Privacy-Preserving Location-Based Services by using Intel Software Guard Extensions

Vaibhav Kulkarni, Bertil Chapuis and Benoît Garbinato
{firstname.lastname}@unil.ch
Distributed Object Programming Laboratory
University of Lausanne, Switzerland

ABSTRACT

We are witnessing a rapid proliferation of location-based services, due to the useful context-aware services they provide their users. However, sharing sensitive location traces with untrusted service-providers has many privacy implications. Although, user-data monetization is the core economic model of such services, offering private services to concerned users will be a beneficial functionality in the coming years. Existing solutions include location perturbation, k-anonymity and cryptographic primitives that trade service accuracy or latency for enhanced user privacy. We introduce a novel approach for privacy preserving location-based services by using the Intel Software Guard eXtensions (SGX). We implement a simple location-based service using SGX and gauge its performance in terms of efficiency and effectiveness, in comparison with its bare-metal implementation. Our evaluation results show that SGX contributes a marginal overhead but also provides near-to-the-perfect results in contrast to spatial cloaking with k-anonymity whose performance deteriorates as the degree of desired privacy increases. We show that hardware-based trusted execution-environments are a promising alternative for offering proactive and de-facto location-privacy in the context of location-based services.

CCS Concepts

•Security and privacy → Privacy-preserving protocols;

Keywords

Location privacy; Intel SGX; Privacy-preserving LBS

1. INTRODUCTION

The ubiquitous nature of mobile phones equipped with internet connectivity and global positioning functionality (GPS) has led to the widespread development of location-based services (LBSs). Such services collect and store a large amount of user-location data in the untrusted cloud, which exposes users to several privacy risks. Data breaches and unlawful exchanges [2] can enable curious adversaries to derive personally identifiable user information (PII) by applying simple heuristics [6]. This information can be used for blackmailing or stalking purposes [1]. Thus, user privacy consideration will be a key factor in determining the success and adoption of LBSs in the coming years.

Service provider's (SP) application usually resides in the cloud where the user data is processed to yield the desired result. The SPs rely on virtual machines or containers to insulate the underlying platform from the users and to offer isolated execution. Although, such measures safeguard the SPs against the users, the latter have to implicitly trust the SPs and the execution platforms. Several solutions have been proposed to address the privacy concerns in LBSs, such as spatial cloaking [8], k-anonymity [7] and cryptographic primitives [5]. However, such techniques are not widely adopted in practice, either due to their low accuracy or high latency.

We propose an architecture for Private-LBS, it relies on Intel's next generation hardware-based trusted execution-environment called **Intel SGX**[1]. Intel SGX provides a reverse sandbox that enables independent software vendors to run a software module on an untrusted cloud. It designates a container that isolates the program and data from all the other software, potentially malicious OSs and the hypervisor. Furthermore, it offers a verification mechanism for authenticating the remote hardware platform and its state. We use these features to implement a **Point-of-Interest Locator** (POI-Locator) application that imposes anonymity and indistinguishability to enforce user privacy. We quantify the overheads involved in such a system, with respect to its bare-

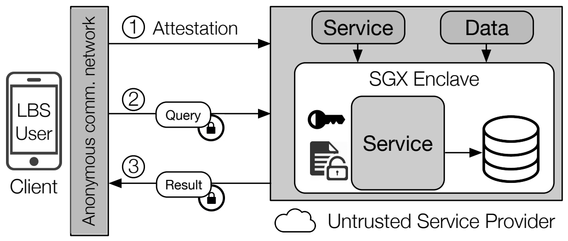

Figure 1: Private-LBS: The client can verify the application security by performing attestation. The application and database is embedded in the enclave. The query is encrypted, which can only be decrypted and processed inside the enclave. The result sent by the service-provider can only be decrypted by the client. Thus ensuring end-to-end-to-service-provider encryption.

