Towards Ensuring Client-Side Computational Integrity (A position paper)

George Danezis
Microsoft Research, Cambridge

Benjamin Livshits
Microsoft Research, Redmond

Abstract

Privacy is considered one of the key challenges when moving services to the Cloud. Solution like access control are brittle, while fully homomorphic encryption that is hailed as the silver bullet for this problem is far from practical. But would fully homomorphic encryption really be such an effective solution to the privacy problem? And can we already deploy architectures with similar security properties? We propose one such architecture that provides privacy, integrity and leverages the Cloud for availability while only using cryptographic building blocks available today.

1 Introduction

"*Yes, in this immense confusion one thing alone is clear. We are waiting for Godot to come.*"

– Samuel Beckett
Waiting for Godot

Cloud computing services promise scalable outsourcing of computations, networking and storage. Yet, offering such services as a commodity has met resistance due to privacy concerns. Users and consumer groups are increasingly sceptical of infrastructures that centralise personal data for processing.

Yet, the drive to store personal data in clouds is not likely to end. Social trends such as consolidation of medical records with sevices such as Microsoft HealthVault or Google Health, or consolidation of personal fiance data with Mint.com, as well as others see users record, store and process an increasing amount of data about their personal lives. At the same time traditional services, such as car insurance, electricity provision, road tolling, and taxation start relying on more fine-grained information about individuals lives and actions. So far the favoured architecture for these services has been a centralised repository or conduit of personal information.

Three key solutions have been proposed for resolving the privacy problem of processing personal information on clouds.

First, security policies can be applied to ensure only authorised entities and processes get access to the data. This is the prevalent model, and it relies on both physical security and correct access control. At the same time it is a brittle model: data is stored in clear and vulnerable to corrupt insiders, phishing attacks on system administrators (as it was the case with the Google Aurora attacks[1]), or a higher authority that can override the security policy.

A second solution involves trusted hardware at the servers or the clients to ensure the correctness of processing as well as the confidentiality of the data. Often Trusted Computing Modules (TPM) present on most modern motherboards and even mobile hardware are relied upon, but these are not safe against adversaries with physical access to the module. Robust secure co-processors, such as the IBM4758 are expensive and slow compared with a modern computer and using those to perform all computations would deny most benefits of cloud computing.

Finally, and from an academic viewpoint most interestingly, cryptography is presented as a solution to privacy concerns. Convincing solutions have been presented for secure storage of encrypted data [], as well as a restricted set of operations on encrypted data (such as searching an index [11]). The practicality of those is debatable on the basis of cost [7].

At the same time, fully homomorphic encryption seems to be hailed as the holy grail, that will "solve" the privacy problem [9, 16, 14]. An increasing number of publications at major cryptography conferences are looking at constructions for such schemes

[1] `http://en.wikipedia.org/wiki/Operation_Aurora`

or applications of such schemes to secure outsourcing of computations. In general these schemes promise the ability to perform computations on encrypted data. They decompose any computation to an equivalent logic circuit, and implement the basic gates in terms of the "plus" and "multiply" operations. The circuit results in a ciphertext encoding the result of the computation that is sent back to the user for decryption.

There are two key problems with this approach: first, no practical fully homomorphic encryption schemes exists yet [10]; second, as we will argue, *even* if fully homomorphic encryption was available at the cost of other cryptographic operations today, it would still be inefficient for most computations and could be replaced with a simpler architecture that is already realisable at a low cost today.

We devote the remaining of this paper in describing a cryptographic architecture that could be made available today to solve aspects of the problem of privacy in the cloud at a relatively similar cost as if homomorphic encryption was used. While the network overheads of the proposed approach will be higher, its advantage is that it can be deployed today.

1.1 Contributions

The contributions presented in this position paper are as follows:

- We argue that even cheap homomorphic encryption is not likely to solve the privacy problem in the cloud.
- We show that integrity and privacy can be cryptographically achieved today for a broad class of processing problems on personal data. These are based on efficient models of private computation.
- We argue that cloud computing can be used to leverage and enhance those technologies, in particular in terms of availability.
- We present two in-depth case studies where our architecture can be deployed to solve real world privacy problems. One relates to the recreational use of life logging. The other is a privacy-friendly high-value location based service, such as pay-as-you-drive insurance.

