Design and Implementation of File Authentication System Using Timestamp Mechanism for a Distributed File System

Tetsuya Ogaki,[†] [‡]   Hidenobu Watanabe,[‡]   Tsukasa Iwama,[‡]

Mitsue Den,[‡]   and   Ken T. Murata[‡]

† Graduate School of Information Systems, The University of Electro-Communications

1-5-1 Chofugaoka, Chofu-city, Tokyo, 182-8585 Japan

‡ National Institute of Information and Communications Technology

4-2-1 Nukii-Kitamachi, Koganei-city, Tokyo, 184-8795 Japan

E-mail: † ohgaki@hpc.is.uec.ac.jp

Abstract Gfarm, which is a distributed file system for petabyte-scale data-intensive computing or shared storage for large scale data, is gradually being used as a system infrastructure in science research or business. Although it is a promising file system, it does not provide a function to prevent data from being falsified. We implemented a system of file authentication using a timestamp mechanism to work with Gfarm. This paper reports the way we combined the timestamp mechanism with Gfarm and the results we obtained by evaluating dummy files based on real files in a cloud service that used Gfarm.

Keywords: Distributed file system, cloud security, file authenticity, timestamp authentication

1. Introduction

Gfarm[1] has recently become popular in petabyte-scale data-intensive computing and as the basis of shared storage for large scale data. Gfarm is a global distributed file system and was developed to manage data-intensive computing. It is used as the basis of storage for the High Performance Computing Infrastructure (HPCI)1 and the National Institute of Information and Communications Technology (NICT) Science Cloud2[2]. Research and development on an enterprise Gfarm[3] is also being carried out.

According to a report[4] drawn up by the Ministry of Economy Trade and Industry (METI), one major concern in cloud services is security. Cloud security needs confidentiality, integrity, and availability that are standard requirements to secure information. In addition, it also needs authenticity, accountability, and reliability. Development of accountability[5] is ongoing for Gfarm security, and authenticity and integrity are also being undertaken.

We have focused on authenticity and propose a file authentication system using timestamps[6]. We developed a module (for authentication) in which Gfarm collaborates closely with a timestamp server. Our system enables Gfarm to request file admissions from the timestamp server and to validate stored files immediately to confirm whether these files have been falsified or not. This paper reports the design and implementation of the proposed system, and a method of combining Gfarm and the timestamp server. Section 2 describes Gfarm and how it authenticates files. We explain the file authentication system in Section 3 and the authentication module in Section 4. We demonstrate that collaboration is not a bottleneck from our evaluation of the results in Section 5. Conclusions are drawn and future work is described in Section 6.

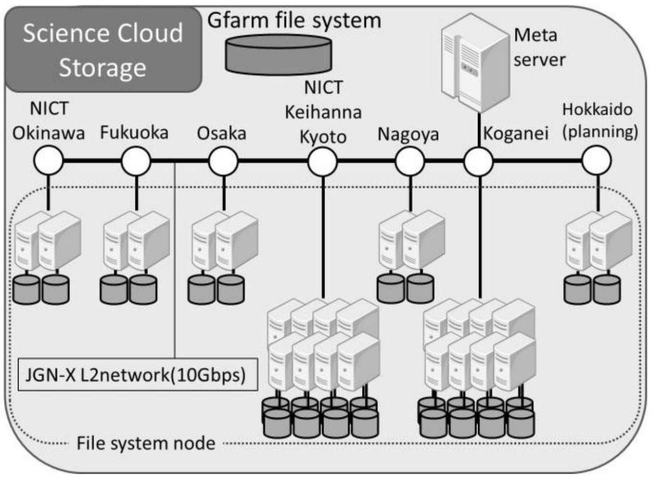

Fig.1 Storage system using Gfarm in NICT Science Cloud.

1 https://www.hpci-office.jp/
2 https://seg-web.nict.go.jp/

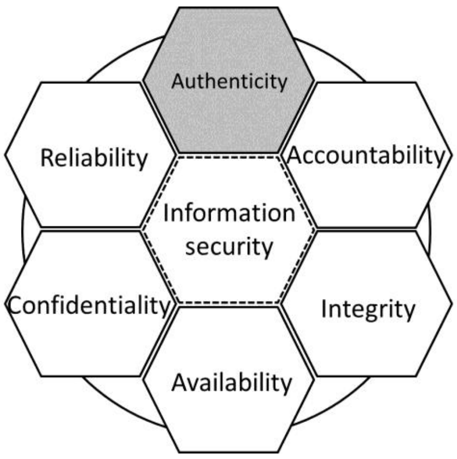

Fig. 2 Requirements for information security.