HumanSys'17, November 5, 2017, Delft, Netherlands
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-5480-6/17/11…$15.00
DOI: https://doi.org/10.1145/3144730.3144739

[1]Intel SGX: https://software.intel.com/en-us/sgx

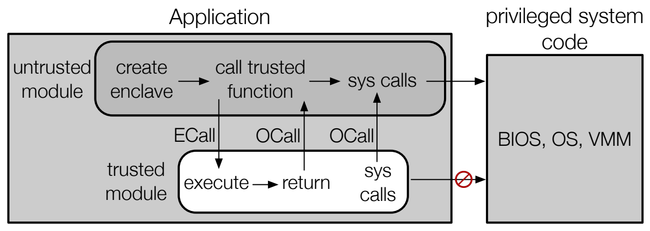

Figure 2: SGX application with trusted and untrusted modules

metal counterpart. We also compare its performance with a popular hybrid location-perturbation algorithm: spatial cloaking with k-anonymity. An architectural overview of our system is depicted in Figure 1.

The remainder of the paper is organized as follows. We discuss the background information necessary to illustrate our system in Section 2 and, in Section 3, the related work on location-privacy preservation is discussed. We present the system design and architecture in Section 4 and the system evaluation in Section 5. Finally, we conclude the paper is Section 6.

2. BACKGROUND

In this section, we present the background information regarding the features offered by Intel SGX. These features are leveraged in our system design to offer private location-based services.

2.1 Intel Software Guard eXtensions (SGX)

SGX is Intel's new architecture extension for providing a strong and provable isolation of binary code that runs concurrently and shares resources. It enables an application to construct protected regions of memory at a virtual address space called **enclave**. The enclave can be created and destroyed using certain privileged instructions. An application code and the data can be embedded into the enclaves and is ensured protection from the outside world. SGX provides guarantees that no privileged software, even with the root access, can view the contents of the enclave. Furthermore, all the contents belonging to the enclave that lie outside the enclave are encrypted. As an enclave has a limited size, we can create multiple enclaves that are isolated from one another and distribute data using shared keys.

As these features can also be used to create super malware, the enclave is prohibited from executing any privileged instructions, including systems calls and I/O operations. Additionally, the enclave code can only run in the user-mode and not in the kernel-mode. A typical SGX application consists of two modules: the untrusted module that executes security uncritical code and the trusted module that executes critical code inside the enclave as shown in Figure 2. These two modules communicate via two function calls: ECall (trusted) and OCall (untrusted). An ECall function enters an enclave and the OCall leaves the enclave. Therefore, an OCall is made every time the enclave wants to execute a privileged instruction. Evoking an OCall triggers the CPU to switch from the enclave mode to the user mode. The switching results in a certain overhead and can open up the enclave to various attacks. Thus, the OCalls for I/O operations are only used during the enclave debugging phase.

SGX also provides a remote attestation scheme to attest to the security offered by the untrusted cloud-provider. This

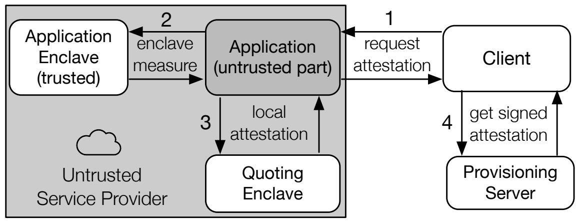

Figure 3: Remote attestation procedure

feature enables a remote user to verify whether an application is running inside a legitimate enclave and does not leak any information, thus, leaving only the processor operation and the security keys printed on the die to be trusted by the user.

Memory Encryption. All the enclave data and code is transparently encrypted in the memory by the SGX Memory Encryption Engine (MEE). The MEE uses a combination of Merkle trees and a 56-bit AES counter, producing a 128-bit integrity key and a 512-bit universal hash key to encrypt the enclave pages. The keys are generated at boot time and are placed in the privileged MEE registers that are destroyed at system reset. Thus, access to this protected memory region called Enclave Page Cache (EPC) is restricted at the hardware level. The Enclave Page Cache Map (EPCM) restricts the pages that the enclave is permitted to access in the EPC. The MME therefore also protects the data in the RAM from unauthorized access.