1.2 Paper Organization

The rest of this paper is organized as follows. Section 2 presents the problem of privacy for cloud services and examines the practicality of solutions based on homomorphic encryption. Section 3 frames a special but important class of services for which we provide a privacy solution. Section 4 outlines our privacy solution. Section 5 presents two case studies. Finally, Section 6 concludes.

2 Aspects of Cloud Privacy Problem

We consider the common architecture where personal data from multiple users is stored and processed on a cloud. The cloud could either belong to the service provider or be a utility cloud managed by a further third party — making the privacy and trust issue even more complex. Examples of such architectures include most large database systems, back end systems used in all branches of government, customer management systems, financial transaction systems, social networks, etc.

From the point of view of the service provider, centralising personal information provides availability and integrity. When the data is needed it is guaranteed to be at hand, since it is stored in clear at the provider. Furthermore, the correctness of any computations on the inputs is guaranteed to be performed correctly since it is performed on the provider's own infrastructure.

There are also strong non-security reasons for storing all data on the provider side. Unforeseen computations can be performed on it at any future time. The data can be "anonymized", sampled, aggregated and used for other back-end processes. Finally, the data is available to customise a user's experience on the server side (where traditionally web pages are rendered, or bills are printed).

From the user perspective, this architecture is convenient, but comes with privacy problems — namely the wholesale availability of personal data to a third party. Regulations are in place to prevent the wild sharing and use of this data in some jurisdictions, and industry initiatives such as privacy policies and seals are used to provide some comfort to users. Still data is misused, seized, lost or compromised regularly across the industry.

How is the application of fully homomorphic encryption going to solve this problem? Its proponents suggest that users will store on the cloud encrypted data items instead of plaintext. Computations on those data items will then be expressed as circuits and logical gates that can be implemented using the two ring operations allowed through the homomorphism on the ciphertext. That will result in a ciphertext encoding the output of the computation.

Even if one assumes the cryptographic operations are efficient, this deviates from the current model. First of all, a decryption operation must be performed on the resulting ciphertext for it to be of any use to the service provider. One could consider

sending the result back to the user for decryption – introducing an additional round trip and a dependency on the liveness and reachability of a user device. Additionally, the service needs to prove to the user that the result is indeed the correct output, namely the result of the correct function applied to the correct encrypted input. If the users decrypts ciphertexts without verifying their well formedness, their can simply be used as decryption oracles to decrypt any arbitrary ciphertext.

Another possibility for decryption is using a trusted third party holding decryption keys. This party is vulnerable to compulsion or abuse, it might have to rely on secure hardware, and would also have to check the well formedness of ciphertexts before decryption – making it expensive to run (i.e. as expensive as performing the computation or verifying it as we will see in our scheme).

We will show that for on-line service provision an alternative cryptographic architecture guarantees integrity and privacy with technology that is currently available [2].

3 Ingredients for a Solution

As we have discussed the problem of privacy in on-line services is linked with the issue of integrity of computations as well as availability of the data when it will be needed. Yet, a large number of on-line services share further characteristics:

- The users are entitled to all personal information about them, as information about the processing their personal data will be subject to. This is a key principle of data protection frameworks as implemented in the EU, Australia and Canada, and a key component to process data lawfully (*consent*) and fulfilling *subject access rights*. Technically, user data and the functions applied to it are not a secret to the users.
- Users are increasingly accessing on-line content on full computational platforms, such as modern web-browsers or smart phones. Furthermore, commonly those devices have good connectivity.
- Services want to detect early any problems on the user side that may prevent them from delivering a service. For example the lack of credit card records, or an account with certain privileges, or a faulty sensor that does not report data. They wish to detect such faults even before they use the personal data, e.g. for payment or analysis.
- Finally, services may process data in ways that was not envisaged when the data was collected, if subsequently the user has given consent for such processing. Personal data may also be aggregated, generalized or sampled in an effort to "anonymize" it, in ways and on dimensions that were not foreseen during collection.

We also note some rather established cryptographic results:

- The problem of secure storage on untrusted clouds to guarantee confidentiality and integrity is well studied, and satisfactory solutions exist [11]. Thus we can use the cloud for pure storage without worries.
- Relatively efficient cryptographic mechanisms exist for showing that a certain value is the result of a computation on certain inputs. Furthermore, zero-knowledge proofs allow for the inputs to remain secret – by only requiring commitments to be made public [8].

