Integration of Privacy Protection Mechanisms in Location-Based Services

Gianluca Dini, Pericle Perazzo
Dept. of Information Engineering
University of Pisa
Pisa, Italy
Email: [name.surname]@iet.unipi.it

***Abstract*—In the next few years, we will see the upcoming of location-based services. Such LBSs will be extremely heterogeneous. Protecting the privacy of the users in such a situation requires flexible approaches. A single privacy protection mechanism is often insufficient. The contribution of this paper is two-fold. First we present LbSprint, a middleware architecture for location-based services which integrates different privacy mechanisms by means of the standard XACML language. The system administrator can configure and extend the set of such mechanisms. To the best of our knowledge, this is the first proposal of an architecture which integrates many privacy mechanisms in an extensible way. Secondly, we present practical optimizations which considerably improves the performance of the XACML policy evaluation process.**

***Keywords*-privacy; location-based services; integration; middleware;**

I. Introduction

Due to the proliferation of tracking technologies, like GPS, the interest in location-based services (LBSs) is growing fast. Nowadays, a plethora of technologies are capable of localizing people: palm GPS, RFID, video cameras, and so on. Recent studies [1] showed that users feel that the privacy risks of using LBSs still outweigh the benefits. Therefore, in order to be accepted by users, LBSs must be trusted on the standpoint of privacy [2], [3]. Protecting privacy in an LBS is not an easy task. Privacy policies need be flexible, integrated, customizable and context-aware [1], [4], [5], [6]. Current solutions [1], [6], [7], [8], [9], [10] focus on *access control* approach, i.e. the system decides which information will be released and which not, basing on some authorization rules. Such rules are *context-aware*, meaning that they take into account the current date and time, the location of the user, and the situation of the user or the system itself (e.g. the presence of alarms in a particular building, etc.). Though access control is essential, the "permit-or-deny" logic behind it forces the users to choose from having the service or having the privacy. Recent years have seen the emergence of more flexible mechanisms [11], [12], [13], [14], [15]. Many applications prefer *anonymization approaches*, in which the identity of the user is detached from the information. Many others prefer *obfuscation approaches*, in which the system artificially degrades the precision of information before releasing it. In this way, users can tailor their own trade-off between privacy and quality of service.

However, this may not be enough. For many services, a single mechanism is not sufficient to meet the privacy requirements, which can be instead fulfilled only by a proper *integration* of different protection mechanisms. Think about a simple find-the-nearest-restaurant service. The user gives her position to a service provider in order to get the name of the nearest restaurant which meets some requirements. How can we protect the privacy of the user? Her identity is unnecessary, so the location information could be anonymized. Also, the service provider does not need the exact position, so an obfuscation algorithm could be applied.

The contribution of this paper is two-fold. First, we present LbSprint (Location-Based Service PRivacy INTegrator), a middleware layer for privacy protection in location-based services. LbSprint supports multiple privacy-protection mechanisms, allows system administrators to define new ones, and allows users to setup flexible and context-aware privacy policies. LbSprint implements these features by means of the XACML language capabilities [16]. Secondly, we present optimizations that considerably improves the performance of the XACML policy evaluation. These contributions apply to LBSs as well as on privacy-protection architectures in general.

The rest of the paper is organized as follows. Section II presents some related works, and underlines similarities with and differences from LbSprint. Section III explains the system requirements and presents a typical use case. Section IV includes a detailed description of the system architecture and the privacy protection module. Section V analyzes the performance of the system and describes some optimizations to improve it. Finally, we draw our conclusions in Section VI.

II. Related works

To the best of our knowledge, the problem of privacy mechanism integration in LBSs has not yet been addressed by research or industry.

Commercial LBSs have a poor support for privacy protection. See [1] for a complete survey. The most known exam-

ples are maybe Foursquare [8], Loopt [10] and Google's Latitude [9]. Recent facts suggest that they will move towards a better protection of users' privacy, especially after the diffusion of so-called "stalker apps" [17].

Yahoo!'s Fire Eagle [7] has been one of the first commercial LBS platforms posing particular attention to location privacy. Users are capable of specifying relatively complex privacy policies in terms of who can access their locations and when. It gives also support for obfuscation. Locations can be specified with several degrees of granularity (exact position, ZIP code, neighborhood, city, etc.). However, it does not offer the possibility to integrate different privacy protection mechanisms and to define new ones.