Remote Attestation. Typically, the clients have no assurance regarding the software running at the remote server. To address this, SGX implements the remote attestation mechanism that guarantees that the application is not tampered with, and is transparent about how the private user data is treated. Every enclave, when initialized, generates a certificate containing its measurement, vendor ID, product ID and other enclave attributes. The remote attestation procedure is depicted in Figure 3. During the initialization phase, the measurement of the enclave contents is taken by performing a hash over its memory pages. First, the enclave obtains a signed attestation for itself from a specific Intel enclave, known as the quoting enclave, through a local attestation procedure. The attestation is performed over the enclave's measurement to create a report. The quoting enclave checks the report and generates a signed attestation quote, that, intern is sent to the remote user.

When the remote user demands an attestation, a loader process which connects to the Intel's provisioning server is initiated. The purpose of the service is to verify the signed attestation quote of the enclave. Intel burns two secret keys into the CPU: a provisioning secret burnt during manufacturing and a sealing secret burnt at the boot time. The provisioning secret is shared with Intel for the attestation service, whereas the latter is not accessible outside the CPU. The provisioning service checks the key, derived from the provisioning secret to attest the enclave. In the case of successful attestation, a report is created and digitally signed, attesting that the CPU is indeed running in a secure mode where the memory is encrypted. The loader process now sets up a secure channel to the provisioning server and can download the intended software and data into the encrypted RAM for execution. The software module and the data can also be encrypted and saved to the disk. When the SGX-enabled processor connects to the provisioning server via the enclave, it also receives an attestation key, that can be sealed and stored to create further attestations. This reduces the entities to be trusted to only Intel's remote attestation service, as all other infrastructure is locked out by the encryption.

Sealing. As discussed above, when the enclave is instantiated, the MEE provides the data integrity and confidentiality. However, the enclaves are stateless, i.e., upon terminating the enclave process, the data stored within the enclave will be lost. Sealing is a special feature provided in order to store data outside the enclave, if the data is meant to be re-used at a later stage. When invoked, the data is sealed using persistent sealing keys derived from the CPU to encrypt and integrity-protect the data. The sealed data block can be unsealed either by the enclave that sealed it or by the software vendor, depending on the key used for sealing. This ensures the confidentiality, integrity and authenticity of the data.

2.2 SGX in Practice

SGX has been used to enforce privacy in the smart-grid infrastructure, to secure the energy consumption traces of users [12]. Here, to perform analyses over the user data, SGX is used as an intermediary entity between the smart metering devices and the SP. Only the meta records created by the intermediary entity are then sent to the SP for billing purposes. The security features offered by SGX have also been used to implement a content-based routing (CBR) engine inside the enclave [16]. CBR is not very popular as it necessitates routers to view the data in plain-text, which poses security threats. However, when message filtering is performed inside the enclave, the routers remain oblivious to the messages. [9] uses SGX to make secure two-party function evaluation more efficient as compared to traditional cryptographic operations that are too slow for practical applications. Along similar lines, [15] proposes the use of SGX for providing privacy guarantees for MapReduce operations. In all the above applications, it was found that porting the applications inside the SGX enclaves results in a superior performance compared to the cryptographic mechanisms. These results motivated us to exploit the privacy guarantees offered by SGX and apply it to a domain where privacy is of extreme importance: LBSs.

3. PRESERVING LOCATION PRIVACY

Research efforts in the domain of location-privacy preservation have been focused on enforcing two key principles: anonymity and indistinguishability. Anonymity is essentially decoupling the user identity from the location-traces so that the SP cannot link a location to a particular user. Indistinguishability prohibits the SP from distinguishing between the actual and fake locations sent by a user. In this section, we discuss the related work on location-privacy preservation in the context of LBSs.

