VOL. 11, NO. 22, NOVEMBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2016 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

DEFENSE AGAINST CACHE-BASED SIDE CHANNEL ATTACKS FOR SECURE CLOUD COMPUTING

Munish Chouhan and Halabi Hasbullah
Department of Computer and Information Sciences, Universiti Teknologi Petronas, Perak, Malaysia
E-Mail: munish4111989@gmail.com

ABSTRACT

Cloud computing is a combination of various established technologies like virtualization, dynamic elasticity, broad band Internet, etc. to provide configurable computer resources as a service to the users. Resources are shared among many distrusting clients by abstracting the underlying infrastructure using virtualization. While cloud computing has many practical benefits, resource sharing in cloud computing raises a threat of Cache-Based Side Channel Attack (CSCA). In this paper a solution is proposed to detect and prevent guest Virtual Machines (VM) from CSCA. Cache miss patterns were analyzed in this solution to detect side channel attack. Notification channel between client and cloud service provider (CSP) is introduced to notify CSP about the consent of client for running the prevention mechanism. Cache decay mechanism with random decay interval is used as a prevention mechanism in the proposed solution. The performance of the proposed solution is compared with previous solutions and the result indicates that this solution possess least performance overhead with a constant detection rate and compatible with existing cloud computing model.

Keywords: security, attack, cloud computing, cache-based side channel attack, cache usage analysis, cache access pattern.

INTRODUCTION

Cloud computing provides configurable computer resources such as software, operating systems, storage, computing units, etc as a service to customer (Mell & Grance, 2009). It also brought software and data from client datacenter to cloud. Cloud model is referred to a system setting where any type of software can run on any hardware architecture (canonical software and hardware), which means client should not have to change application code or configuration according to the hardware architecture of the cloud (Godfrey & Zulkernine, 2013).

Cloud computing has three service models (Savolainen, 2012) those are Software as a service (SaaS), Platform as a Service (PaaS), and Infrastructure as a service (IaaS). First service model provides software access to client through thin client like browser or a thick client like some program interface; some of the software is email, CRM, Billing, and rating system etc. Client can also change some of the application configuration settings. In Paas user can install any application on the Operating System (OS) or other platforms provided on cloud. In Iaas user can access any computer resource like network, memory, storage, and install any platform on it. Cloud has four deployment models and those are private, community, public, and hybrid. Private cloud is deployed exclusively for one organization, community cloud is also same as first but it is for a specific community (group of consumers sharing same policies, security concerns, etc.). Public cloud is deployed for general public where all the resources are shared and at last hybrid is the combination of any two deployment models from the first three (Mell & Grance, 2009). Hybrid deployment model comes into play due to insecurity of client about hosting application or storing data in a shared environment. In this deployment model customer needs private cloud inside public cloud, so that client data or application can be accessed from anywhere, but is not sharing cloud resources with other clients (Almorsy, Grundy, & Müller, 2010). Side channel attacks are one of the reasons behind this insecurity.

Cloud computing security become important with the popularity of cloud in the computer industry (Stone & Vance, 2010). Now clients are moving from cloud computing to secure cloud computing. Cloud computing is also complicated as new features have been added like multi-tenancy and elasticity (Almorsy *et al*., 2010), (Chen, Paxson, & Katz, 2010). Multi-tenancy (configurable entities shared between different customers) acceptance also raised many security concerns about security (Zissis & Lekkas, 2012). Resource sharing results in side channel attack, these attacks mainly observe execution time, memory latencies, and power consumption by processor when a particular encryption program is running (Zhou & Feng, 2005).