2. Gfarm file system and authenticity

2.1. Gfarm file system

Gfarm is a global distributed file system that consists of one meta-server and file system nodes (FSNs). The meta-server manages metadata for the files and the FSNs process file operations. Figure 1 outlines a storage system using Gfarm in the NICT Science Cloud.

We can explain the procedure for Gfarm operations using Figure 1. Assume that the client is in Osaka. First, the client accesses the Gfarm meta-server in Koganei when the client creates a file at Gfarm. Next, the meta-server determines the closest FSN using information about network latency between the client's host and the FSNs. Indeed, the closest FSN is the Osaka node in this situation. Then, the client accesses the target FSN and creates a file. Finally, Gfarm returns information that file creation is complete to the client. The process for these file operations are recorded in Gfarm's system log on the meta-server. Therefore, Gfarm achieves high scalability of data I/O by utilizing local disk I/O at FSNs.

Figure 2 shows the requirements for information security especially for cloud services. Gfarm provides a high degree of confidentiality, reliability, and availability using Gfarm functions, which are accomplished through a grid security infrastructure[7](GSI), meta-server duplication, and data replication. In addition, Gfarm also proposes log a trace function for accountability. This function guarantees that no removed files or their copies exist at any FSNs. However, Gfarm does not provide authenticity and integrity because its main purpose is to process data-intensive computing on the peta-byte scale, and it therefore takes responsiveness to users into account.

The NICT Science Cloud provides an online storage service using Gfarm for scientific research as a cloud service. It has a meta-server in Koganei, has 36 FSNs distributed throughout regions in Japan, and stores a great deal of observational data. Our main motivation was to guarantee the authenticity of Gfarm for NICT Science Cloud security, so that we could prevent data from being falsified.

2.2. Authenticity

The access control list (ACL) and write once read many (WORM) are commonly used to control access to data in general cloud services. These technologies are able to prevent data from being falsified before they are accessed. Challenge and response[8], functional encryption[9] and timestamp authentication also guarantee data have not been falsified after they have been rewritten. It is important to routinely check data have not been falsified on the system side to ensure a high degree of authenticity in addition to controlling data access. We chose timestamp authentication from the viewpoint of introducing many examples as the best way of guaranteeing data had not been falsified after having been rewritten.

Timestamp authentication is a public key infrastructure (PKI) technique that incorporates a reliable time source. The timestamp server admits a timestamp token created from the time and the hash value of a file. The timestamp server is managed by the certificate authority (CA) that is legally licensed. If a file was written by a malicious user, it can be proved to have been falsified because of the different hash values of the timestamp token. The timestamp token is the time admitted by CA and guarantees authenticity. NICT is the strictest time calibration organization in Japan. A promising utility value is for the NICT Science Cloud to adopt timestamp authentication.

3. Design of file authentication system

3.1. Requirements for file authentication system

We designed a file authentication system collaborating with a timestamp server and Gfarm. Our system uses a timestamp server (SX-3640TSS), which is a product made by Seiko Precision Inc. This timestamp server provides a timestamp application programming interface (API) that enables the hash value of a file to be admitted to the timestamp server.

It is important to find the FSN storing the target file from the Gfarm's system log and to calculate the hash value of the file to admit a file to a timestamp server using the timestamp API in Gfarm. It is also necessary to manage an admitted hash value (timestamp token) and to compare the present hash value of the target file with the timestamp

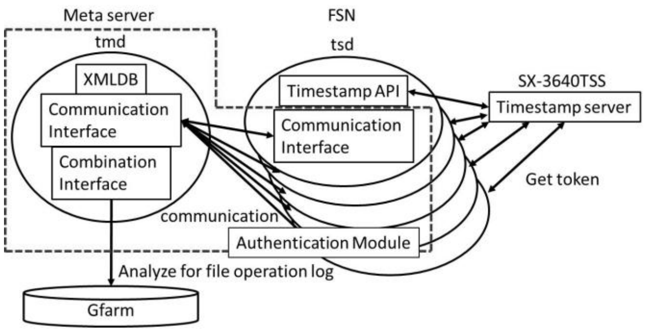

Fig.3 Proposed system architecture.

token to check whether files have been falsified or not. Therefore, the meta-server and FSNs also dynamically need to share authentication information such as timestamp tokens. The five main requirements of a file authentication system are as follows.

