International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.3, No.10, October 2014 DOI:10.15693/ijaist/2014.v3i10.60-66

Performance Enhancement of PDP in Location-based Services

Abdullah M. Al-Shomrani.
Ph.D. Student/Department of CS
King Abdul-Aziz University, Saudi Arabia

Jonathan Cazalas
Assistant Professor/Department of CS
King Abdul-Aziz University, Saudi Arabia

Abstract— As location based services become ubiquitous, protecting the privacy of users is paramount to success. The efficiency of most privacy protection approaches need enhancement to adopt the growth of number of users, with one such approach being the usage of XACML policy. Performance of the XACML policy for LBS needs improvement due to large number of policies and attributes. In our work we present two mechanisms to enhance the XACML performance. First, we present policy hash index for policies. Second, we present policy cache. These two mechanisms considerably improve the performance of the XACML polices.

Index terms - privacy, location-based services, PDP.

I. INTRODUCTION

Technology has effectively integrated into our lives with the introduction of different types of technology, including GPS, RFID and others. Smart phone applications are increasing every day, which make peoples' lives more convenient [1]. These technologies adopt Location Based Services (LBS) to provide services to users. LBS are information services that provide users with certain requested information retrieved from a specific spatial database. The main use of LBS is to supply users with location information. Providing users with added value to more location information is a complex task. LBS service is informative and entertaining based on the usage [2]. There are many services that can be utilized by using LBS. Examples include trace my luggage, find the nearest police station, or monitor my child location to keep an eye on him [3].

Also, track-the-doctors service is active on the hospital. This service allows an operator to track the position of the doctors for management and emergency purposes. Find-my-friends service is a social service that informs users about their friends. Track a critical item for military purpose is another service. LBS provide great service for users. Adopting all technology to be used and utilized for the benefit of the service customers. Within the smart phone all functionality and service can be used. Stolen car is another example of the service. These requests contain secure information for the requestors, including the current location of the user as well as user interests or habits.

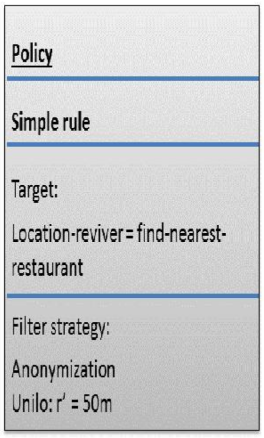
This information can be used in a good or bad attention. However, LBS introduced privacy issues for location privacy due to the information transmitted over the open network [2]. Different usage needs different privacy protection. There are very critical applications, which need more protection and high security policies. The Location Based Service (LBS) privacy approach uses different practices to deal with privacy issues. This compromised the privacy of location based services. That does not just affect the privacy but it threatens their lives. Users fell that they are not secure and their movement can be seen by other people. [4] Studied shows that the benefit of the LBS still not up to the threat that cause for people privacy. Policy evaluation needs more investigation to overcome efficiency problems [5]. Most of the previous work focuses on the correctness of the data rather than the performance [6].

Privacy policy needs to be integrated, flexible, context-aware and customizable [7]. Many privacy solutions have been proposed to protect the user's privacy. One proposed solution is to use integration of privacy protection mechanism [3]. In most studied it concentrate in one mechanism while this studied focus on applying more than one mechanism to secure user privacy. XACML (eXtensible Access Control Markup Language) policies have been used for location privacy. XACML was developed to be used as uniform control policies. XACML is an XML-based language standardized by the organization for the Advancement of Structured Information Standards (OASIS)[6].

There are two major types of LBS privacy: Query privacy and Location privacy. Query privacy relates to users' private information as related to LBS query attributes. Location privacy refers to user's locations, as well as other information that can be found from the location.

There are many proposed approaches for LBS privacy protection. These can often be grouped to [1]: Policy-based schemes and location perturbation and obfuscation schemes. Figure 1 shows example of simple policy. Policy-based are legal terms that explain and define what service providers should do with users' personal data. The other schemes are looking to enforce the privacy protection of the users. One well known approach is K-anonymity protection mechanism which applied on data to protect privacy of users.