Locaccino [6] is a privacy-centric application for location sharing, based on the Facebook platform. Its features came from some surveys the authors did to investigate privacy preferences of people. Users can create simple policies specifying who, when and where can see their location. Locaccino is focused on access control, and gives to the users the option to permit or deny the access to their location. It poses little focus on integration between privacy mechanisms. The authors chose not to include any obfuscation mechanism, because surveys [5] showed that people do not use it to protect their privacy. However, the obfuscation method investigated in such surveys was quite an inflexible one. It gave the user the possibility to release either her exact position, or the position with city-level granularity. In LbSprint, we take into account more flexible obfuscation methods, that offer finer granularities. These methods are suitable for a plethora of location-based services, as we will show in Section III.

From the research world, one of the first privacy-centric tracking systems has been LocServ [18]. In LocServ, location-based applications make their location requests and specify the privacy policies they will adopt in using such information. The users provide for location data and specify their own privacy preferences, represented by components called *validators*. Before releasing information to an application, LocServ asks the correspondent validator whether the policy of the application is compatible with the user's preferences. A validator, during its decisional process, can consult the user or other (possibly external) subordinate validators.

A related problem is the reliable communication of privacy policies [19]. Geopriv is a standard developed by IETF, aimed at the representation and transmission of location data [20]. The basic idea is that the privacy policies and the location data are encapsulated in a unique entity, the *location object*. So they are always transmitted together. The integrity of the location object is guaranteed by a digital signature. If Alice wants to share her position, she will define a privacy policy, which specifies how her location must be used and distributed. Each user able to read Alice's position is himself aware of the rules stated by Alice. Geopriv does not really forbid other users to share Alice's information. It just ensures that, if someone breaks a rule, he cannot claim he was unaware of it. LbSprint focuses on the orthogonal problem of privacy mechanism configurability and extensibility. Geopriv mechanisms are utilizable in LbSprint as well.

III. The case for integration of privacy protection mechanisms

The following is a use-case story which illustrates several location-based services in an example scenario: an airport. Each service needs a customized mix of privacy protection mechanisms.

> John has to take a flight together with his little son, Tim. They reach the airport and they get to the check-in area. Before leaving his luggage, John subscribes to the *track-my-luggage* service. Such a service tracks the position of John's baggage for security purposes. See [21] for an example of this. John is informed about it whenever he wants through his smartphone.
>
> Once they checked-in, John and Tim want to have lunch. John prefers vegetarian food but he does not have time for searching a vegetarian restaurant. So, he uses the *find-the-nearest-restaurant* service, which finds the nearest vegetarian restaurant. After lunch, Tim wants to visit a toy shop he saw before. John lets him go, but he wants to track him position, because he does not want him to get too far. To do this, John uses the *track-a-child* service. After a while, John realizes that the boarding time is soon. He quickly reaches Tim, because he knows his position, and they easily get to the gate.
>
> For all the story time, John and Tim had used several other location-based services. For example the *track-for-safety* service, which tracks John and Tim's positions in order to help rescuers in case of fire, and the *find-my-friends* service, a social service that informs John about the proximity of friends.
>
> Also, the *track-the-employees* service is active on the airport. Such a service allows an operator to track the position of the personnel for management purposes.

Each of the above services requires a different mix of mechanisms for the protection of the privacy. Track-my-luggage needs only authorization, since only John is allowed to know the location of his baggage. Permitting anyone else could be harmful for the security (e.g. luggage stealing). Find-the-nearest-restaurant does not need to know the exact position or the identity of John. Thus, John's position is obfuscated and John's identity anonymized. The service provider is allowed to receive such a position, other entities are not. Track-a-child requires authorization, as only John is