(A). Monitor Gfarm system log.

(B). Dynamically exchange authentication information between meta-server and FSNs.

(C). Calculate hash values.

(D). Validate hash values using timestamp token.

(E). Manage timestamp token.

3.2. System architecture

We designed a file authentication system with a timestamp mechanism for Gfarm at the application level. Figure 3 outlines the proposed architecture for the system. It consists of Gfarm and a timestamp server, including the timestamp API and the authentication module that we developed. The module has two resident daemons. The first is a tmd daemon running in the meta-server and the second is a tsd daemon running in each FSN. The tmd has a Combination Interface, a Communication Interface, and an XML database (DB). The tsd has a Communication Interface and the timestamp API provided by the timestamp server (SX-3640TSS).

We developed a Combination Interface to fulfill requirement (A) where it routinely monitors the Gfarm system log of the meta-server, and detects a file path for the created target file and IP address of the FSN storing the file from the log. We also developed a Communication Interface that fulfills requirement (B) where it provides a hash calculation function and a hash validation function of a target file in addition to communication. We considered that the meta-server should not calculate hash values because the server becomes a single failure point in central intensive models that many global distributed file systems including Gfarm are adopting. Therefore, our system enables processing loads of hash calculations in each FSN to

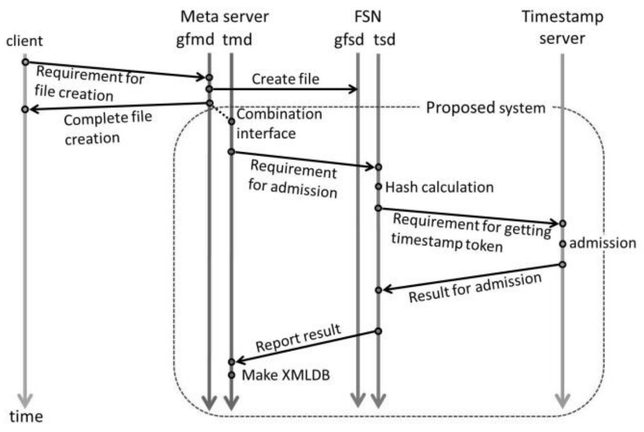

Fig. 4 Process of admitting timestamp in proposed system.

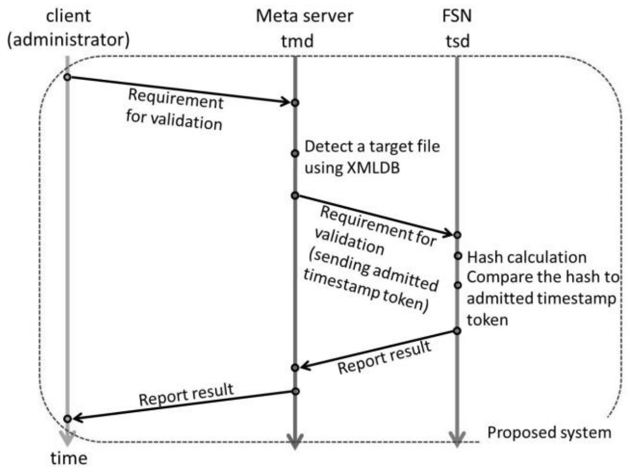
client (administrator)
Meta server tmd
FSN tsd
Requirement for validation
Detect a target file using XMLDB
Requirement for validation (sending admitted timestamp token)
Hash calculation Compare the hash to admitted timestamp token
Report result
Report result
Proposed system
time

Fig. 5 Process of validating timestamp token in proposed system.

be distributed. Requirements (C) and (D) are fulfilled by the interface of tsd. Our system manages the timestamp token using eXtensible Markup Language (XML) to fulfill requirement (E). Gfarm supports extended attributes that can be defined by a system administrator and is also able to manage these in an XML database (XMLDB). Namely, all FSNs can obtain timestamp tokens using an XML function (XPath) at any time and from anywhere.