A technique called spatial cloaking [8] was devised to perturbate the user's true locations. Here, an intermediary location broker is used, which transforms the actual user location along the spatial and temporal dimensions to satisfy certain anonymity constraints. The query containing the perturbed location is then sent to the SP, thus protecting the users identity and the location. Another commonly used mechanism for protecting user identity is called spatial-k-anonymity [7]. In this case, an intermediary server replaces the location of a user with a location lying in an anonymizing region consisting of at least k-1 other users. This makes the probability of attacking a certain user at most $1/k$. Such approaches dependent on an intermediary trusted-entity have several disadvantages, such as bottlenecks in communication and a single point of failure because all the sensitive user information is stored at a single central entity.

In order to address these limitations, decentralized approaches have been devised; they exploit the peer-to-peer network to eliminate the dependency on an intermediary trusted server. Here, the participating users form a peer group that is used to route their queries to the SP [4]. Peers are selected at random to forward the location traces. Another common technique is to send dummy locations, along with a true location, to enforce indistinguishability [11]. In the above cases, the extent of privacy protection and LBS accuracy is dependent upon the population and road density in the vicinity of a user's location. [14] solves the above problem by first projecting the 2-D location coordinate onto a Hilbert curve. Homogeneous noise is then added to the points on the curve to perturb the true user locations. The perturbed points are projected back to the 2-D space before sending them to the SP. The results show higher LBS accuracy and privacy guarantees compared to conventional location-privacy techniques. However, the dimensionality reduction and perturbation process leads to an increased latency.

Cryptographic techniques relying on homomorphic encryption are also used to access LBSs without directly revealing the location traces [5]. [17] proposes an efficient spatial range query algorithm over ciphertext; it protects user's query privacy and LBS data confidentiality in an outsourced cloud. Similar to the previous work, [13] enables different levels of queries on encrypted location traces, with the added benefits to operate on a mobile phone. Although the above contributions guarantee a high level of LBS accuracy, they involve a high latency and complexity due to the operations performed on ciphertexts. As the SGX based solution does not demand location perturbation or computing on ciphertext, it is implicitly more efficient than the existing solutions, and provides highly accurate services.

4. SYSTEM DESCRIPTION

In this section, we present our system model, the adversary model and the protocol design.

4.1 System Model

Our system model consists of the following three entities.

- The **Client** is the end user having a subscription to the LBS. In our application, the client sends her current location to the POI-Locator service provider to receive information regarding the nearby (user-configurable distance) points of interest (POI) that can include restaurants, pubs, cafes, gas stations etc.
- The **Service Provider** receives the client's current location, computes the nearby points of interest using a local database and returns the result back to the client. The result should include the name and category of the place, address and distance from the user's current location.
- The **Infrastructure Provider** hosts the POI-Locator application and provides the SGX-based machine to run the service provider's application: for example, a public cloud platform offering cloud services.

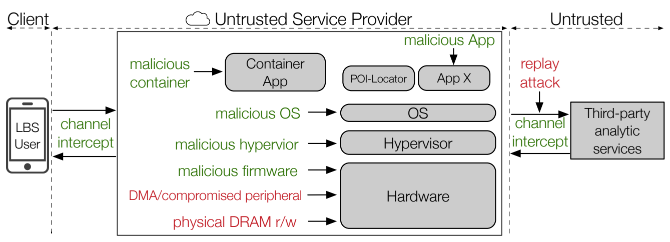

Figure 4: Adversary Model

4.2 Adversary Model

Considering the above system model, the adversaries from the client's perspective are both the SP and the infrastructure provider (IP). We also assume that the SP does not trust the IP. Hence essentially, the SP wants to keep its code confidential and the client wants to keep her location-coordinates private.

The hardware platform and the system software running at the client's end are assumed to be trusted. The IP is treated as untrusted and malicious or compromised, capable of executing any arbitrary software or modifying the OS or the bootloader. The attacker is assumed to be able to control all the privileged software, including the hypervisor, firmware and the entire management stack. As the resources in a public cloud domain might be shared amongst multiple SPs, we also assume that all the other services/applications running at the IP's end are malicious. The IP administrators are not trusted and are assumed to be curious or malicious. The SP is assumed to be honest but curious, i.e. the application always computes and returns correct results to its clients, however they can use the information regarding a user's identity and/or location traces for any kinds of activities. The SPs can also leverage analytical services from other third party entities such as Azure[2]. We also assume that such services are honest but curious.