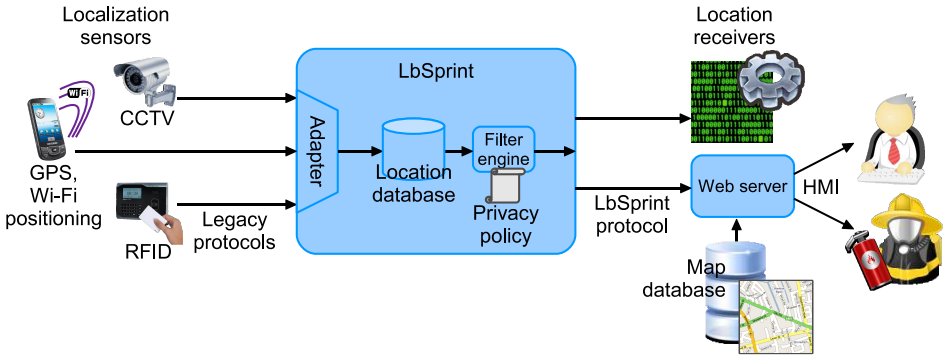

Figure 1. LbSprint architecture

Service type:	Authorized receivers:	Anonymized:	Obfuscated:
Track-my-luggage	the owner	–	–
Find-the-nearest-restaurant	the service provider	yes	yes
Track-a-child	the parents	–	–
Track-for-safety	the safety operators *context:* fire alarm on	–	–
Find-my-friends	the service provider	–	optional
Track-the-employees	the personnel manager *context:* working hours	–	yes

Table I
SUMMARY OF THE PRIVACY POLICIES IN USE-CASE STORY

permitted to access the location of his son. However, a parent wants to know its children's position very precisely. So no obfuscation method is applied. Track-for-safety requires authorization, because only rescuers can access passengers' locations, but no anonymization neither obfuscation, as they could hinder the emergency operations. However, the passengers' locations can be accessed only in a particular context, i.e. in case of a fire alarm. Find-my-friends needs to know John's identity and those of his friends, but does not need their exact positions. Thus, John can decide whether to obfuscate or not his own position. The service provider (i.e. the social network platform) is allowed to receive such a position, other entities are not. Finally, track-the-employees needs authorization, as only the personnel manager can access employees' positions and only during the working hours, and obfuscation. The current time and the presence of alarms are examples of context attributes. Table I summarizes the privacy mechanisms applied for each service. The "authorized receivers" column lists who is allowed to receive location information. The "anonymization" and "obfuscation" columns tells us whether the locations will be respectively anonymized and obfuscated.

It follows that an effective location-based service middleware must support different mechanisms for privacy protection as well as different ways of integrating them.

IV. Architecture and Implementation

LbSprint follows a centralized architecture, as shown in Figure 1. Every *location receiver* first authenticates to LbSprint, and then asks for and receives location notifications. All communications are secure in terms of confidentiality, authenticity and integrity. Different receivers have different rights for accessing location data, as summarized in Table I. LbSprint supports various localization technologies: handheld GPS and Wi-Fi positioning, RFID, CCTV (Close-Circuit TeleVision) [22]. The localization sensors generate streams of raw location measurements, and send them to LbSprint through a plethora of legacy protocols. Some sensors (GPS, Wi-Fi positioning, RFID) provide for the identity of tracked users too. Some other (CCTV) does not, so that location measurements are associated to pseudonyms

Figure 2. LbSprint human-machine interface

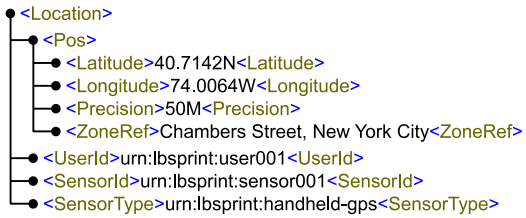

Figure 3. Example of `Location`

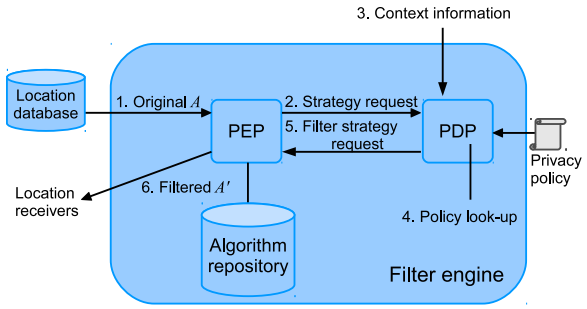

Figure 4. Workflow of the filter engine

(e.g. urn:lbsprint:user001). An adapter module converts the different formats of location measurements in a common format (*location data*). A *filter engine* manipulates it, basing on a *privacy policy*. Finally, the location data stream is forwarded to the receiver by means of the LbSprint protocol. LbSprint provides also for a human-machine interface (HMI), which visualizes the location data on a map. The HMI is a web application which authenticates as a normal receiver. Figure 2 shows a screenshot of it, taken from a practical implementation in a company.