We will describe the process of admitting and validating timestamps in the following sections.

3.3. Process of authenticating timestamps

We implemented an authentication module. Figure 4 outlines the flow for the process of admitting a timestamp in the proposed system. The gfmd and gfsd are resident daemons of Gfarm. After a file is created in Gfarm, the Combination Interface of tmd recognizes the event from the Gfarm system log of the meta-server and obtains the IP address of the target FSN and the file path. The Communication Interface of tmd sends the information to the

Table 1 Format and example of Gfarm log.

Format	operation_sequence_number/second_from_unix_epoch.microsecond/ global_user/client_ip_or_hostname/client_port/**operation**/ metadb_server_host/metadb_server_port/**gfsd_hostname**/ **inode_number**/inode_generation_number1/ inode_generation_number2///"**pathname1**"///"pathname2"
Log code	Example of output log
CREATE	131908/1345723320.926539/ohgaki/gfarm-meta01/37406/ **CREATE**/gfarm-meta01/601//**29582**/5////"**/home/ohgaki/1.txt**///
REPLICATE	131909/1345723320.927305//// **REPLICATE**/gfarm-meta01/601/**gfarm-fsn01**/**29582**/5///////
UPDATEGEN	131910/1345723320.929230//// **UPDATEGEN**/gfarm-meta01/601//**29582**/5/6//////

Communication Interface of tsd in the FSN for the admission process. Next, tsd calculates the hash value of the file and requests the hash value from the timestamp server using the timestamp API. Finally, the meta-server registers the admitted timestamp token transferred from the FSN in XMLDB. In addition, the file path, IP address of the FSN, and admission time are registered in the DB along with the token.

Figure 5 outlines the flow for the process of validating a timestamp token in the proposed system. The validation involves a falsification check function for the system administrator who needs to easily check all files in Gfarm. If the administrator requests a timestamp token to be validated, the request including the target file path is transferred to tmd by a script program. First, tmd obtains the IP address of the target FSN from XMLDB under the file path. Next, tsd starts calculating a hash value after it receives the request from tmd and compares the hash value with the admitted timestamp token. Finally, tmd reports the result sent from tsd to the administrator, and updates it to XMLDB along with the validation time.

4. Implementation of authentication module

We implemented a Combination Interface, a Communication Interface, and an XML database as the authentication module. We also developed the modules in Java according to a Java-based timestamp API.

4.1. Combination Interface

The Combination Interface needs to obtain the target file path and the IP address of FSN storing the file for the timestamp API. The interface detects a new file has been created from the Gfarm system log. Gfarm defines the format of the system log. Table 1 summarizes the format of a Gfarm system log and gives an example log for file creation.

Figure 6 shows the process for the Combination Interface. When a file is created, Gfarm sequentially outputs "CREATE", "REPLICATE" and "UPDATEGEN" in the

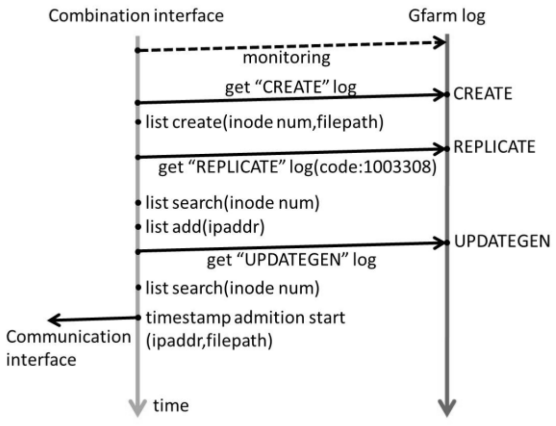

Fig. 6 Process for Combination Interface.

operation field of the format. The CREATE log and REPLICATE log are output when the meta-server determines the closest FSN from the client. The CREATE log has the file path of the target file but does not have the IP address of the FSN. The IP address is included in the gfsd hostname field of the format in the REPLICATE log. The "REPLICATE" log has two types of codes. The first is the "1003308" code which is the code of a master file and the second is the code for duplicated files. At this point, the interface can have the information that the timestamp API needs, but the file has not yet been created. Once a UPDATEGEN log is output, the master and replication file exit. We focused on the inode number included in each log. An inode number is the proper ID of a file. The Combination Interface monitors the operation field of the Gfarm system log. When the CREATE log and REPLICATE log are output, the interface caches the file path from the CREATE log and IP address of the closest FSN from the REPLICATE log in memory as a queue list under the inode number. Then, it waits until the UPDATEGEN log is output. When the interface finds the log, the cased information is passed to the Communication Interface and the process of file admission is started. We used a characteristic of the Gfarm system log where "CREATE", "REPLICATE", and "UPDATEGEN" were sequentially output without failures because of a high-response search that nearly hit a first inode number cached in memory. Of course, cached information was deleted when the process finished normally.