Computer has many benefits from cache, but there is one drawback, that is increase in the coupling between program cache usage pattern and program input, therefore cache become vulnerable to side channel attacks (Crane, Homescu, Brunthaler, Larsen, & Franz, 2015). This drawback of cache is the basis of CSCA. Modern computer architecture mainly use L3 shared cache and this shared cache is used to establish a side channel attack by (Kim, Peinado, & Mainar-Ruiz, 2012) as shown in Figure-1 where attacker is reading the cache hits and misses by analyzing the time difference between cache access and memory access (during cache miss) using RTDSC (Time Stamp Count)and construct a substantial information from the victim like encryption keys (Zhang, Juels, Reiter, & Ristenpart, 2012). In these types of attacks, attacker accesses the timer or observes the cache usage pattern of victim's VM.

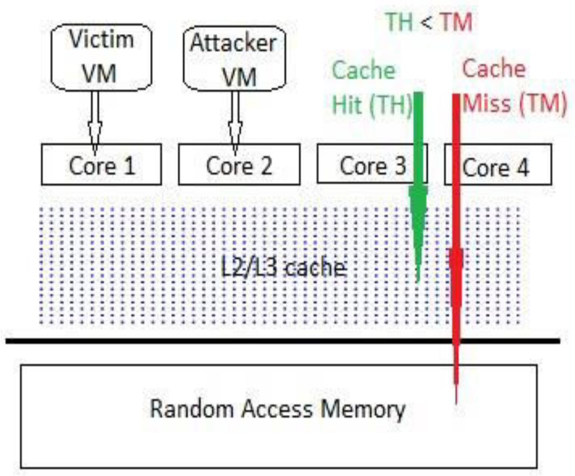

Figure-1. Cache-based side channel attack using shared cache.

Cloud clients use Internet to access configurable resources of cloud. Secure sockets layer and other encryption based protocols are used for the communication over Internet, this makes Cloud computing security is dependent on encryption methods like RSA and AES. These encryption methods are mathematically strong but lacks in term of the relationship between their execution behavior and computer resource usage like power consumption and cache usage. This weakness of encryption methods is exploited by CSCA attackers.

Generally CSCA is divided into two types (Kim *et al*., 2012) as shown in Figure-2 and those are time driven cache attacks, and trace driven cache attack.

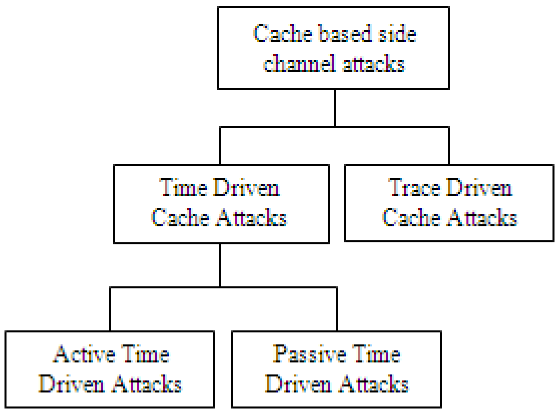
Cache based side channel attacks
Time Driven Cache Attacks
Trace Driven Cache Attacks
Active Time Driven Attacks
Passive Time Driven Attacks

Figure-2. Types of cache-based side channel attacks.

1) Time driven cache attacks: Attacker analyses the sum of execution time of a cryptographic process and the variations in the execution time of a security critical program running on victim's VM with different inputs. Time driven CSCA is further divided into two types on the basis of attacker's access to the victim's cache.

a) Active time driven attacks: Attacker has access directly to the cache and timer. It can alter the state of cache which might result in collisions with the victim's cache lines. It also measures the execution time of interested program with high precision.

b) Passive time driven attacks: Attacker cannot probe or alter the victim's cache and also cannot access the timer. Attackers mainly accesses cache remotely and thus there is noise due to network delays.

2) Trace driven cache attacks: Attacker analyses the usage pattern of cache line by the victim. This attack also requires the local sharing of cache between both VMs. As in this attack, attacker has more information about the victim's application behavior, which makes this attack method more dangerous than previous attacks.