The LbSprint protocol is built over SOAP [23]. Through the entire architecture, the location data is represented by an XML structure called `Location` (Figure 3). A `Location` contains the position along with possible other meta-data, such as the identity of the user, the sensor ID and type, and so on.

A. Filter engine

LbSprint uses the filter engine to protect the privacy of the users. The filter engine's privacy policy is configurable by the system administrator and by the users. The privacy policy takes into account many factors, including the context, and specifies different privacy mechanisms, e.g. authorization, anonymization and obfuscation.

Figure 4 shows how the filter engine works. Before LbSprint sends a `Location` A to a location receiver, it passes A to the *policy enforcement point* (PEP). The PEP makes a *filter strategy request* to the *policy decision point* (PDP). The PDP is a module which interprets the privacy policy and decides which manipulations must be applied on A (*filter strategy*). The PEP applies the specified

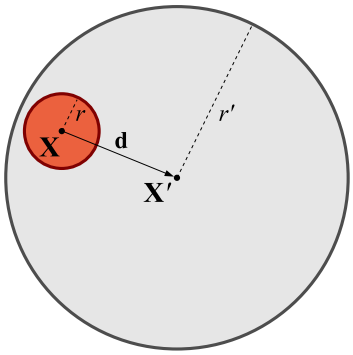

Figure 5. Perturbation example

manipulations and releases A' as output. A filter strategy could force a *complete filtering*, that is the location data is not disclosed at all ($A' = \text{null}$). This corresponds to a denial of access. On the other hand, a *no filtering* strategy releases the location data "as-is", without manipulations ($A' = A$). Other filter strategies could apply one or more manipulation algorithms on A, in order to anonymize it, obfuscate it, etc. Subsection IV-C will show how users can specify and configure such filter strategies.

B. Obfuscation methods

As a proof of concept, we provided LbSprint with two simple obfuscation mechanisms: *generalization* and UniLO [12]. They both replace an exact position with a *location area*, which contains such a position but has a larger extension. The larger is the location area, the less precise is the obfuscated location.

Generalization is based on a hierarchical description of the map (*generalization tree*), which defines zones with different coarseness. For example, a generalization tree applicable to a company could be the following: *Exact position* $\rightarrow$ *Room* $\rightarrow$ *Sector* $\rightarrow$ *Building* $\rightarrow$ *Entire system*. Generalization takes a parameter k and replaces the exact position with a zone which is k levels away from the bottom of the generalization tree. For instance, $k = 0$ corresponds to returning the exact position of the user (*Exact position*), whereas $k = 3$ corresponds to returning the building in which the user currently is (*Building*).

On the other hand, UniLO [12] aims at generating circular location areas. The original location data carries an indication of the *precision radius* of the sensor. For example, $r = 5\,\text{m}$. UniLO takes a parameter $r' > r$, for example $r' = 25\,\text{m}$, and adds noise to artificially increase the precision radius to r'. The noise is a *random vector*, which is added to the original coordinates. Figure 5 shows an example of UniLO application. $\mathbf{X}$ and r are respectively the original position and precision radius, whereas $\mathbf{X}'$ and $\mathbf{d}$ are the perturbed position and the random vector. The circle centered at $\mathbf{X}'$ and with radius r' is the resulting location area. If the location is asked again, and the user

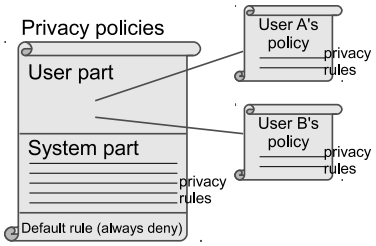

Figure 6. Privacy policy structure

is still inside the same location area, the same location area is returned. This is in order to avoid that someone intersects many location areas to find the true location of the user. The random vector has a specific probability distribution, in order to generate obfuscated locations as uniform as possible from the probabilistic standpoint.