4.2. Communication Interface

We implemented two modes for the Communication Interface. The first was the admission mode and the second was the validation mode. tsd recognizes the type of process from the mode indicated by tmd.

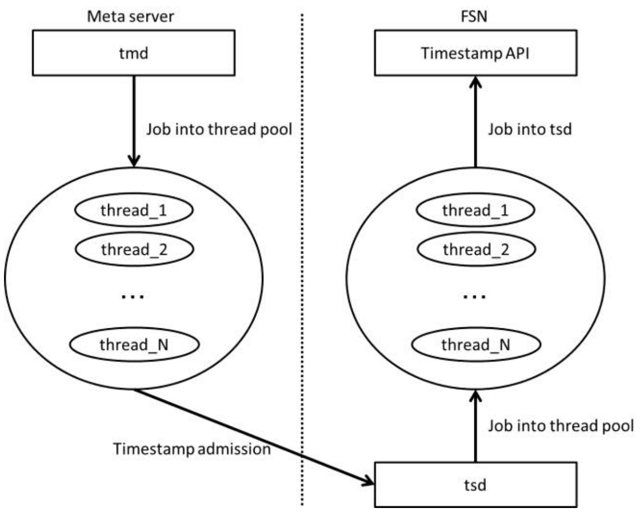

Fig. 7 Thread management between tmd and tsd.

```
<?xml version="1.0" encoding="utf-8"?>
<home>
  <ohgaki>
    <gfarm-disk>
      <file.txt>
        <ipaddr_or_hostname>gfarm-fsn01</ipaddr_or_hostname>
        <TStoken>/usr/local/tmd/recv/gfarm-fsn01_file.txt</TStoken>
        <admition_time>2012_08_30_16:27</admition_time>
        <verification_time>2012_08_30_03:45</verification_time>
      </file.txt>
    </gfarm-disk>
  </ohgaki>
</home>
```

Fig. 8 Data structure of XMLDB.

The Communication Interface needs to establish many connections between tmd and tsds without imposing too much load on machine resources. Our interface creates a Java virtual machine (JVM) thread pool to easily manage connections. Figure 7 outlines the architecture to manage threads between tmd and tsd. The processes for the communication socket are controlled in the thread pool from which a system administrator can define capacity. If there are more connections than can contain, a new thread pool is created for the new connections. Therefore, the interface is generally able to make resources more efficient than those in establishing one-on-one connections.

4.3. XML database

XMLDB manages the target file path, the admitted timestamp token, the IP address for FSN storing the file, the admission time, and the validation time as mapping information. Figure 8 shows the data structure of XMLDB. We implemented a shell script to validate the hash values of numerous files. If the administrator wants to check all files in a directory, what he/she should do is just to run the script with the target directory path in the client machine. The script connects to tmd and transfers the directory path. The tmd recognizes the connection as a validation request and searches the mapping information of all files in the directory path from XMLDB using the "xpath" command. The information is the timestamp token and IP address of FSN. Then, the interface orders hash validation from each FSN sequentially storing the file. The rest of the processes such as hash calculations and hash comparisons are executed at FSNs.

5. Evaluation

5.1. Requirements of evaluation

We evaluated the performance of the Combination Interface in admission processing, and discuss whether the interface is a bottleneck or not.

We used dummy files based on real files stored in the NICT Science Cloud for this evaluation. Examples of real files are global positioning system (GPS) observation data (200 KB/file), visualization processing data (100 MB/file), and simulation data (1 GB/file). The NICT Science Cloud stored 350 observation files/min, 30 visualization files/min, and 30 simulation files/min in three months from January to March in 2012. The two items we evaluated were the processing time and its load.