Given those observations we propose an architecture that can be practically realised today to address privacy concerns.

4 Overview

In our architecture information from sensors, third parties or volunteered by the user is can be collected and processed in a privacy friendly, way while preserving privacy and integrity. Figure 1 illustrates the data flows of our protocols.

First of all we assume that all parties in the system, the user, the cloud service, sensors and third parties producing data all have public / private key pairs both for the purpose of signing and verifying messages as well as encrypting and decrypting them.

First, we propose for data to be specifically packaged and "certified" for further cryptographic processing. Personal data (r_i) is encoded as a stream of cryptographic commitments ($\text{Commit}(r_i)$) such as Pedersen [6] or Groth [4] commitments. The binding property of commitments ensures the integrity

[2] Proponents of the applications of fully homomorphic encryption have proposed architectures where computations are simply outsourced to the cloud, and sent back to the users without the results ever being revealed. Even ignoring that any encryption is many orders of magnitude slower than a logic gate, evaluating a circuit takes always $O(N)$ computations in the size of the input at least, compared with many $O(\log N)$ algorithms used for searching indexes, ranking or sorting, etc. Furthermore, there will always be a need to check that any ciphertext is the result of the correct computation before relying on it. Yet, outsourcing computations is not the main focus of our argument — which is centred on service provision based on the processing of personal information.

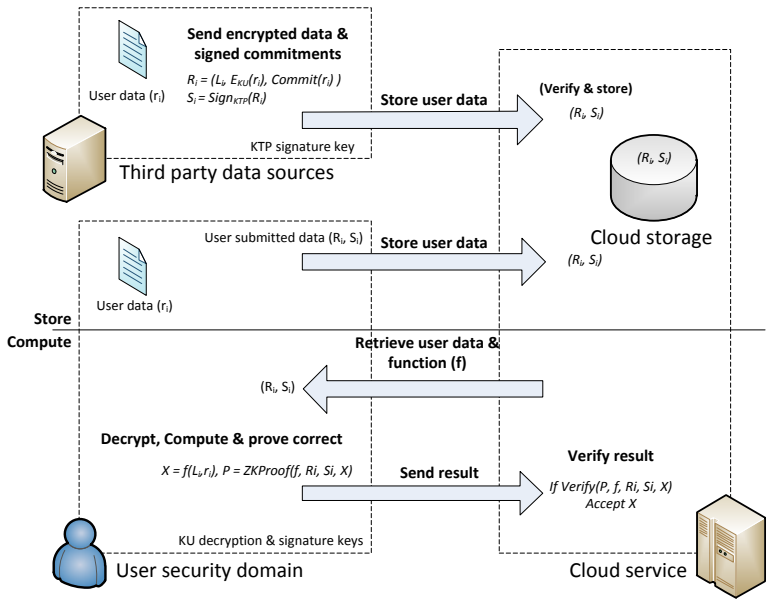

Figure 1: Flows of personal data to support client-side computations with integrity.

of committed values, while the hiding property ensures the confidentiality of the personal information. The personal data is encrypted under a key (K_U, public or symmetric) only known to the user, under any secure encryption scheme [11]. Commitments ($\text{Commit}(r_i)$), encrypted personal data ($E_{K_U}(r_i)$) and any public information (L_i) can be signed by the third party that generates the data under a key K_{TP} or the users themselves if the data is volunteered to a service[3]. The commitments, encrypted and public data (R_i) and signatures (S_i) are sent back to the provider and stored in the cloud.

Computations can be performed on the personal data items by relying on clients to download them, decrypt them, compute the necessary function, and send the results back to the service along with a zero knowledge proof of correctness.

More specifically, a user can download their personal information and the associated signatures denoted $R_i = (E_{K_U}(r_i), \text{Commit}(r_i), L_i)$ and S_i respectively. They can decrypt the this information using their key K_U, to recover the personal information r_i. Alongside any public information L_i, personal information can be used to compute any desired function $X = f(r_i, L_i)$. In parallel a non-interactive zero knowledge proof of knowledge can be built attesting correctness, namely $\exists r_i.X = f(r_i, L_i)$ and a valid signed commitment $\text{Commit}(r_i)$, without revealing anything further about the personal information r_i. We denote this proof $P = \text{ZKProof}(f, R_i, S_i, X)$.