Many developments and research works on cloud computing security is going on in many institutions and companies to create a secure cloud (Ryan, 2013). Most of the researchers used a common cloud setting as shown in Figure-3 for their experiments (Godfrey & Zulkernine, 2013), (Kim *et al.,* 2012), (Zhang, Juels, Oprea, & Reiter, 2011). It consists of two or more virtual machines and a shared top level cache (L2 or L3 cache) between them. One virtual machine is behaving like an attacker and others as victim.

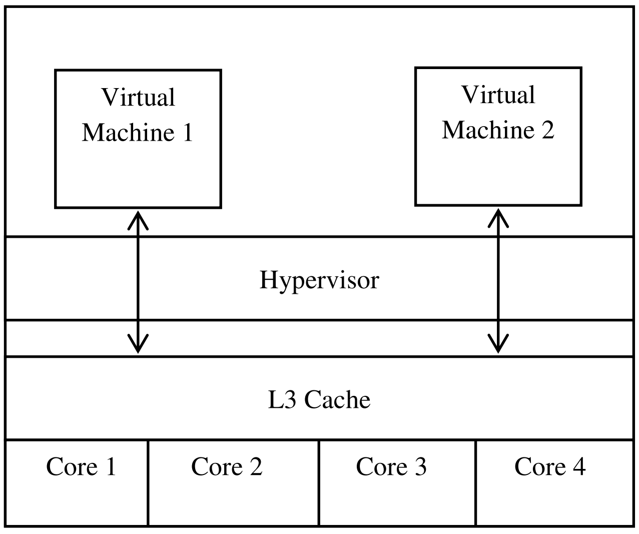

Figure-3. Virtual machines with shared cache in cloud.

To maintain security in cloud without affecting its performance, cloud service provider needs to deploy solutions which can secure the cloud from CSCA problem with less overhead and without violating cloud model (Ryan, 2013). Mainly CSCA are carried out for extracting encryption keys and security related information, therefore strong detection and prevention system is required for securing client's VM from CSCA. Previous research work shows the extraction of encryption key of the program running on victim's VM by using CSCA (Zhang *et al.*, 2012), (Ristenpart, Tromer, Shacham, & Savage, 2009) and CSCA is also carried out in platform as a service model and it is already possible in infrastructure as a service model as described in (Zhang, Juels, Reiter, &

Ristenpart, 2014), which makes it more important to find a suitable solution for this problem.

The rest of the paper is organized as follows: Firstly it describes the problem statement of this research and after that briefly describes the background of already existed solutions for the defense against CSCA and then the methodology of this research. Lastly results are discussed and conclusion is presented.

PROBLEM STATEMENT

Cloud computing is economical as compared to the traditional computing because resources are shared between different clients. Resource sharing is achieved by using virtualization in cloud, where multiple virtual machines are sharing same physical machine known as multi-tenancy. This sharing raises the threat of information leakage from victim's VM to attacker through CSCA. Clients are resistant to multi tenancy due to risk of CSCA and demand for separate physical machines (Stone and Vance, 2010), but if everyone wants their own physical machine, then there is no dissimilarity between cloud and private datacenter (private datacenter wastes computing power and storage capacity). Security in computer society required many assurances, but defense against unexpected attacks like CSCA is of utmost importance (Zissis and Lekkas, 2012).

In CSCA attacker analyses the cache usage pattern of another VM and generates substantial information (like encryption keys) about the application running on victim's VM. In previous solutions there are either changes needed in client's application (Shi, Song, Chen, & Zang, 2011), (Crane *et al*., 2015), (Kim *et al*., 2012) or changes in cloud hardware which violate cloud model (Fletcher, 2013). Some of the solutions have less overhead, but not compatible with cloud model and some are compatible but have high performance overhead. To address this problem, this research work proposed a defense method against cache-based side channel attacks for maintaining the security in cloud.

LITERATURE REVIEW

This section presents the study of various research papers and journals related to cache-based side channels attacks CSCA and proposed solutions for CSCA. Some of the papers are discussed and summarize here.