The other approach is generating dummies with movement patterns similar to those of real users providing LBS servers with mixed locations of real users and dummies. One of the


Policy

Simple rule

Target:

Location-reviver=find-nearest-restaurant

Filter strategy:

Anonymization

Unilo: r' = 50m

Figure 1: example of policy

well-known algorithm is the clocking area, instead of sending the exact position the system send an area which the user in that area.

The problem with policy-based approach it does not enforce the policy. So in [3] they combine the two approaches for more protection of user privacy and that give user more flexibility of applying certain policy based on which service that he looking for. LbSprint (Location-Based Service PRivacy INTegrator) is a middleware layer for privacy protection in location-based services [3] which implement XACML policy. The architecture is shown in figure 2. The request received by the adapter which reformats the request to unify location format. Policy Enforcement Point (PEP) passes the request to Policy Decision Point (PDP) which evaluates the request. Privacy policy has two parts as shown in figure 3; the first part is the user part while the second one is the system part. The user part can be written by the user themselves or by the system administrator. The system part is written by the system administrator. PDP first search the user part policy, if it did not find the user policy then it searches the system policy. Then it will apply the retrieved policy. In case there is neither user policy nor system policy it will apply a complete-filtering default rule.

The PEP applies the specified rules and releases A' as output. When A' = null, force a complete filtering corresponds to a denial of access. Location data must be manipulated before releasing it.

Filtering strategy could apply one or two algorithms based on type of request. All requests have to go through policy evaluation. If no filter to apply then no access will be granted.

This solution has the advantages of integrating both approaches together but it has a major disadvantage which is the performance. XACML policy is good for security policies from performance point of view but in case of LBS which introduce large number of users it affect the efficiency of the solution.

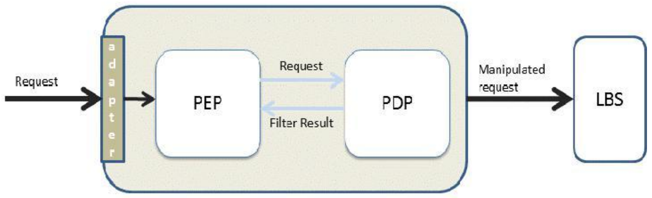


Figure 2 LbSprint Architecture

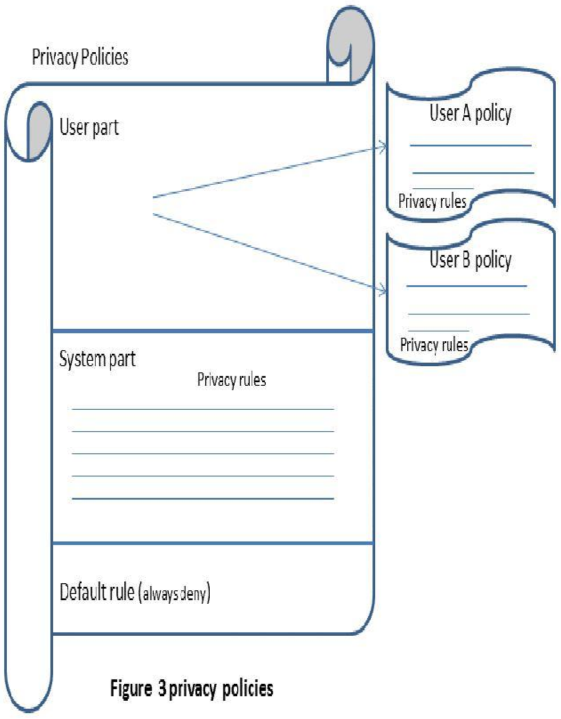


Figure 3 privacy policies

A user could request the find-the-nearest-restaurant service which will be represented as follows:

If** the location receiver is **find-the-nearest-restaurant

***Then** apply specific rules*

- The "**if**" part represents the target
- The "**then**" part represents the filter strategy.

PDP queries each request through the LbSprint architecture. So, it is very critical part for the performance of the solution.