Efficient non-interactive zero-knowledge proofs of knowledge have been studied for the past 15 years, and they most notably allow:

- Prove at a value is in fact part of a commitment, or multiple values are part of a commitment [13].
- Lineal algebra involving committed values and constants [2].
- Proving equality [13] and inequalities [3] amongst committed values.
- Proving logical statement with AND, OR and NOT connectives on predicates applied to committed values [2].
- Multiply committed values [3].
- Division and modular operations relating committed values [5].
- Efficient set membership [3] and look-up operations on committed keys-value pairs [12].

The set of operations allow us to express any computation as a circuit and prove its correctness, as for the fully homomorphic encryption case. Furthermore for many common special computations more efficient proofs are available – which we leverage in our case studies.

[3]Even user generated data can have value: a user being able to prove that an amount provided for their revenue is the same as the figure provided to the tax authorities would be of great interest to many services.

Finally, the client sends back to the service the computed value X along with the proof P. The service can verify the correctness of the calculation before accepting it.

Discussion. The proposed architecture achieves integrity for computations on private data, while maintaining their confidentiality by performing computations on computing resources controlled by users. Yet, those computations can be verified preventing users from cheating the service. Putting a user in the loop is a departure from the current architecture, but is no different from a security standpoint to implementing a similar architecture using fully homomorphic encryption if it ever becomes practical. A fully homomorphic scheme would require both interaction with the users, as well as a proof that the computations were performed correctly.

Cloud services are leveraged to by our architecture in key ways: first, encrypted data is available to any user device to perform the privacy friendly computations. All that needs to be migrated from device to device is the decryption key. Second, the service provider can verify the availability and even validity of the encrypted data at the time of collection, ensuring that an alarm is raised as soon as possible if expected data items are missing or invalid. Finally, the collection and verification of results are offloaded to the cloud making the approach scalable to large numbers of users. Further, processing of the results of private computations can be done on the cloud or any back end system.

5 Case Studies

Much of the inspiration for the two case studies in this section comes from the idea of *living by numbers*, i.e. that our day-to-day existance can be recorded using a variety of sensors, analyzed, uploaded, shared, etc. Examples of this often include health and wellness applications, recording one's day-to-day behavior to share with other, both trusted and untrusted entities, etc. The reader is referred to a special issue of the Wired magazine for more details [17].

In many living-by-numbers scenarios, both the integrity and the privacy of the data that is being collected is of interest. The user might have an incentive to forge their driving records to get lower insurance rates. Or the user might check in at fictitious exotic locales to bump up their Facebook social status. At the same time, the same user might be uncomfortable sharing this, either forged or genuine data, with various parties. We illustrate some of these issues with the two representative case studies below.

5.1 Training Regimes for Athletes

As sensors are increasingly becoming part of everyday life, more and more data about the user's daily behavior becomes available. In some scenarios, this data remains local to the user. An example of this is the DigiFit device[4], which supports a heart rate monitor, a pedometer, a cycling speedometer, a sleep monitor, as well as a set of other excercise and wellness sensors supporting the Ant+ protocol. The data is aggregated and processed, although this is done locally, on a mobile device such as an iPhone. Much of the time, While many health and wellness scenario have important privacy implications, the user has no incentive to forge data as they are themselves the primary data consumer.

However, when data is shared with third parties (such as insurance companies for the purpose of getting a lower insurance rate due to a "healthy lifestyle" bonus) or other individuals (as in the case of fellow dieters in an online Weight Watchers support group), the *integrity* of the data can sometimes be suspect.

To elaborate one particular scenario, imagine an athlete who is training to a major competition such as the Olympics. As part of her regimen, she has to report her heart rate at 15 minute intervals and distances she walks daily. This information can be collected by a bracelet-like device she keeps on her wrist, which is assumed to be unforgeable and will detect when it is removed, helping to mitigate the integrity threat by an unscrupulous athlete who wants to lie about her daily workout routine, for example. The threat to the athlete's privacy is present as well: knowing where the athlete lives in combination with location maps (background data) might allow her coach to understand where she goes and when. Moreover, frequent pulse data might be used to infer the time of, or type of, romantic activity that occurs when she visits her boyfriend's house, for instance.