Attack	Description
Denial of Service	Host machine physically taken off the network
Port attack	Malicious software running via the debug ports
Bus tapping attack	Tap motherboard bus's to track or modify traffic
Chip attack	Power and timing analysis to reverse engineer code
Side channel attack	Reverse engineering via performance monitoring
Cache timing attack	Learn correlation between memory access & time
Microcode attacks	Reprogram the machine code functionality

Table 1: SGX is vulnerable against above hardware attacks

We also consider that the processors equipped with the SGX functionality are to be trusted and an attacker is not capable to physically tamper with them. Figure 4 shows the possible attacks considered by our system. SGX provides implicit protection against the attacks marked in green but not those marked in red. Table 1 shows a detailed list of hardware attacks, against which SGX does not offer implicit protection. We do not consider these attacks, as they require physical access to the hardware, which is easier to detect in most cases.

4.3 System Design

Our system and protocol design focuses on the computation and communication security between all the entities involved.

[2]Microsoft Azure: azure.microsoft.com

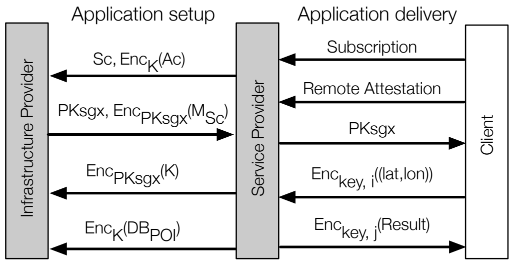

Figure 5: Communication protocol in POI-Locator

4.3.1 *Application Setup Phase*

In order to setup and launch the POI-Locator application, the SP first transfers a setup code, Sc to the IP. Sc is not confidential and can be sent in plain text. Next, the application code, Ac is encrypted with a confidential key, K and sent to the IP. Upon reception, IP will setup and instantiate an enclave and run the Sc. After running the Sc, a log called enclave's measurement, M_{Sc} is created, which attests that Sc is running in isolation within a legitimate enclave. This log is encrypted using the public key of the SGX core, $PKsgx$ and sent to the SP. The SP verifies the log with the provisioning server and sends the key K encrypted with $PKsgx$ back to the IP. After receiving the K, Sc can decrypt Ac and can initiate the application inside the enclave. The interaction between the SP and the IP is depicted in Figure 5.

Next, the database file containing the POI's is sent to the IP, encrypted with K. This file is later decrypted inside the enclave and is sealed using the enclaves TweetNaCl keypair[3].

4.3.2 *Service Provisioning*

Once the service is running, and before serving the clients the enclave generates TweetNaCl keypairs . Upon the successful generation of keypairs, the enclave outputs a public key that is provided to the client after subscribing to the service. This key can be used by the client to encrypt the requests and authenticate the results.

First, the client runs the remote attestation service to verify that the SP is running the promised program securely. The client views it as a public-key certificate, where the SP, along with the Intel provisioning server, endorses the application. After this step, both the parties have the enclave public key, $PKsgx$. We rely on the SGX re-encrypt[4] mechanism to establish a secure communication protocol between the client and the SP. In order to gain access to the service, the client sends a request with the current location and range encrypted with a key ID, i. Along with this message, the client also sends a key ID, j: it is the encryption key ID of the result to be returned by the SP.