In LbSprint, they use two obfuscation mechanisms; generalization and UniLO [8]. Generalization is based on a hierarchical description of the map. UniLO generates circular location areas. Both of the algorithms generate location area to

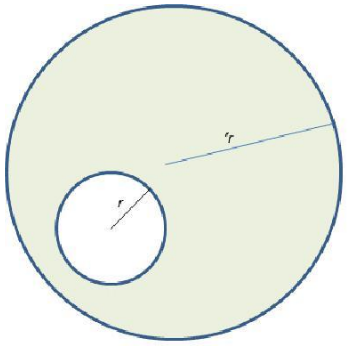


Figure 4 UniLO example

hide the identity of the users. The larger the area the more privacy gaining by the users but it is affects the precise of the location. At the same time there are many different algorithms that could be used instead [9], as anonymity [10], [11], [12] or data obfuscation [13], [14], [15], which could be used for the integration. These two algorithms have been selected for simplicity.

UniLO algorithms noise is added on a random vector to the original coordinates. The algorithm uses the radius from the sensor (*r*). Then it modify the radius to be r'. r' > r . Figure 4 shows an example of UniLO. Table I shows the filtering strategy for policy.

Generalization uses generalization tree as a hierarchical description of the map. For example, a generalization tree applicable to a university which could be as following: Exact position – Class room – Sector – Building – university campus. Generalization takes a parameter *k* and replaces the exact position with a zone which is *k* levels. For example, *k=0* corresponds to returning the exact position of the user, whereas *k*= 2 corresponds to the class room in which the student currently is. Figure 5 shows an example of location structure. This introduced Policy Decision Point (PDP) as bottleneck for the performance of the solution. The PDP is the decision maker. The PDP gets the request and generates an authorization-respond depending on policies. In PDP there are two types of policies; user's policies and system's policies. Lookup for Users policies is not efficient in a huge number of

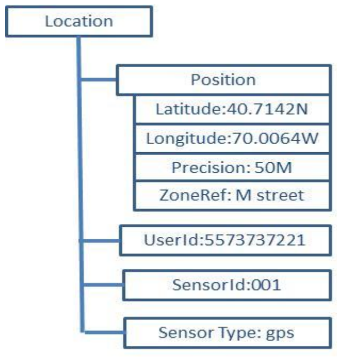
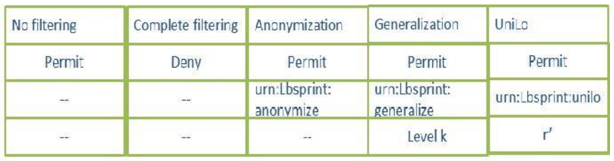


Figure 5: Example of location



No filtering	Complete filtering	Anonymization	Generalization	UniLo
Permit	Deny	Permit	Permit	Permit
--	--	urn:Lbsprint: anonymize	urn:Lbsprint: generalize	urn:Lbsprint:unilo
--	--	--	Level k	r'

Table I: policy Filter Strategies

communities for LBS users When PDP receives a request from PEP, it searches through the entire policies to see if it is applicable on that request.

The response of the PDP IS EITHER 'DENY' OR SPECIFY A SPECIFIC POLICY AND RULES TO BE APPLY ON THE REQUEST BY PEP. LBSPRINT WAS A GOOD STRATEGY to address location privacy, which apply both mechanisms together. But at the same time efficiency of the solution need more work to enhance the performance of the request evaluation.

Performance optimization focuses on PEP-PDP communication only. The enhancement of their improvement enhances the efficiency of about 60%.

The efficiency of the privacy solution has major impact of using LBS. when the user feel secure utilizing the service and can get the service in an acceptable response time not effecting their decision will encourage him to use other services. But when the response time is not acceptable, will not use the privacy solution which will have a negative result to his life. At the other end he me not use the service at all. Now, most service providers are taking consideration of the user's privacy.

Infrastructures need to powerful to accommodate all LBS services. Communications need to be secure. Network Band width has to be high. If we implement a back-end solution with high performance lacking an efficient network, that degrade our solution at the end.