In reality, all that is required may be an aggregate measure of her pulse data to ensure that her pulse has been above 120 for at least 1 hour every day and that she has walked or ran at least 5 miles. Currently, such applications seem to be blissfully unaware of privacy and integrity implications of what they are doing, as examplified by TrainingPeaks software[5].

[4] `http://www.digifit.com`

[5] `http://home.trainingpeaks.com/`

5.2 Automatic Car Insurance Tolling

Another scenario comes from a case of mileage metering to pay for auto insurance, already examined in [15, 1]. One of the emerging schemes involves paying a rate that is prorated depending on the number of miles driven, either linearly, or using several brackets. Of course, given that your insurance company knows much about you, including your address, much can be inferred given precise daily mileage data. One such insurance provider, MileMeter.com says the following on their site:

> **How do you know how much to bill me?**
> We still respect your privacy, and don't use invasive and expensive electronic tracking devices to report on your mileage. Instead, we trust our customers and count on you to help us keep your rates as low as possible. As of September 14th, when you purchase or renew a policy, we'll ask for a digital photograph of your car's odometer, with your driver license visible in the photograph. We realize this is a new step we're asking you to take, but gathering this information helps MileMeter bill you more accurately and efficiently – and we can pass the savings on to you!
>
> **When do I have to provide a photo?**
> We'll ask for a photo when you first purchase, and when you renew every six months. We also may ask you for another photo at a random time for statistical purposes. You do not have to provide a photo to get a quote!

However, this solution is clearly lacking: MileMeter appears to be clearly aware of fraud possibilities and tries to walk the fine line between accusing their customers of being untrustworthy and blindly relying on customer-provided data.

A solution that would provide both integrity and privacy for collected data is may involve a hardware device installed in the car (not dissimilar to a typical automotive transponders used in cars for recording tall charges). The device would detect basic kinds of fraud such as device removal or tampering. It would report the amount that the driver needs to be billed for, computed using daily mileage readings and a formula that is provided by the insurance company.

6 Conclusion

This paper proposes a practical strategy that may be used to achieve both confidentiality and integrity on the client, for many important classes of computation. We point out problems with the fully homomorphic encryption approach and offer a more immediate solution that in our experience meshes well with real-world scenarios, such as the two case studies we present. We have already built and deployed systems that implement this privacy design pattern for the case of smart metering and billing [12], and we argue that it is applicable very widely to solve issues of privacy in the cloud.

References

[1] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens. Pretp: Privacy-preserving electronic toll pricing. In *USENIX Security Symposium*, pages 63–78. USENIX Association, 2010.

[2] S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT*, pages 318–333, 1997.

[3] J. Camenisch, R. Chaabouni, and A. Shelat. Efficient protocols for set membership and range proofs. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[4] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In C. Blundo and S. Cimato, editors, *SCN*, volume 3352 of *LNCS*, pages 120–133. Springer, 2004.

[5] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT*, pages 107–122, 1999.

[6] D. Chaum and T. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *LNCS*, pages 89–105, 1993.

[7] Y. Chen and R. Sion. On securing untrusted clouds with cryptography. In E. Al-Shaer and K. B. Frikken, editors, *WPES*, pages 109–114. ACM, 2010.

[8] M. Dworkin. Cryptographic protocols of the identity mixer library, v. 2.3.0. IBM research report RZ3730, IBM Research, 2010.

[9] C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.

[10] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[11] S. Kamara and K. Lauter. Cryptographic cloud storage. In R. Sion, R. Curtmola, S. Dietrich, A. Kiayias, J. M. Miret, K. Sako, and F. Sebé, editors, *Financial Cryptography Workshops*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer, 2010.

[12] A. Rial and G. Danezis. Privacy-preserving smart metering. Technical Report MSR-TR-2010-150, Microsoft Research, November 2010.

[13] C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[14] N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[15] C. Troncoso, G. Danezis, E. Kosta, and B. Preneel. PriPAYD: privacy friendly pay-as-you-drive insurance. In P. Ning and T. Yu, editors, *Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society, WPES 2007*, pages 99–107. ACM, 2007.

[16] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[17] Wired Magazine. Living by numbers. *Wired Magazine*, July 2011.