Generalization and UniLO cannot be applied together. Since no method is better than the other, the policy writer must choose the most suitable method for each case. Generalization is probably more intuitive. Nevertheless, it requires a hierarchical specification of the map, which could be missing. On the other hand, UniLO is less intuitive, but generates circular location areas. Circles have simpler shapes than rooms or buildings, and they are easier to be processed by geometry-based algorithms.

We implemented two simple obfuscation algorithms and a simple anonymization algorithm only to demonstrate the integration capabilities of the system. Many other mechanisms exist [24], aimed at anonymity [14], [25], [26] or data obfuscation [11], [13], [27], which could be integrated as well. The analysis of these techniques falls outside the scope of this paper.

C. Privacy policies

A privacy policy comprises two parts: a *user part* and a *system part* (Figure 6). The user part contains policies specific to single users. These can be written by the users themselves or by the system administrator. The system part is written by the system administrator. The PDP tries first to apply the user part, in particular the policy of the user which location data refers to. If a user did not write her own policy, or her policy does not apply to the case, the system part will be applied. If neither the system part is applicable, a complete-filtering default rule will be applied. The system administrator can install a new privacy protection mechanism. For example a new obfuscation algorithm. In such a case, she must define the URI of the algorithm (e.g. urn:lbsprint:my-obfuscation), its parameters, and finally provide a Java class implementing the new algorithm. This makes our middleware extensible and configurable.

The receiver, the location, and the context are represented as collections of *attributes*. Each attribute is a name-value pair. Each *privacy policy* is a list of *privacy rules*. A *simple*

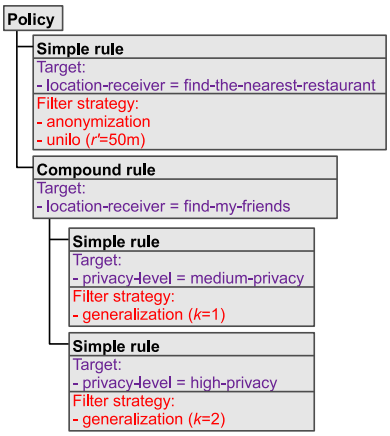

Figure 7. Example of policy tree

privacy rule is composed of a *target* and a *filter strategy*. The target is a boolean expression on (the attributes of) the location receiver, the location data, and the context. It decides whether the rule is applicable or not. If it is applicable, the corresponding filter strategy will be applied on location data. Otherwise, the next rule will be evaluated, and so on. To fix the ideas, in the use-case story of Section III, a user could specify the following rule for the find-the-nearest-restaurant service:

> "if the location receiver is *find-the-nearest-restaurant*, then apply anonymization and UniLO obfuscation with $r' = 50\,\mathrm{m}$"

The "if" part represents the target, the "then" part represents the filter strategy.

In such a way, complex policies are represented by long lists of rules. Such a "flat" schema is simple, but it is not modular. For example we may need to group all the rules referring to a particular service type, in such a way to write more compact, modular, readable policies. In addition, tree-shaped policies outperform flat ones, as shown in Section V. For these reasons, LbSprint allows the administrator to define *compound* privacy rules. Compound rules contain no filter strategy, instead they have a target and a list of *sub-rules*, each of them can be both simple or compound. The list of sub-rules cannot be empty. A compound rule is applicable only if its target evaluates to true and at least one of its sub-rules is applicable. The filter strategy of the applicable sub-rule will be the filter strategy of the entire compound rule. Compound rules allows us to define tree-shaped policies, which perform better in terms of processing time. An example of policy could be the following:

> "if the location receiver is *find-my-friends*, then: if the privacy level is medium, then apply a generalization of 1 level, otherwise if the privacy level is high, the apply a generalization of 2 levels"

Figure 7 shows an example of policy, with a simple rule and a compound one.