CSCA performed by Bernstein in (Bernstein, 2005) exploits the collisions between several table lookups of the cipher to recover the security key of an encryption program. This attack has three phases:

a) Learning phase: Assume that the CSCA attacker knows the encryption key of encryption program running on victim's VM. In this phase attacker sends random plaintexts to the cipher program and record the encryption time for these plaintexts.
b) Attacking phase: Encryption time for different plaintext is measure in this phase same as in (a). In this phase unknown encryption key is used.
c) Key recovery: Encryption timing sets of previous two phases is compared to recover the unknown key.

CSCA detection technique proposed by (*Zhang et al.,* 2011) called "HOMEALONE", that uses side channel to detect the presence of attacker's activity in its physical machine by analyzing the cache usage. HOMEALONE is a client oriented solution, which showed the lack of security by cloud service provider. This solution monitors a small part of cache (approx. 1/16th part) and therefore results in less overhead on processing time. Another detection technique proposed by (Yu, Gui, & Lin, 2013) named CSDA, which analyzes the resource utilization to detect CSCA. It consists of two stages, first one is Host detection, it uses shape test by observing cache miss sequence. Second one is guest detection, it uses regularity test by analyzing the virtual CPU and virtual memory utilization sequence. The detection rate of this solution is moderate but with zero false negative rate, which make this detection solution good for secure cloud computing. Analyzing resource utilization can help in detecting CSCA as used in these detection techniques directly or indirectly. In case of CSCA, monitoring cache utilization can lead to a better detection technique.

CSCA prevention technique proposed by (Shi, Song, Chen, and Zang, 2011) protects security-critical operations of application running on guest VM from attackers by dynamically partitioning of cache using page coloring. Application notifies the VM manager and then VM manager allocate a secure space in cache to the data of that security critical application. Reserved cache lines are exclusively allocated to the security sensitive application and no one else can access it, which makes cloud secure to a certain level as still attacker can analyze the timer for that particular program execution or the power consumption. Another research work by (Kim, Peinado, and Mainar-Ruiz, 2012) proposed a system-level defense method for CSCA in the cloud. It reserves a cache set per core, never remove it from cache, and each VM can share the reserved cache set for its sensitive data. Therefore attackers cannot read memory access patterns of other VMs, which make it difficult to establish CSCA. In modern computers exclusive L1 cache for every core is already exist and if the same is done to the higher level cache then there is no advantage of using more than one level of cache as it contains the same data present in lower levels of cache.

Next two prevention techniques by (Godfrey and Zulkernine, 2013) and (Crane, Homescu, Brunthaler, Larsen, and Franz, 2015) require changes in the hypervisor and an addition of a lower level VM respectively. These solutions are compatible and can be implemented in the modern cloud. M Godfrey *et al.* in (Godfrey and Zulkernine, 2013) classified CSCA into two types: Sequential CSCA, and Parallel CSCA attacks based on their implementations and proposed prevention techniques for both. Selective cache flushing is used against sequential CSCA, where cache will be flushed in between prime and trigger steps (to know more about prime probe trigger method please refer (Wu, Xu, and Wang, 2013)) if there are chances of CSCA. The presence of CSCA in physical machine is checked by using taint checking. Cache partitioning is used against parallel CSCA, where cache sets has been divided into different VMs and they only access them for computation. Although the solution proposed by S Crane *et al.* in (Crane *et al.*, 2015) randomizes the execution flow of program. Many replicas of software has been generated, their execution paths are different but semantically same. At runtime application was randomly switching between these generated replicas, which make it difficult for attacker to analyses the behavior of application. These two solutions possess considerably high performance overhead. The implementation of later solution is complicated and requires many changes at client as well as cloud end.

A hardware based solution for preventing CSCA is proposed by (Fletcher, 2013). This solution contains secure processor architecture with oblivious RAM techniques to fleece memory access patterns and differential power analysis resistance technique to hide data-dependent power draw and results in the prevention of privacy leakage. Memory access patterns and variation in power is also considered in side channel attacks. This solution is designed at hardware level and therefore it has least effect on the performance of cloud, but the implementation of this solution is expensive, as practically whole cloud infrastructure need to be changed for this solution.