The SP receives the request and decrypts the ciphertext using the key ID, i, providing the user location coordinates (lat, lon) and the range. The application then retrieves all the POI's lying within the range of the user's location, encrypts it using the key ID j and sends the result to the client. To view the result, the client decrypts the it using the the key j. All the plaintexts are encrypted using AES-GCM with 128-bit keys[5] and elliptic curve schemes over p256, which provides 128-bit security. In order to preserve anonymity, we rely on the anonymous routing component,

[3]TweetNaCl: https://tweetnacl.cr.yp.to/
[4]SGX re-encrypt: github.com/kudelskisecurity/sgx-reencrypt
[5]AES-GCM: tools.ietf.org/html/rfc5084

Enclave Task	Execution Time	Enclave Task	Execution Time
Create	22.41μsec	Copy (128 Bytes)	0.155μsec
Entry	0.752μsec	Seal (128 Bytes)	0.137μsec
Exit	0.631μsec	Keypair	13.445μsec
Encrypt (128 Bytes)	0.0154μsec	Hash (128 Bytes)	0.264μsec
Token	24.9944μsec	Quote	15.39μsec

Table 2: Micro-benchmarks of enclave tasks

Tor[6] at the client's end. This is simply achieved by using a Onion Proxy mobile client [7] to connect to the service provider; this mobile client uses a type of source routing to achieve communication anonymity between the client and the SP.

5. EVALUATION AND RESULTS

To evaluate the system performance, we base our results on a database of points of interest in Switzerland retrieved from the Open Street Maps[8]. Our implementation is run on a 64-bit, Intel 4-Core i5-7500 CPU clocking at 3.40GHz and running Ubuntu-16.04. We use the Linux 2016-06 SGX SDK[9], and the Enclave Page Cache was set to the maximum available size of 128 MB.

5.1 Benchmarking SGX Overhead

In order to quantify the overhead involved due to the SGX, we benchmark the latency of basic enclave operations. The execution time of enclave creation, enclave entry and exit (ECall and OCall), encryption, generating the keypairs, measurements and tokens, copying and sealing data is shown in Table 2. All the results are derived after taking into account the average and variance over 100 runs. We rely on SGX-log [10] to implement and quantify the latency of the micro-benchmarks.

The latency due to the enclave creation, copying and sealing is a one time cost involved during the service initialization phase. Every new client also has to bear the initiation cost of retrieving the measurement quote and generating the SGX public keypair. The other tasks, such as encryption, ECalls and OCalls, are recurring costs and contribute to the core of the overhead involved due to the SGX.

5.2 Bare-Metal Comparison

Here, we compare the overhead contributed by the SGX to the bare-metal implementation of the same application. We select a random coordinate lying within the POI dataset and select a range of 1000 meters in the query. These two parameters are kept constant for this evaluation. We quantify the overhead in terms of number of total instructions executed as the size of the POI-dataset increases, as shown in Figure 6. SGX results in a modest 10-12% rise in the number of total executed instructions. A majority of these additional instructions result due to transferring the execution between the enclave and the non-enclave modules. More specifically, the OCalls that the enclave has to initiate in order to execute system calls. Additional overhead is contributed by the instructions that need to be executed to encrypt and decrypt the array that contains the result and

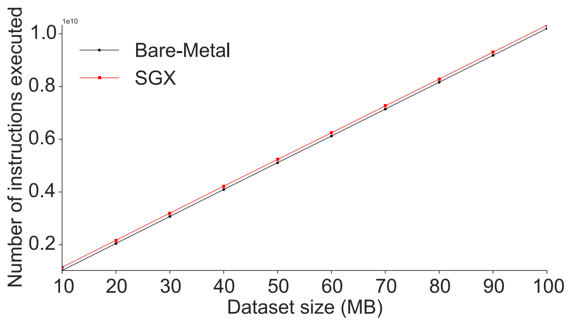

Figure 6: Comparison of number of total instructions executed

the user's location. However, these costs are marginal and do not lead to noticeable service delays.

5.3 Precision Comparison

Next, we compare our SGX-based approach to a popular location-privacy preserving technique: spatial cloaking with k-anonymity. The central idea of cloaking is to perturbate and anonymize the user's true location by creating cloaked regions. Spatial cloaking typically requires a trusted third party, called a location anonymizer, responsible for generating the cloaked regions. The anonymizer has to ensure that the cloaked region contains the number of users greater than or equal to k. Here, k refers to a privacy parameter that can be chosen by the user and corresponds to the desired degree of privacy.