The privacy solution, middle ware, hardware, has to be very powerful to handle all requests. Application and database servers are very important player in performance. One solution to deal with the server's performance is clustering the servers to run parallel evaluation. The response time has to be very high to satisfy the user requirement.

In general, our contributions are enhancing the performance of PDP by implementing the policy index and using policy cache which speed up the searching and retrieving requested policy and rule. The remainder of the paper is organized as follows. Section II presented some related works. Section III shows the solution architecture and the detailed design. Section IV analyzes the performance of the solution. Finally, we draw the conclusion of our work.

II. RELATED WORK

The location-based service privacy still needs more effort from researchers to satisfy the user requirements. In [4] they show a survey of privacy for LBS. commercial LBS providers don't make an effort to protect user's privacy. Foursquare

[16], Loopt [17] and Google's Latitude [18] are the most well-known service providers. In [19] they recommend that they will move towards a better protection of users' privacy, especially after the diffusion of so-called "stalker apps". Yahoo!'s Fire Eagle [20] is the first commercial service provider that consider the location privacy. Based on the Facebook platform the Locaccino [13] is a privacy-centric application for location sharing. The surveys [21] showed that people do not use any obfuscation mechanism to protect their privacy. It is focus on the access control. And it gives the user the ability to permit or deny.

Up to my knowledge no one has address the performance of Policy Decision Point (PDP) in location privacy policy. PDP In LocServ[14], location-based applications specify the privacy policies that will be adopted for their own privacy preferences using what they called validators. Then the application check the validators before releasing information, the validator can check with the user for the data to be released. One of the problems that face the location privacy is the reliability of communication [10]. In [22] the privacy policies and the location data are encapsulated in a unique entity so they are transmitted together quires database for each request which affect the efficiency of the performance [3]. With large city as Beijing having millions of customers the performance of PDP will be degraded and unacceptable.

Policies growth increase complexity of the policy repository which decreases policies evaluation performance [23]. PDP needs to consult a policy set and apply the rules to Permit or Deny each request and so can become a performance bottleneck [24]. The most time spend by PDP is searching to match the request against the policy set. Policy keeps growing and the evaluation is getting complicated and takes longer
[25]. In this paper they put a framework for performance testing for XACML policy so it can be used by researchers. Location is threatened by unauthorized access getting to secure data and with unwanted advertisements or to learn about users' medical conditions, alternative lifestyles or unpopular political views. XEngine was proposed as scheme for efficient XACML policy evaluation [26]. XEngine converts a textual XACML policy to a numerical policy then it converts a numerical policy with complex structures to a numerical policy with a normalized structure. After that it converts the normalized numerical policy to tree data structures for efficient processing of requests.

In [27] they proposed distributing the PDP engine and policy decomposition. The system forwards the request to the idle processor. A policy can be decomposed by the decomposition criteria based on the combination of three attributes: subject, action, and resource Using cache is a common way to enhance the performance of policy evaluation. [28]Cache can be used for PDP, when an event happens the PDP retrieve the policy and update the cache. Next time when this is requested the PDP first queries the policy through cached item in this case no need to visit the policies database.

In [29] they presented a performance enhancement by sing a graph based on the tribute IDs of a policies tree. Two data structures were created which are: Matching Tree and Combining Tree. The main concept of this approach is to transform the target matching of the policy sets, policies and rules into a decision diagram (DD). The reason for the performance advantage of this approach is the limitation of the evaluation time by the length of the evaluation paths in the decision diagrams. In turn, the maximum length is limited by the number of attributes used in the various target sections and is not bound to the number of policy sets, policies and rules. And they showed the enhancement of performance by applying their way. Meanwhile, still need more work for better XACML performance.

III. ARCHITECTURE AND IMPLEMENTATION

In our work we focus on enhancing the performance of the PDP. Request evaluation is done in PDP which search the entire policies to match the request. One of the requests 'element is the user-id which will be used for searching the Database. This user-id will be the mobile number. The work tries to speed the policy lookup by adding policies index and by caching the most recent used polices. Figure 6 shows the proposed solution architecture. There can be thousands of policies deployed on a single PDP it makes sense to have some index over the policies to identifies the applicable policies as fast as possible. The communication between the user and the system is secure [3]. Data that will be sent to the service provider will be anonymized. The solution is reliable and secure at the same time. Performance is optimized by using policy cache and policy index.