Table-1. Related work.

References	Strengths	Limitations	Remarks
(Zhang *et al.*, 2011)	• Detects presence of attacker VM using side channel attack.	• Client need to install detector in their VM.	This solution is client dependent; it will be more effective if this solution is at server side.
(Shi *et al.*, 2011)	• On demand Cache isolation for any application.	• Need changes in client's program. .	This solution can be used if it is cloud oriented (only requires changes in cloud software).
(Kim *et al.*, 2012)	• Cloud oriented solution (only requires changes in cloud). • Few changes required in hypervisor.	• Requires changes in client's program. • Reserving cache lines per core.	This solution has reasonable performance overhead, but it is affecting the actual meaning of higher level of cache.
(Godfrey & Zulkernine, 2013)	• Solution required changes in hypervisor only.	• High performance overhead. • Cache flush will increase the cache miss rate.	The overhead can be decreased if a solution contain detection method also which restrict the cache flushing to the most susceptible cases.
(Yu *et al.*, 2013)	• Detection rate of 60%. • False negative rate 0.	• High false positive rate. • High performance overhead.	This solution can be used in cloud, if the performance overhead is less.
(Fletcher, 2013)	• Totally eliminate the possibility of side-channel attacks.	• Hardware need to be changed. • New cloud infrastructure required.	This is a hardware based solution, which makes it difficult to implement on cloud.
(Crane *et al.*, 2015)	• Applicable to every type of CSCA.	• Need some manual changes. • Might not work for all client programs.	This method requires some changes in the client's application and in case of security sensitive application; client might be reluctant for the changes.

Previous works on defense against CSCA are compared in Table-1. Some solutions were violating cloud model (either required change in cloud hardware or in client application). Some solutions are compatible with today's cloud infrastructure, but possess considerably high performance overhead. From this review, there is an opportunity to research on a cloud compatible solution with less overhead for defense against CSCA for secure cloud computing.

METHODOLOGY

This research proposed a solution to detect and then prevent client's security critical program from CSCA. The proposed solution is based on a cloud setting with two VMs installed on a same physical machine using baremetal hypervisor sharing highest level cache i.e. L3 cache as present in Iaas cloud as shown in Figure-3. To overcome CSCA, this research proposed a solution, which is more compatible with cloud model and with less performance overhead. This technique has three steps as shown in Figure-4:

1. CSCA Detection in cloud by analyzing cache misses patterns.
2. Check if VM using affected cache (affected by CSCA) requires security.
3. CSCA prevention by introducing noise in cache.

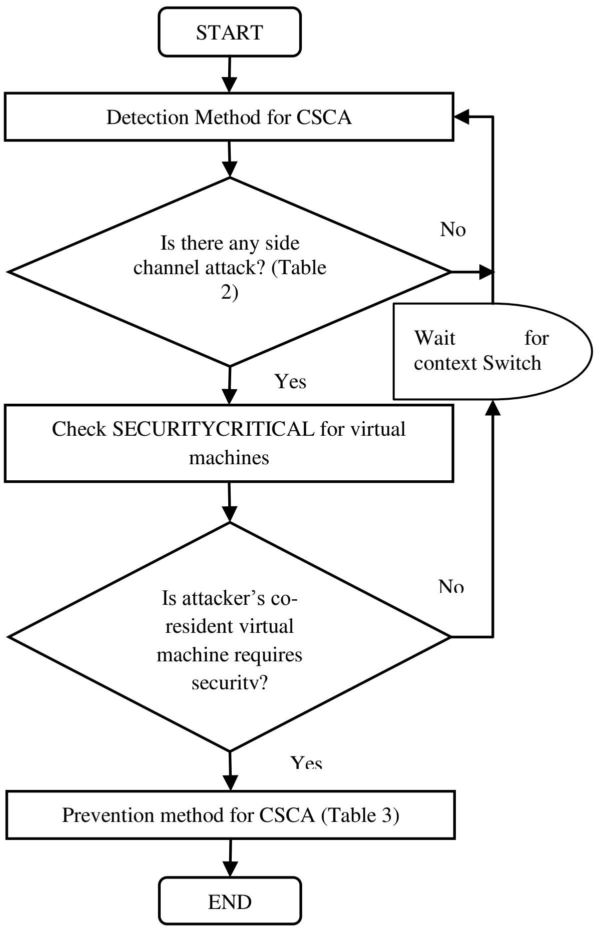
START
Detection Method for CSCA
Is there any side channel attack? (Table 2)
No
Yes
Wait for context Switch
Check SECURITYCRITICAL for virtual machines
Is attacker's co-resident virtual machine requires security?
No
Yes
Prevention method for CSCA (Table 3)
END