Filter strategy:	Final effect:	Obligation:	Parameter:
No filtering	Permit	-	-
Complete filtering	Deny	-	-
Anonymization	Permit	urn:lbsprint:anonymize	-
Generalization	Permit	urn:lbsprint:generalize	level (k)
UniLO	Permit	urn:lbsprint:unilo	radius (r' in meters)

Table II
FILTER STRATEGIES

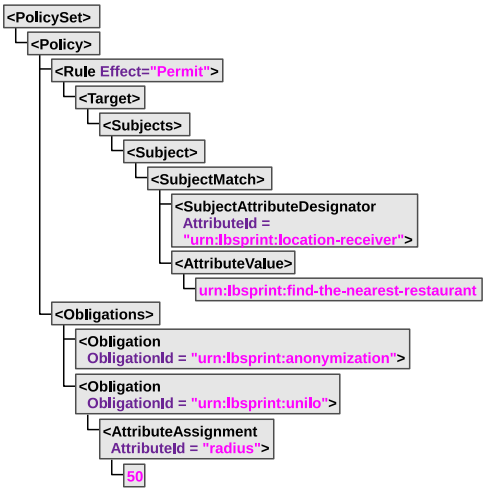

Figure 8. A simple rule in the XACML language

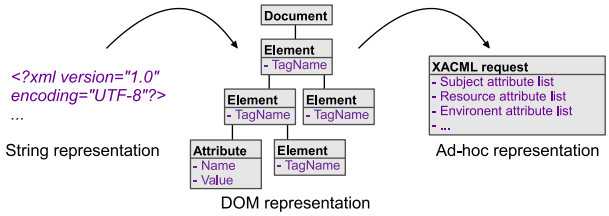

Figure 9. XML conversion

The privacy policies are written in the XACML language [16]. XACML (eXtensible Access Control Markup Language) is an XML dialect standardized by the OASIS consortium. It expresses extremely flexible rules for access control. We used SunXACML, the off-the-shelf PDP provided by Sun Microsystems, written in the Java language [28]. The XACML language allows every authorization decision to have only two valid outcomes: "Permit" or "Deny". In order to specify more complex filter strategies, we used the XACML `<Obligation>` tag. Figure 8 shows the find-the-nearest-restaurant simple rule of Figure 7, expressed in the XACML language. The attributes of the location receiver, location data and context are referred by URIs, along with the manipulation algorithms. The `<PolicySet>` tag represents either the entire privacy policy, or a compound rule. A simple rule is represented by the `<Policy>` tag. A target is represented by the `<Target>` tag. A filter strategy is represented by the `Effect` XML attribute and the `<Obligation>` tag. An `Effect` equal to `Deny` corresponds to a complete filtering. The `<Obligation>` tags are used in XACML to dictate additional duties that the PEP must fulfill before granting the authorization. For example sending a notification email to the system administrator. We use such tags to specify (mixes of) manipulation algorithms to be applied on location data. The algorithms are applied in the same order of the corresponding `<Obligation>` tags. XACML allows us to specify the parameters of an obligation, by means of `<AttributeAssignment>` tags. We use them as the parameters of the algorithm.

Table II tells us how basic filter strategies are expressed in terms of final effect, obligations and parameters. Mixed filter strategies, like "anonymization and generalization" are expressed with the union of the relative obligations, as shown in Figure 8. In such a case, the manipulations are applied in order of appearance.

Note that our integration method can be seamlessly adapted to XACML extensions focused on geographic data, like Geo-XACML [29]. For the sake of simplicity —and without loss of generality— we refer to the basic XACML standard.

V. PERFORMANCE

The PDP is queried for each `Location` flowing through the LbSprint architecture. Therefore, it is a quite critical module for the system efficiency. Since our PDP is based on XML, which is a textual representation, some performance-related concerns could arise. We present a practical ready-to-use optimization that considerably improves the performance of the policy evaluation process.

