A Mechanism Design for Privacy-Preserving Computation on Shared Data

Saffija Kasem-Madani[1]

Abstract: In times of surveillance and data retention, sharing information often comes together with privacy concerns. However, information sharing has benefits, e.g. sharing log files for including the knowledge gained from a broader view for security analysis, or sharing healthcare data for the use in studies for improving medical treatments. We present an information sharing framework design that meets both privacy and utility requirements of the information sharing parties. We utilize homomorphic encryption and show how it can be used for offline data analysis.

Keywords: encrypted computing, homomorphic encryption, multiparty computation, information sharing, privacy, utility.

1 Introduction

When information is shared among multiple parties, the interests of the data holders in keeping cleartexts confidential for privacy collide with the interests of the data receiver in maximal utility.

Consider the following application scenario: Data holders want to share log files (e.g., see [SY04]) with a centralized analysis entity, called data analyzer. The data holders are interested in the results of the analysis the analysis entity performs. The analysis entity profits from collecting log files from multiple sources because it gives him a broader view on the field, e.g. a business area.

On the other hand, data holders want to keep their data confidential. One reason is the privacy of the data owners. This contradicts the data analyzer's interest in data utility.

In this work, we assume that the data holders and the data analyzer formulated a compromise between the need of privacy and utility, i.e. a privacy-utility trade-off, in contracts called policies. A policy consists of conditions the exchanged data has to fulfill in order to meet all utility options that meet both the analyzer's and the holders' requirements. Utility options formulated in a policy may include the ability to calculate overall sums of numerical data for statistical analysis, or disclosing IP addresses in a log file under certain conditions, or checking equality of encrypted data. The conditions stated in a policy must be selected very carefully to ensure that the privacy of the data owners stays preserved. For a formulation of such policies, we refer to [KMM15].

Before sharing the data with the analyzer, the data holder must transform the data such that it will meet the policy. We call the result of such a transformation *data appearance*.

Depending on the use case and the sensitivity of the data under consideration, the selection

[1] Rheinische Friedrich-Wilhelms-Universität Bonn, Institut für Informatik, Friedrich-Ebert-Allee 144, 53113 Bonn, kasem@cs.uni-bonn.de

of the provided utility still must keep so many information confidential that an attacker is not be able to use the utility properties of the data appearances for information linkage and correlation attacks with the help of external information.
One approach that can be used is homomorphic encryption. In homomorphic encryption, the cleartext data p_1, p_2 are encrypted with an encryption mechanism E to $E(p_1), E(p_2)$. E ensures that certain operations can be performed on $E(p_1), E(p_2)$ and are equivalent to operating on the plaintexts p_1 and p_2. I.e., $E(p_1) * E(p_2) = E(p_1 + p_2)$ for appropriate operations $*$ and $+$. Note that the result of the encrypted computation is encrypted.
In the application scenario described above, the data holder would use a homomorphic encryption mechanism to encrypt the cleartext content of the log files to data appearances. After receiving the data appearances, the analyzer performs the agreed operations on the data for analysis. To check the operation results, the analyzer needs to decrypt them. This usually requires knowledge of the secret (private) key. However, having access to the private key enables the data analyzer to decrypt the content of the data appearance and hence, violating the privacy of the data owners.
To overcome the described problem, we present a mechanism that ensures that only results of allowed computations can be disclosed (i.e. decrypted) to the analyzer. We describe the parties of the mechanism and the design assumptions made for ensuring the privacy preservation of the shared data. The mechanism utilizes Paillier, a public-key probabilistic homomorphic encryption scheme [Pa99]. With the prospected upcoming improvement of homomorphic encryption schemes[3], we aim at providing feasible solution for its use for privacy preserving information sharing.

2 Description of the Mechanism

We describe the stakeholder of the mechanism, how they receive the data and metadata, and privacy-preserving decryption of results of operations performed on the Paillier encrypted data.

2.1 Stakeholders

For privacy-preserving information sharing for data analysis, we consider a mechanism that consists of the data holders, the data analyzer, the activator, the private-key holder, the preprocessor, and the decryptor. The stakeholders ensure that no single party gains more information than stated in the privacy-utility policy. The stakeholders of the mechanism are described as follows:

- The **data holder** has full access to plaintext data. His purpose is to share the data with the analyzer. He creates data appearances of Paillier-encrypted data. There can be multiple data holders.
- The **data analyzer** receives the data appearances from the data holders. He performs operations over the encrypted data contained in the collected data appearances.

[3] See e.g. the HEAT project https://heat-project.eu/

- The **preprocessor** holds the public keys necessary for completing the operations instead of the analyzer. This limits the types of operations the analyzer can perform.
- The **activator** has the main part in ensuring that only analysis results are disclosed to the analyzer. For this purpose, he checks for the validity of the encrypted results against a database of all hash values of encrypted components of data appearances. If the results are valid, he activates the decryption process.
- The **private-key holder** holds all private keys of all data appearances for the decryption process. The private keys are bound to certain data rather than being bound to a subject.
- The **decryptor** decrypts the encrypted operation results and sends them to the analyzer.

