authorize users only. An identity-based encryption scheme is specified by four randomized algorithms: Set, Extract, Encrypt,Decrypt:

• Set ( ): It take input a security parameter k and returns parameters and master-key. The system parameters has decryption of a finite message space M, and a description of a finite cipher text space C. In this, the system parameters will e publicly known, while the master-key will be known only to the "Private Key Generator" (PKG).

• Extract (): It takes as input parameters, master-key, and an arbitrary ID € {0, 1}*, and returns a private key d.Where ID is an arbitrary string that will be used as a public key, and d is the corresponding private decryption key. Extract Algorithm is used to extracts a private key from the given public key.

• Encrypt (): It takes as input parameters, ID, and M € M. It returns a ciphertext C € C.

• Decrypt (): It takes as input parameters, ID, C € C, and a private key d. It return M € M. Decrypt the message using private key.

V. SECURITY DISCUSSION

We now analyze possible attacks to our framework. We assume that attackers may have sufficient Java programming skills to disassemble a JAR file and prior knowledge of our CIA architecture. We first assume that the JVM is not corrupted, followed by a discussion on how to ensure that this assumption holds true.

A. Attacks on JAR files:

The common attack that we can assume is accessing the data in JAR file without being noticed. This kind of attacks can be found out by auditing. In this, if someone tries to download the JAR files, the actions are recorded by the logger and the log record is sent to the user. By this steps data owner will be have knowledge of his/her JAR file download.

B. Unauthorized user:

If some unauthorized user who don't have permission to access that data, in this first we have to check the his/her integrity by the authentication system before giving the access to actual data. Let us consider third party tries to access the data or hack the data. But he will receive the disassembled Jar file and log record which is encrypted and if he/she need to decrypt it to get the actual data, and also breaking the encryption is computationally complex.

VI. CONCLUSION

It is more important today, to secure unwanted and unauthorized disclosure of their confidential data from the third party. In this paper, the authors have studied and review the security and privacy issues in cloud computing. This paper presents effective mechanism, which performs authentication of users and create log records of each data access by the user. Data owner has ability to audit his content on cloud, and he can get the confirmation that his data is safe on the cloud. Data owner also able to know the duplication of data made without his knowledge. Data owner has secure storage his data on cloud using this mechanism and data usage is transparent, using this mechanism.

ACKNOWLEDGMENT

The authors would like to thanks the Department of Computer Engineering JSPM's College of Engineering & Research, Pune, Indiafor the guidance and cooperation.

REFERENCE

[1] Smitha Sundareswaran, Anna C. Squicciarini and Dan Lin, "Ensuring Distributed Accountability for Data Sharing in the Cloud,", IEEE Transaction on dependable a secure computing, VOL. 9, NO. 4, pg 556- 568, 2012.

[2] B.Crispo and G.Ruffo, "Reasoning about Accountability within Delegation" Proc. Third Int‟l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.

[3] S. Pearson, Y. Shen, and M. Mowbray," A privacy Manager for Cloud Computing," Proc. Int'l Conf. Cloud Computing (cloudcom), pp.90- 106, 2009.

[4] S. Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud, "Proc First Int'l conf. Cloud Computing, 2009.

[5] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, "A Logic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.

[6] A. Squicciarini , S. Sundareswaran and D. Lin, " Preventing Information Leakage from Indexing in the Cloud," Proc. IEEE Int'l Conf. Cloud Computing, 2010

[7] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li,"Enabling public auditability and data dynamics for storages security incloud computing", in INFOCOM.IEEE,2010,pp. 525-533.

[8] C.Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing,"in INFOCOM. IEEE, 2010, pp. 525–533.

[9] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. Int‟l Cryptology Conf.

[10] HP Cloud Website

[11] Advances in Cryptology,pp. 213-229, 2001.B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1993.

[12] http://en.wikipedia.org/wiki/Identity-based_encryption