In XACML standard, the authorization requests and the authorization decisions are represented by XML code. Such a code is usually sent to the PDP as an XML string. This is the simplest approach, since the construction of the request involves only string concatenations. The request is successively converted in a DOM (Document Object Model) representation, and then in a PDP-specific internal representation (Figure 9). The first conversion involves an XML parsing that is burdensome in terms of computational resources. If the PEP and the PDP communicate locally, it is better to send the filter strategy request directly in a

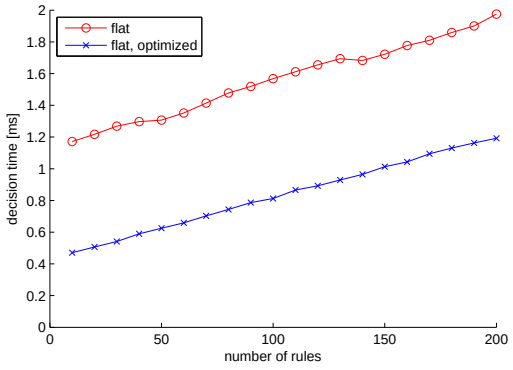

Figure 10. SunXACML time performance (flat policies)

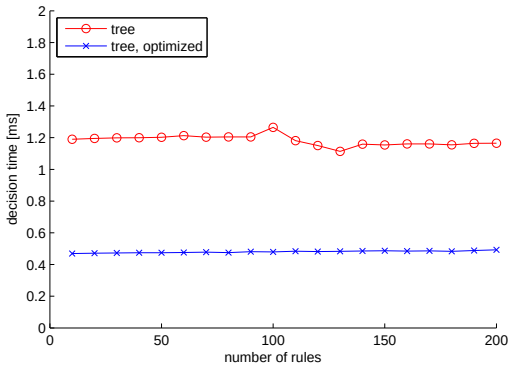

Figure 11. SunXACML time performance (tree policies)

DOM format. Such a DOM must be built node by node, by means of Java XML APIs. This requires more programming complexity than just making XML string concatenations. However, it has a good effect on performance.

We have conducted a series of tests aimed at evaluating the effect of such an optimization. Time efficiency has been evaluated on synthetic privacy policies containing up to 200 rules. Each target contained string matches and other comparisons on multiple attributes. Our experiments were carried out on a laptop PC running Windows 7 with $4\,\mathrm{Gb}$ of memory and an Intel Core i7-Q720 processor, with 4 cores at 1.6GHz and $6\,\mathrm{Mb}$ cache. Figure 10 shows the trend of the time performance of the evaluation of a flat policy versus the number of rules. The efficiency improvement goes from 40% (at 200 rules) to 60% (at 10 rules). In order to test the optimization effect on compound rules, we reorganized the policy in a balanced k-ary tree with $k = 10$. Non-leaf nodes represents compound rules, whereas leaves represents simple rules. All the targets evaluate a numerical attribute. The targets of the compound rules test it for being comprised inside a range. The targets of the simple rules test it for being equal to a single value. If the target of a compound rule evaluates into false, those of its children will not be evaluated. Thus, in a balanced tree, the number of target evaluations will grow as $\log n$, where n is the number of rules. The balanced tree is the best-case structure from the performance standpoint, whereas the flat structure is the worst case. Figure 11 shows the trend for such a hierarchical policy. From the experiments, we see that the evaluation performance is roughly constant with respect to the number of rules. This is because the time which the PDP takes to perform rule-independent tasks (e.g. conversion from DOM representation to ad-hoc representation) is preponderant with respect to the target evaluation time. The optimization brings a uniform efficiency improvement of about 60%.

More optimizations are possible on the PDP, based on more complex techniques like rule indexing, attribute numericalization, etc. [30], [31] Our optimization focuses on PEP-PDP communication, rather than on PDP internal working. So it can be applied in addition to such techniques, to obtain even more efficient policy decisions.

VI. Conclusions

We presented LbSprint, a middleware layer for privacy protection in location-based services. LbSprint supports and integrates different privacy protection mechanisms, basing on some user-customizable policies. The privacy rules are written in the standard XACML language. We used the XACML obligations to integrate different privacy mechanisms. Then, we presented a practical optimization, which considerably improves the performance of the XACML policy decision process.

Acknowledgment

This work has been supported by The EU FP7 Integrated Project PLANET (Grant agreement no. FP7-257649).

References

[1] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh, "Location-sharing technologies: Privacy risks and controls," in *In Research Conference on Communication, Information and Internet Policy (TPRC*, 2010.

[2] L. Ackerman and J. Kempf, "Wireless location privacy: A report on law and policy in the United States, the European Union, and Japan," DoCoMo USA Labs, Tech. Rep., 2003.