The solution consists of location sensors and users having smart phone. Second, communication network is used to transmit and receive information. Then the system consists of an adapter which translates all requests to a common format. Third, the solution consists of PEP which function is to pass the request to the PDP. Once PDP send the obligation to PEP, it applies the rule to the request and sends it to the service provider. The last component of the solution is PDP which the main focus of our research in this paper. PDP is the focal point of the solution. PDP get the request and look up for the policy for the user requesting the service through the entire policy database. The service provider is considered as part of the solution. it respond to the user with the requested information. Figure 7 shows the PDP request processing.

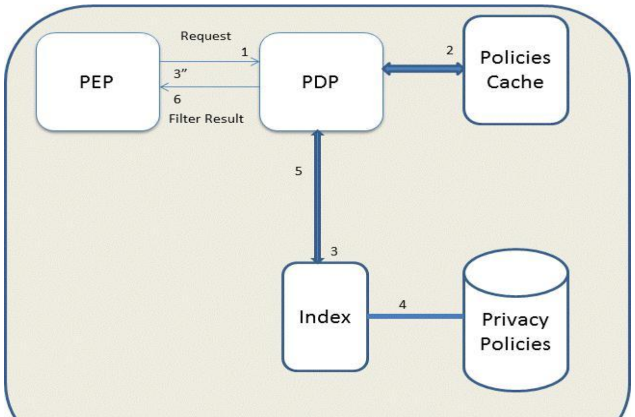


Figure 6 Solution Architecture

The communication between the user and the system is secure [3]. Data that will be sent to the service provider will be anonymized. The solution is reliable and secure at the same time. Performance is optimized by using policy cache and policy index.

A. *Location based Policy index.*

In a data base having millions of policies in a city like Beijing it will be a bottleneck on the process for look up for a policy. The most accesses to the policy repository will be read-access because one single policy is not going to change often. We are proposing a hash table for policies to be used. We are assuming the system will provide the services to large number of users which require a very quick response for the user's requests. So we will design our hashing algorithms by applying hash function of the last 5 digits of the user mobile number assuming it is the user Id. We will use hash function with linked list approach. PDP will receive the request from PEP then it look for the policy in the policy cache if it found then it respond to the PEP with the appropriate rules if it did not find then it will search the index for the user-id.

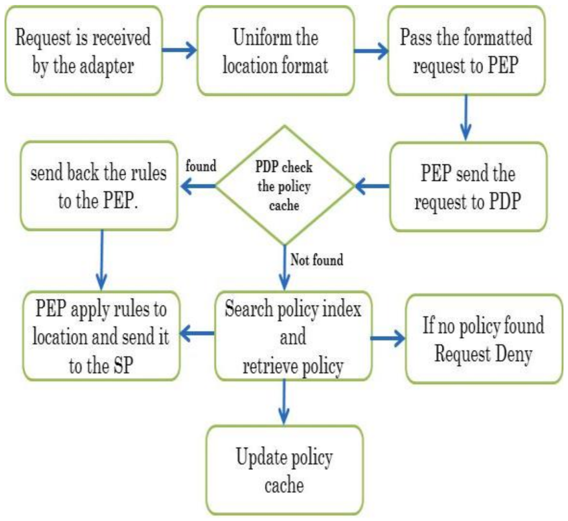

Request is received by the adapter
Uniform the location format
Pass the formatted request to PEP
PEP send the request to PDP
PDP check the policy cache
found
send back the rules to the PEP.
Not found
PEP apply rules to location and send it to the SP
Search policy index and retrieve policy
If no policy found Request Deny
Update policy cache

Figure 7: PDP XACML request processing

The search time will be almost *O (1)* and the rules retrieval will be direct. Index table will eliminate the searching time through the XACML policies. Normal policy search for a record is very slow or *O(n)* since we search the entire table. Our proposed solution will first use the hash function to generate the index for search.