Figure-4. The proposed solution.

Initially, computeCahceMisses(T) function in detection algorithm presented in Table-2 uses cache profiler tool to analyze the cache usage patterns and returns the cache miss sequence (CMS) for the time interval T. Then CMS is compared with already computed data about cache miss sequence during CSCA represented as CCMS in this paper. This CCMS is computed by repeatedly executing CSCA on test environment and record the sequence of cache misses at the time of attack. Positive result of the comparison between CMS and CCMS indicates the presence of CSCA.

Table-2. Detection algorithm.

```
cacheStateChanged(){
            CMS = computeCahceMisses(T);
            IF (CCMS.isContains(CMS))
                    preventCSCA();
            else
          Return;                }
```

After detection, prevention algorithm in Table-3 verifies if any VM running on affected cache is executing any security critical application. For Security critical verification, a notification channel is introduced between client and cloud service provider for notifying cloud service provider that client application needs security from CSCA.

Table-3. Prevention algorithm.

```
preventCSCA( ) {
if (isSECURITYCRITICAL() == true){
introduceNoise();           }
Else{
wait(cachestatechanged);
cacheStateChanged();        }
}
```

A file is maintained in cloud for storing the response of clients received via notification channel. These responses indicate whether client's VM is running any encryption program and require security against CSCA. These responses are in the form of a value of parameter SECURITYCRITICAL for every VM present in the cloud. False value of parameter SECURITYCRITICAL halts the detection process until another VM is not using the affected cache. True value of SECURITYCRITICAL (can be checked by using IsSECURITYCRITICAL() function) enables prevention mechanism to introduce noise (as shown in Algorithm 2) in the cache by using the mechanism of cache decay (Kaxiras, Hu, & Martonosi, 2001). A random function is used to generate random delay intervals for cache decay mechanism to randomly turn off and on cache lines. Now attacker cannot identify the cause of cache miss is because of eviction of data or cache decay (Keramidas, Antonopoulos, Serpanos, and Kaxiras, 2008).

Performance overhead (represented as *P* in the below Equation (1)) is an additional expense occurring in addition to normal cost. In this research, it is the variation in processing time of CPU when proposed solution is running in the cloud.

$$P = \frac{\textit{Processing time with proposed system by VM}}{\textit{normal processing by VM}} * 100 \qquad (1)$$

Existing solutions has continuous performance overhead as shown in Equation (2), as the changes required are either static or manipulate cache at continuous intervals and performance overhead is calculated as follows:

$$X = NY \tag{2}$$

where X is total overhead and N is the total number of times cache state changed. N can be number of times encrypted program executes on client's VM or number of context switches or any another parameter used to satisfy the condition for running the protection method. Y is performance overhead generated by single run of protection method.

Now in the case of proposed solution the performance overhead is calculated as follows:

$$X = nY + Z \tag{3}$$

where $n<N$ and n is the number of times CSCA detected and Z is the performance overhead by detection mechanism. The value of Z in the equation (3) is very less as compared to the difference between n and N, as the proposed detection mechanism only consists of a single comparison as key operation. In this solution the protection mechanism executed after CSCA is detected.