[3] L. Barkhuus, B. Brown, M. Bell, M. Hall, S. Sherwood, and M. Chalmers, "From awareness to repartee: Sharing location within social groups," in *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI '08. ACM, 2008, pp. 497–506.

[4] M. Benisch, P. G. Kelley, N. Sadeh, T. Sandholm, J. Tsai, L. F. Cranor, and P. H. Drielsma, "The impact of expressiveness on the effectiveness of privacy mechanisms for location-sharing," in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ser. SOUPS '09. ACM, 2009.

[5] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarca, J. Tabert, and P. Powledge, "Location disclosure to social relations: why, when, & what people want to share," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '05. ACM, 2005, pp. 81–90.

[6] E. Toch, J. Cranshaw, P. H. Drielsma, J. Springfield, P. G. Kelley, L. Cranor, J. Hong, and N. Sadeh, "Locaccino: A privacy-centric location sharing application," in *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, ser. Ubicomp '10 Adjunct. ACM, 2010, pp. 381–382.

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

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

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

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

[11] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 13–27, Jan. 2011.

[12] G. Dini and P. Perazzo, "Uniform obfuscation for location privacy," in *Proceedings of the 26th Annual WG11.3 Conference on Data and Applications Security and Privacy*, ser. DBSec'12. IFIP-Springer, 2012.

[13] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Proceedings of the 3rd International Conference on Pervasive Computing*, ser. PERVASIVE'05, H. W. Gellersen, R. Want, and A. Schmidt, Eds., vol. 3468. Springer Berlin / Heidelberg, 2005, pp. 152–170.

[14] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, ser. MobiSys'03. ACM Press, 2003, pp. 31–42.

[15] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia, "Privacy-aware proximity based services," in *Proceedings of the MDM 2009: 10th International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 2009, pp. 31–40.

[16] OASIS, "OASIS eXtensible Access Control Markup Language (XACML)," 2007. [Online]. Available: http://www.oasis-open.org/committees/xacml/

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

[18] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 56–64, 2003.

[19] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments," in *UbiComp 2002: Ubiquitous Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2498, pp. 237–245.

[20] A. Cooper and J. Morris, "An Architecture for Location and Location Privacy in Internet Applications," RFC 6280 (Informational), Internet Engineering Task Force, Jul. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6280.txt

[21] E. A. Crider and V. D. Francies, "Method and apparatus for electronically tracking luggage," U.S. Patent, May 2009.

[22] S. Gezici, "A survey on wireless position estimation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 44, pp. 263–282, Feb. 2008.

[23] World Wide Web Consortium, "Simple object access protocol 1.1 (SOAP1.1) specification," May 2000. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[24] C. Bettini, S. Mascetti, X. Wang, D. Freni, and S. Jajodia, "Anonymity and historical-anonymity in location-based services," in *Privacy in Location-Based Applications*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, vol. 5599, pp. 1–30.

[25] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, pp. 1–18, 2008.

[26] L. Marconi, R. Di Pietro, B. Crispo, and M. Conti, "Time warp: how time affects privacy in LBSs," in *Proceedings of the 12th international conference on Information and communications security*, ser. ICICS'10. Springer-Verlag, 2010, pp. 325–339.

[27] M. L. Damiani, E. Bertino, and C. Silvestri, "Protecting location privacy against spatial inferences: the PROBE approach," in *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, ser. SPRINGL '09. ACM, 2009, pp. 32–41.

[28] Sun, "Sun XACML implementation." [Online]. Available: http://sunxacml.sourceforge.net/

[29] Open Geospatial Consortium, "Geospatial eXtensible Access Control Markup Language (GeoXACML)," 2008. [Online]. Available: www.opengeospatial.org/standards/geoxacml

[30] A. X. Liu, F. Chen, J. Hwang, and T. Xie, "XEngine: a fast and scalable XACML policy evaluation engine," in *Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '08, 2–6 Jun. 2008, pp. 265–276.

[31] F. Turkmen and B. Crispo, "Performance evaluation of XACML PDP implementations," in *Proceedings of the 2008 ACM workshop on Secure web services*, ser. SWS '08. ACM, 2008, pp. 37–44.