H(x) = I; H is the hash function; *x* is the *ID* which in our case is the mobile number. Then *I* will be used to locate the record. In case of collision link list will be used. Hashing function is *O(1)* and locating the record is also *O(1).* By using the last five digits of the mobile number we are expecting the collision will be minimal. Adding and updating policy is not an issue since the policy is not very often updated. Policy can be added by the user or the administrator and our system will handle the indexing.

B. *Policy cache.*

Another approach to enhance the efficiency of the PDP is by using caching technique for policy. Part of our work we implement policies cache using FIFO algorithms. The other algorithm that could be used is the Least-Frequently Used (LFU) counts how often an item is needed; those that are used least often are discarded first. By using FIFO we satisfy the reading requirement and optimizing the adding and removal time. But we scarify the most frequently used policy. So, at the end it is debatable decision. Most of the time, the same authorization query can hit the PDP multiple times; therefore it creates a considerable performance hit if the requested policy in cache.

Maximum number of policies that can be kept in cache depends on the resources available. Cache must be updated when any new policy retrieved from store. So, as the cache has more room for new cache policy it will be added to the cache. Once the cache is full then it use the principle of FIFO which means the first policy will be removed from the cache and the new one will be added to the cache. Updating policy cache does not considerable to have an impact on the performance. Cache will enhance the performance once the policy has been loaded. And we are assuming that there are good numbers of re-requesting the same location information. The index will use numerical Id; which is a direct access. To maximize the cache policies a carful cache implementation should be consider. We used heap binary tree which make searching is fast, O(logn). Insertion to the heap is O(logn), and the same goes for deletion.

V. PERFORMANCE

PDP queried for each request which make it very critical point for the evaluation process. In our test and simulation we consider that each user has only one policy. With that searching by using index on numerical data makes the searching very well. The main goal of our work is to enhance the performance of the policy evaluation. Using policy index

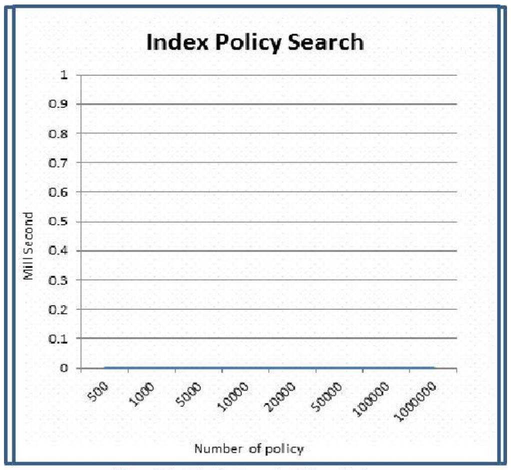


Figure 10. index policy search time performance

get performance almost to O(1) in most cases. Policy cache will eliminate the database visit in case of the item found. Evaluation is optimized. The others part of the system will still affect the performance; the hardware specifications and network speed. Numbers of policies and rules and request will cause the system to delay response when resources are not up to the requirements. Our experiments were carried out on a laptop running Windows 7 with 4Gb of memory and an Intel Core i7-Q720 processor, with 4 cores at 1.6GHz and 6Mb cache. We generate a simulator for 500-50000 users having the same number of policies.

We run the evaluation to search through the policies searching the textual data; figure 8 shows the result of the evaluation. As we can see the time spent for 50000 users is too high. Then we run the evaluation to search through the policies searching the numerical data; figure 9 shows the result of the evaluation. The third simulation is to use the policy index and applying the hashed table. The result was very good.

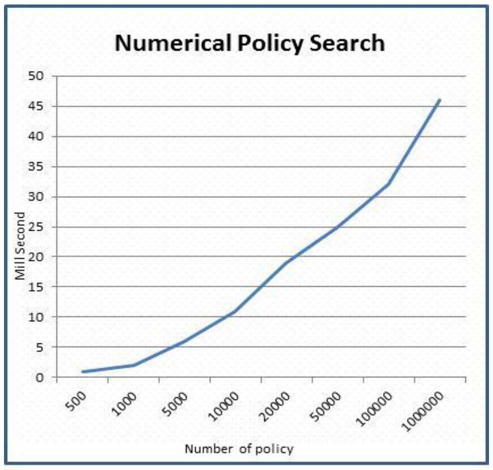


Figure 9. numerical policy search time performance

The other test we simulate is to run the simulation with cache in two different cases; policy already found in the cache and the other case policy not in cache, we run the simulator with indexing. As a result the analysis showed as we can see in figure x comparing between the different scenarios that implementing the index enhance the performance.