RESULTS

The proposed solution protects client's VM from CSCA and is compatible with cloud model with less processing overhead. As prevention technique will add some overhead on the victim's VM, but it is very less as the prevention technique is executed once the attack is detected. The protection mechanism in the proposed solution is introduces noise in cache by using cache decay as that in (Crane *et al*., 2015) where noise is introduced by using software diversity. This leads to an average overhead of 2%. After observing the cache access patterns and analyzing different CSCA detection techniques, it is concluded that the proposed detection mechanism added 1% to 2 % of performance overhead. Therefore, the expected average performance overhead is 2% to 3%. Notification channel further reduces the performance overhead by restricting cloud provider to run the prevention technique only when the client sends a positive response through notification channel. Comparison of performance overhead in Figure 5 clearly shows that the performance of proposed solution is least as compared to existing solution. This solution will help cloud service providers to maintain security as well as multi-tenancy without affecting cloud performance to achieve secure cloud computing.

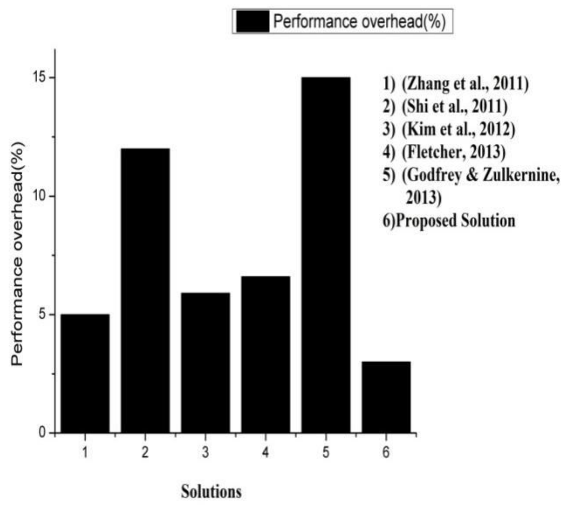

Figure-5. Comparing performance overhead of existing and proposed solution.

Detection rate of proposed solution depends upon the preciseness of the information about the CMP at the time of CSCA. As mentioned in (Yu *et al.*, 2013), proposed solution also posses detection rate more than 60% and detection rate increases with the increase in information about the CMP at the time of CSCA. More information about CMP increases the probability of deciding whether the current CMP is due to CSCA or not. The accuracy of deciding whether the current CMP is a due to CSCA is directly proportional to the accurate information about the CMP at the time of CSCA. This accurate information is collected by repeating the CSCA on cloud and analyzing the cache usage patterns.

Comparison of Detection rate in Table-4 clearly shows that the proposed solution have the highest detection rate with no variation as it is in (Zhang *et al.*, 2011). Proposed solution has no variations as it is dependent on the cache miss patterns and not on any particular hypervisor or operating system. The detection rate increases with the collection of more accurate information about CMP. Therefore with the increasing detection rate and no variation makes proposed solution more efficient than other solutions.

Table-4. Comparing detection rate.

Solution	Detection Rate
(Zhang *et al.*, 2011)	15% - 85%
(Yu *et al.*, 2013)	60%
Proposed Solution	>60%

CONCLUSIONS

Cloud computing popularity and threats due to multi-tenancy motivated this research work to make cloud computing more secure. CSCA is one of the dangerous attacks in this category and is the prime concern of this work. This attack is dangerous because it mainly used for stealing encryption keys and today's Internet mostly use encryption based security techniques like secure socket layer protocol. To resolve this problem, a solution is proposed to detect and prevent the CSCA attack, which outperformed 2% better than existing solutions. Detection method is based on the cache miss patterns at the time of CSCA and prevention technique is primarily making attacker blind by introducing noise in the cache. In detection mechanism, with the increase in the amount of precise data about CMP at the time of CSCA also increases the number of comparisons. These comparisons can be resolved by separating more frequent CMP form others. A notification channel has been introduced between client and cloud service provider to minimize the performance overhead by this solution. This solution only required changes in Hypervisor and host operating system and no changes are required in cloud model. This solution will help cloud service providers to maintain security as well as multi-tenancy for clients to run their security critical applications in a secure cloud.