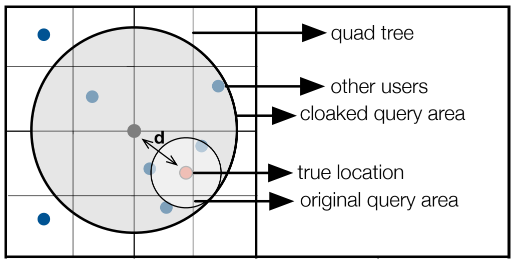

Figure 7: Spatial cloaking with k-anonymity

The POI-Locator application can be queried with a location and a desired range. Hence, for comparison, we assume a scenario wherein a set of queries are performed by different users who lie within the POI-dataset range. The cloaking implementation for this experiment is based on a simple spatial perturbation with k-anonymity technique described in [8]. The algorithm first indexes the locations of all the users in a quadtree. Given the location of a user, it then searches for the first cell that contains this location and less than k queries. The parent of this cell is guaranteed to contain a number of queries greater than or equal to k and is returned as the cloaked region. The algorithm then computes a new range by adding the distance between the initial location and the center of the cloaked region, d to the initial range specified by the user as shown in Figure 7. Thus, the center of the cloaked region and the cloaked range is used to send the query to the SP. The anonymizer does a good job at cloaking the user-location and range, however, this comes at a great cost in terms of precision.

In Figure 8, we show the relationship between the actual user query range and the cloaked query range as k increases. We consider three query ranges: 100, 500 and 1000 meters and as seen, the difference between the two sig-

[6]TOR Proxy: www.torproject.org
[7]Onion Proxy: www.torproject.org/docs/android.html.en
[8]OSM Switzerland: planet.osm.ch
[9]Linux SGX-SDK: https://github.com/01org/linux-sgx

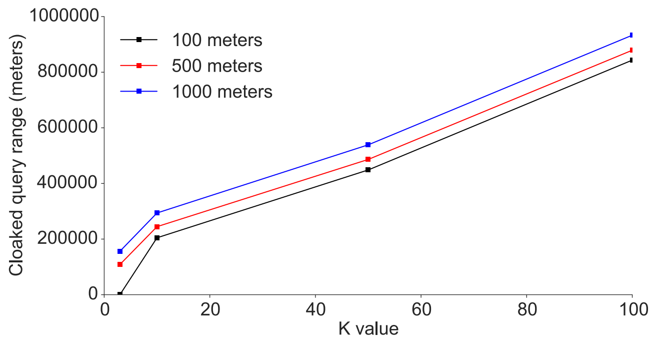

Figure 8: Relationship between query to clock range with k

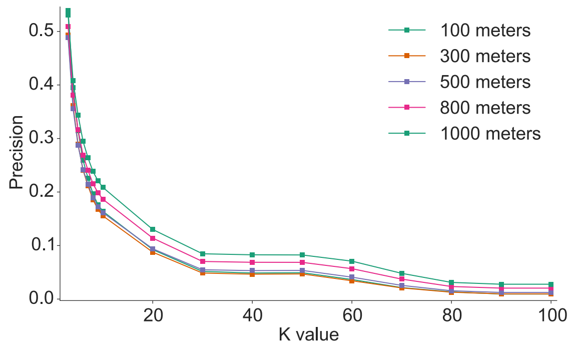

Figure 9: Relationship between the result precision and k

nificantly increases with k and leads to imprecise results. Furthermore, we observe in Figure 9 that as the privacy requirement (k) increases the precision lowers significantly with different query ranges. In this case, we define precision as the ratio between the number of POI's lying within the original range specified by the user from her true location ($true\,positives$) to the number of POI's retrieved by the SP ($true\,positives + false\,positives$). Note that measuring the effect of cloaking on recall is not relevant because the original queries and the cloaked queries return all the relevant results.