There are many parameters affecting the performance of the PDP, such as the number of policies in the policy repository, is it centralized or distributed PDP. The size of the cache and the update algorithms affect the performance of evaluating time. This is one aspect of performance measurement only. This focuses on enhancing the evaluation process of the policy. Reducing the evaluation time is main part of the efficiency requirements.

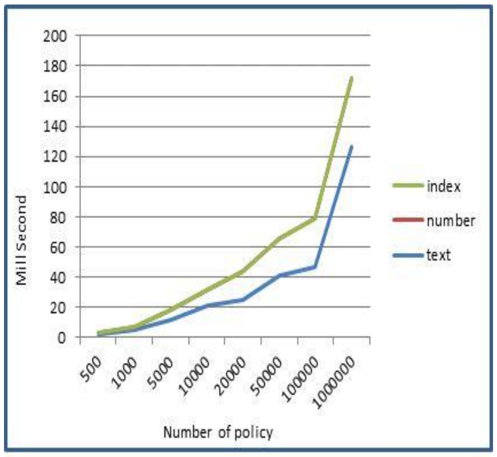


Figure 11 comparing time performance

IV. Conclusion

We presented an improvement to the PDP performance by implementing two mechanisms. First, implement policy index. This index will be used to allocate the policy for the users. Once the policy-id found then it will retrieve the policy on direct accesses. This has a major improvement of the evaluation. The second mechanism, is implementing policy cache which is used first to locate policy if it has been request before. Cache policy improve the performance in case if policy already in cache. It is very important to secure the user information and have efficient system. This will encourage users to use this kind of services

LBSs are now part of the uses daily usage. They use the services for information purpose. That makes people life more easer. You can do any location application with mobile device. So, services are available but privacy mechanism not up to user’s expectation yet. At the same time performance is major issues for LBS sage. This area needs more research and focus from service providers and researches.

REFERENCES

[1] Shin, K. G., Ju, X., Chen, Z., & Hu, X. (2012). Privacy protection for users of location-based services. *Wireless Communications, IEEE*, *19*(1), 30-39.

[2] Lee, B., Oh, J., Yu, H., & Kim, J. (2011, August). Protecting location privacy using location semantics. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1289-1297). ACM.

[3] Dini, G., & Perazzo, P. (2013, March). Integration of Privacy Protection Mechanisms in Location-Based Services. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*(pp. 717-724). IEEE.

[4] Tsai, J. Y., Kelley, P. G., Cranor, L. F., & Sadeh, N. (2010). Location-sharing technologies: Privacy risks and controls. *ISJLP*, *6*, 119.

[5] Liu, A. X., Chen, F., Hwang, J., & Xie, T. (2011). Designing fast and scalable xacml policy evaluation engines. *Computers, IEEE Transactions on*,60(12), 1802-1817.

[6] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, *4*(3), 224-274.

[7] Consolvo, S., Smith, I. E., Matthews, T., LaMarca, A., Tabert, J., & Powledge, P. (2005, April). Location disclosure to social relations: why, when, & what people want to share. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 81-90). ACM.

[8] Dini, G., & Perazzo, P. (2012). Uniform obfuscation for location privacy. In *Data and Applications Security and Privacy XXVI* (pp. 90-105). Springer Berlin Heidelberg.

[9] Bettini, C., Mascetti, S., Wang, X. S., Freni, D., & Jajodia, S. (2009). Anonymity and historical-anonymity in location-based services. In *Privacy in Location-Based Applications* (pp. 1-30). Springer Berlin Heidelberg.