ACKNOWLEDGEMENT

The authors would like to acknowledge the management of Universiti Teknologi PETRONAS, Malaysia for providing financial support to this research work under postgraduate assistance scheme.

REFERENCES

Almorsy, M., Grundy, J. and Müller, I. 2010. An analysis of the cloud computing security problem. Paper presented at the Proceedings of APSEC 2010 Cloud Workshop, Sydney, Australia, 30th Nov.

Bernstein, D. J. 2005. Cache-timing attacks on AES.

Chen, Y., Paxson, V. and Katz, R. H. 2010. What's new about cloud computing security. University of California, Berkeley Report No. UCB/EECS-2010-5 January, 20(2010), 2010-2015.

Crane, S., Homescu, A., Brunthaler, S., Larsen, P. and Franz, M. 2015. Thwarting Cache Side-Channel Attacks through Dynamic Software Diversity.

Fletcher, C. W. 2013. Ascend: An architecture for performing secure computation on encrypted data. Citeseer.

Godfrey, M. and Zulkernine, M. 2013. Preventing Cache-Based Side-Channel Attacks in a Cloud Environment.

Keramidas, G., Antonopoulos, A., Serpanos, D. N. and Kaxiras, S. 2008. Non deterministic caches: A simple and effective defense against side channel attacks. Design Automation for Embedded Systems, 12(3), 221-230.

Kim, T., Peinado, M. and Mainar-Ruiz, G. 2012. STEALTHMEM: System-Level Protection against Cache-Based Side Channel Attacks in the Cloud. Paper presented at the USENIX Security symposium.

Mell, P. and Grance, T. 2009. The NIST definition of cloud computing. National Institute of Standards and Technology, 53(6), 50.

Ristenpart, T., Tromer, E., Shacham, H. and Savage, S. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Paper presented at the Proceedings of the 16th ACM conference on Computer and communications security.

Ryan, M. D. 2013. Cloud computing security: The scientific challenge, and a survey of solutions. Journal of Systems and Software, 86(9), 2263-2268.

Savolainen, E. 2012. Cloud service models. Paper presented at the em Seminar--Cloud Computing and Web Services, University of Helsinki, Department of Computer Science, Helsinki.

Shi, J., Song, X., Chen, H. and Zang, B. 2011. Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring. Paper presented at the Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on.

Stone, B. and Vance, A. 2010. Companies slowly join cloudcomputing. New York Times, 18, 2010.

Yu, S., Gui, X. and Lin, J. 2013. An approach with two-stage mode to detect cache-based side channel attacks. Paper presented at the Information Networking (ICOIN), 2013 International Conference on.

Zhang, Y., Juels, A., Oprea, A. and Reiter, M. K. 2011. Homealone: Co-residency detection in the cloud via side-channel analysis. Paper presented at the Security and Privacy (SP), 2011 IEEE Symposium on.

Zhang, Y., Juels, A., Reiter, M. K. and Ristenpart, T. 2012. Cross-VM side channels and their use to extract private keys. Paper presented at the Proceedings of the 2012 ACM conference on Computer and communications security.

Zhang, Y., Juels, A., Reiter, M. K. and Ristenpart, T. 2014. Cross-Tenant Side-Channel Attacks in PaaS Clouds. Paper presented at the Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.

Zhou, Y. and Feng, D. 2005. Side-Channel Attacks: Ten Years after Its Publication and the Impacts on Cryptographic Module Security Testing. IACR Cryptology ePrint Archive, 2005, 388.

Zissis, D. and Lekkas, D. 2012. Addressing cloud computing security issues. Future Generation Computer Systems, 28(3), 583-592.