The processing time consisted of the write time when a file was created in Gfarm, the admission time by the timestamp server, and the Combination Interface time, which is the processing time in our system. We also measured the memory and CPU load of each process using the Linux "top" command while processing admissions.

5.2. Results

Table 2 lists the admission processing time and the breakdown for each data size. These results were the average for five times using dummy files.

Admission processing took ~0.02298 sec/file for 200 KB, ~2.7585 sec/file for 100 MB, and ~81.7023 sec/file for 1 GB. The percentage for the Combination Interface was ~0.6 % for 200 KB, ~0.006 % for 100 MB, and ~0.0002 % for 1 GB. These percentages mean that the Combination Interface processing time is less than 0.0002

Table 2 Admission processing time and its breakdown.

Data size	Write time (sec/file)	Admission time(sec/file)	Combination Interface time(sec/file)	Total time (sec/file)
200KB	0.008581	0.01426	0.0001417	0.02298
100MB	1.5233	1.2350	0.0001567	2.7585
1GB	26.5592	55.1429	0.0001839	81.7023

sec/file for any data size.

We also measured the memory and CPU load for the Combination Interface while processing admissions. The utilization of virtual memory was about 3.9 MB, and the physical utilization was about 500 KB. CPU utilization was nearly equal to 0 % for any data size.

These results indicate that the Combination Interface does not create overhead. The reason for this is that the Combination Interface nearly hits the first element of a short search.

6. Conclusion

We designed a file authentication system using timestamps to ensure authenticity in Gfarm and implemented a Combination Interface. The authentication module built in the meta-server and FSNs monitored the Gfarm log, managed timestamp tokens, and carried out falsification validation including hash calculations.

The Combination Interface for monitoring and analyzing the Gfarm system log was able to process at less than 0.0002 sec/file regardless of the data size, and the CPU and memory loads were very small.

In future, we intend to evaluate our proposed system in detail and implement it in the NICT Science Cloud.

Acknowledgements

This work was conducted by using the machine resources of One Space Net (ONS) in the Science Cloud service.

Bibliography

[1] O. Tatebe, K. Hiraga, and N. Soda, "Gfarm Grid File System", New Generation Computing, Ohmsha, Ltd. and Springer, Vol. 28, No. 3, pp. 257–275, 2010.

[2] K. T. Murata, S. Watari, T. Nagatsuma, M. Kunitake, H. Watanabe, K. Yamamoto, Y. Kubota, H. Kato, T. Tsugawa, K. Ukawa, K. Muranaga, E. Kimura, O. Tatebe, K. Fukazawa, and Y. Murayama, "A Science Cloud for Data Intensive Sciences", Proc. of the 1st ICSU WDS Conference on Data Science Journal, Kyoto, Japan, Sep. 2011, in press.

[3] BEST SYSTEMS, "GfarmFSE", http://www.bestsystems.co.jp/solutions/filesystems/gfarm.html

[4] Ministry of Economy, Trade and Industry, Society for the study of dependability and security of information system software in an advanced information society , "Strengthening Efforts to Enhance Dependability and Security of Information System Software ~Toward a rich, safe/secure advanced information society~ -Interim Report-", Ministry of Economy, http://www.meti.go.jp/english/press/data/20090528_01.html, May.28,2009.

[5] K. Onishi, W. Hasegawa, S. Takeuchi and H. Takasugi, "Design, Implementation and Evaluation of Data Traceability on a Large-scale Global Distributed File System for Cloud Computing", IPSJ SIG Technical Report, HPC-130(35), pp. 1–8, 2011.

[6] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, IETF (2001).

[7] R. Butler, V. Wlch, D. Engert, I. Foster, S. Tuecke, J. Volmer, and C. Kesselman, "A National-scale Authentication Infrastructure", IEEE Computer Society Press, Vol. 33, No. 12, pp. 60–66, 2000.

[8] M. Backes, A. Cortesi, R. Focardi, and M. Maffei, "A calculus of challenges and responses," Proc. 5th ACM workshop on formal methods in security engineering, pp. 51–60, Fairfax, Virginia, USA, Nov. 2007.

[9] A. Sahai and H. Seyalioglu, “Worry-free encryption: Functional encryption with public keys”, Proc. 17th ACM conference on computer and communications security, pp. 463–472, Chicago, Illinois, USA, Oct. 2010.