In conclusion, an approach based on SGX presents a clear advantage over an approach based on spatial cloaking with k-anonymity. Low precision has a great impact on the number of results returned by the LBS. This translates to higher computational and bandwidth requirements. In contrast, being able to return highly precise results, and guaranteeing privacy can make the LBS much more efficient.

6. CONCLUSION AND FUTURE WORK

In this paper, we have demonstrated the applicability of a hardware-based trusted execution-environment, i.e. Intel SGX to offer a privacy preserving location-based service. We implement a POI-Locator application using the security guarantees offered by SGX, adopting a privacy-by-design principle. We quantify the overheads involved due to the SGX implementation and compare it with the bare-metal execution. We show that SGX-based approach leads to a marginal overhead and provides near-to-the-perfect results. We experimentally show that SGX is a better alternative compared to popular location-privacy preserving approach: spatial-cloaking with k-anonymity, which has a detrimental impact on the precision as the degree of privacy increases.

Our current work, focuses on safeguarding only the user from the service provider. However, the user can retrieve the complete dataset from the service providers by submitting a large number of queries. This is critical when the services provider hosts a privacy-sensitive database. Our future work will address this issue in order to guarantee the privacy of both the parties involved. Furthermore, we will perform a complete evaluation of the SGX-based service, including the memory efficiency, responsiveness and usability to the clients, the number of simultaneous queries that can be handled and the network performance. We will also compare this approach with other well-known privacy-preserving approaches such as Private Information Retrieval (PIR) [3], in terms of accuracy and overheads.

Acknowledgement: This work is partially supported by the Swiss National Science Foundation grant 146714.

7. REFERENCES

[1] Uber privacy. www.usatoday.com/story/tech/2014/11/19/uber-privacy-tracking/19285481/.

[2] Uber starwood. www.forbes.com/sites/ronhirson/2015/03/23/uber-the-big-data-company/#4af73b4118c7.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS*, 1995.

[4] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *GIS*, 2006.

[5] Y. Gahi, M. Guennoun, Z. Guennoun, and K. El-Khatib. Privacy preserving scheme for location-based services. *J. Information Security*, 3:105–112, 2012.

[6] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez. Show me how you move and i will tell you who you are. In *SPRINGL*, 2010.

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

[8] 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*, pages 31–42. ACM, 2003.

[9] D. Gupta, B. Mood, J. Feigenbaum, K. Butler, and P. Traynor. Using intel software guard extensions for efficient two-party secure function evaluation. In *International Conference on Financial Cryptography and Data Security*, pages 302–318. Springer, 2016.

[10] V. Karande, E. Bauman, Z. Lin, and L. Khan. Sgx-log: Securing system logs with sgx. In *AsiaCCS*, 2017.

[11] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1248–1248, 2005.

[12] K. A. Küçük, A. Paverd, A. Martin, N. Asokan, A. Simpson, and R. Ankele. Exploring the use of intel sgx for secure many-party applications. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*, SysTEX '16, pages 5:1–5:6, New York, NY, USA, 2016. ACM.

[13] X.-Y. Li and T. Jung. Search me if you can: privacy-preserving location query service. In *INFOCOM, 2013 Proceedings IEEE*, pages 2760–2768. IEEE, 2013.

[14] A. Pingley, W. Yu, N. Zhang, X. Fu, and W. Zhao. A context-aware scheme for privacy-preserving location-based services. *Computer Networks*, 56(11):2551–2568, 2012.

[15] R. Pires, D. Gavril, P. Felber, E. Onica, and M. Pasin. A lightweight mapreduce framework for secure processing with sgx. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1100–1107. IEEE Press, 2017.

[16] R. Pires, M. Pasin, P. Felber, and C. Fetzer. Secure content-based routing using intel software guard extensions. In *Middleware*, 2016.

[17] H. Zhu, R. Lu, C. Huang, L. Chen, and H. Li. An efficient privacy-preserving location-based services query scheme in outsourced cloud. *IEEE Transactions on Vehicular Technology*, 65(9):7729–7739, 2016.