[10] Langheinrich, M. (2002). A privacy awareness system for ubiquitous computing environments. In *UbiComp 2002: Ubiquitous Computing* (pp. 237-245). Springer Berlin Heidelberg.

[11] Gedik, B., & Liu, L. (2008). Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, *7*(1), 1-18.

[12] Marconi, L., Di Pietro, R., Crispo, B., & Conti, M. (2010). Time warp: how time affects privacy in LBSs. In *Information and Communications Security* (pp. 325-339). Springer Berlin Heidelberg.

[13] Toch, E., Cranshaw, J., Hankes-Drielsma, P., Springfield, J., Kelley, P. G., Cranor, L., ... & Sadeh, N. (2010, September). Locaccino: a privacy-centric location sharing application. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing-Adjunct* (pp. 381-382). ACM.

[14] Myles, G., Friday, A., & Davies, N. (2003). Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1), 56-64.

[15] Damiani, M. L., Bertino, E., & Silvestri, C. (2009, November). Protecting location privacy against spatial inferences: the PROBE approach. InProceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS (pp. 32-41). ACM.

[16] Foursquare. [Online]. Available: http://foursquare.com

[17] Loopt. [Online]. Available: http://www.loopt.com

[18] Latitude. [Online]. Available: http://www.google.com/latitude

[19] "Foursquare alters API to eliminate apps like Girls Around Me." [Online]. Available: http://aboutfoursquare.com/foursquare-api-change-girlsaround- me/

[20] FireEagle. [Online]. Available: http://fireeagle.yahoo.net

[21] Consolvo, S., Smith, I. E., Matthews, T., LaMarca, A., Tabert, J., & Powledge, P. (2005, April). Location disclosure to social relations: why, when, & what people want to share. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 81-90). ACM.

[22] Morris, J., Cooper, A., Tschofenig, H., Lepinski, M., Barnes, R., & Schulzrinne, H. (2011). An Architecture for Location and Location Privacy in Internet Applications.

[23] Ngo, C., Makkes, M. X., Demchenko, Y., & de Laat, C. (2013, July). Multi-data-types interval decision diagrams for XACML evaluation engine. In *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on* (pp. 257-266). IEEE.

[24] Butler, B., Jennings, B., & Botvich, D. (2011, May). An experimental testbed to predict the performance of XACML Policy Decision Points. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on* (pp. 353-360). IEEE.

[25]Butler, B., Jennings, B., & Botvich, D. (2010, October). XACML policy performance evaluation using a flexible load testing framework. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 648-650). ACM.

[26] Liu, A. X., Chen, F., Hwang, J., & Xie, T. (2008, June). Xengine: a fast and scalable XACML policy evaluation engine. In *ACM SIGMETRICS Performance Evaluation Review* (Vol. 36, No. 1, pp. 265-276). ACM.

[27] Deng, F., Chen, P., Zhang, L. Y., Wang, X. Q., Li, S. D., & Xu, H. (2014). Policy Decomposition for Evaluation Performance Improvement of PDP. *Mathematical Problems in Engineering*, *2014*.

[28] Liu, L., Han, W., Bertino, E., Zhou, T., & Zhang, X. (2013, July). Efficient General Policy Decision by Using Mutable Variable Aware Cache. In *COMPSAC* (pp. 359-368).

[29] Pina Ros, S., Lischka, M., & Gómez Mármol, F. (2012, June). Graph-based XACML evaluation. In Proceedings of the 17th ACM symposium on Access Control Models and Technologies (pp. 83-92). ACM.

Authors Profile

Abdullah M.Al-Shomrani received his B.Sc. degree in Computer Science from King Abdul-Aziz University, Saudi arabia. He then went to Florida Institute of Technology,United Statesof America, where he got his master degree. He is currently a Ph.D. student in the Department of Computer Science at King Abdul-Aziz University, Saudi Arabia. His research interests include data security and privacy, distributed systems and big data.

Jonathan Cazalas received the Ph.D. degree in computer science from the University of Central Florida, United States of America, in 2012. He is currently an Assistant Professor at King Abdul-Aziz University, Saudi Arabia. His research interests include mobile computing, location-based services, wireless networks, and privacy and security.