2.2 Preparation and distribution of data

After stating the policy, i.e. agreeing on the utility that the content of the data appearances of a data holder has to fulfill, the data holder initiates the data sharing process. He sends the data appearance D and and identifier id to the analyzer, enabling it to perform analysis operations as stated in the policy. The identifier id is used later to identify the correct public and private key for the decryption steps. Depending on the utility conditions stated in the policy, it may necessary to use more than one key pair. In this case, the data holder defines an order of the keys and assigns each key pair one hash value as follows: the first key pair is assigned the hash value $h(id)$. The second pair is assigned the hashed hash $h(h(id))$, the third pair is assigned $h(h(h(id)))$, and so on. We refer to an $i-$th hash $h_i(h_{i-1}(\cdots h_2(h_1(id))))$ with $h_i(id)$.
After generating the data appearance according to the policy, the data holder sends it to the analyzer. The analyzer can perform analysis operations on the data appearances.
The data holder computes hashes $h(d)$ for each encrypted item in the data appearance D and a hash value of the appearance's identifier, $h(id)$. D contains m encrypted items. Then he sends the set of hashes $\{(h(id), h(d_i) | i \in \{1, ..., m\})\}$ to the activator. If multiple key pairs are used, $h(id)$ is altered with the corresponding hash value.
Then, the data holder sends the set of all the private keys together with the corresponding hashed identifiers $\{(k_{priv,i}, hash_i(id)) | i \geq 0\}$ to the key holder. The private-key holder identifies the correct keys and send them to the decryptor for decryption.
Then, the data holder sends all public keys together with the hashed identifiers to the preprocessor, as described for the private keys. I.e., the preprocessor receives a set $\{(k_{pub,i}, hash_i(id)) | i \geq 0\}$. The preprocessor can now perform the preprocessing of the encrypted operation results received by the data analyzer.
Note that the keys are bound to certain data, not to subjects. If multiple data holders are involved in an information sharing action, they may agree on using the same keys for encrypting certain values for allowing the analysis of aggregated data from multiple sources.
After the data distribution, each of the stakeholders is prepared for taking part in the decryption process without further interaction with the data holder.

2.3 Data analysis

In this work, we are considering data that is encrypted with the Paillier cryptosystem. Being partially homomorphic, it is possible to perform homomorphic additions of ciphertexts. The public key is required for calculating the correct modulus of the added or multiplied ciphertexts. When the public key is given, it is possible to multiply and add a cleartext to an encrypted value. We omit the mathematical explanations here and refer to the original reference [Pa99].

2.4 Decryption of the operation results

The data analyzer perform the analysis operation op on some $d \subset D$, yielding $enc'(op(d))$, where $enc(op(d)) = enc'(op(d))modn^2$, n is one component of the public key. After performing the required operations on the encrypted data, the analyzer wants to know the cleartext values of the results for interpretation and further processing. In order to complete the calculation, he requests the preprocessor for applying the correct modulus calculation. For this, he sends a request that contains the partially calculated result and the hashed identifier for the public key $\{enc'(op(d)), h(id)\}$. With this, he initiates the conditional decryption process. I.e. the operation result will be decrypted if the data analyzer can show that he is not requesting for the decryption of data appearance content.
Once the preprocessor has received the request for preprocessing, he selects the correct public key, calculates the modulus and applies it to $enc'(op(d))$ with the result $enc(op(d))$. Then he requests the activator for decryption. For this, he sends a message that contains the encrypted result of the operation together with the hashed identifier of the key $\{enc(op(d)), h(id)\}$.
The activator checks whether the hashed value of $enc(op(d))$ is in the database. If this holds, he sends a ”FIN” message to the preprocessor, denying the possibility of decryption. The preprocessor sends the ”FIN” message to the analyzer.
If the activator finds no matching entry in the database, he assumes that $enc(op(d))$ is an encrypted operation result and not contained in the data appearance. This can be assumed due to the use of Paillier, a probabilistic encryption scheme.
The activator then creates randomly selected nonce n and sends it to the analyzer. The activator sends one message that contains $h(id)$ and the nonce n to the private-key holder, and one message $\{enc(op(d)), n\}$ to the decryptor.
On receiving $\{n, h(id)\}$, the private-key holder selects the private key that corresponds to $h(id)$ and sends it together with the nonce n to the decryptor. The decryptor receives the key, identifies it as belonging to $enc(op(d))$ using the nonce and decrypts the analysis result. The decryptor then sends the decrypted analysis result $op(d)$ together with the nonce n to the data analyzer.

2.5 Security assumptions and analysis

In the mechanism, we assume the honest-but-curious adversarial model [PMB]. I.e., we assume that the attacker is curious about knowing the cleartexts of messages she sees,

but follows defined protocols and would not break any security mechanisms. Further, we assume that the activator would not collaborate with the private-key holder to decrypt the data. Nor would he send the hashed identifiers to the decryptor to let him reconstruct which key has been used for a certain dataset. We also assume that the neither the activator nor the private-key holder would collaborate with the data analyzer to decrypt components of data appearances. As a result, the activator would neither learn the analysis results nor the cleartexts of the data appearances content. Due to the use of nonces that do not depend on the data indices, i.e. the hash values, for addressing the decryption part of the process, the decryptor would neither be able to learn to which dataset a decrypted analysis result belongs, nor would he be able to map an already received key to a future ciphertext for decryption. If he would store each private key he receives and collaborates with the data analyzer for disallowed decryption, he would be, at most, enabled to perform a brute-force attack on data items using all the keys.
We assume that the analyzer is not able to perform *misused operations* on the data. With misused operations, we mean adding constants homomorphically to the encrypted data. In Paillier cryptosystems, this requires access to the public key. Due to the probabilistic property of the Paillier cryptosystem, the activator would identify such forged ciphertexts as results of valid computation and, hence, activate the decryption process. Once the analyzer has received the decrypted result, he would get the cleartext data by subtracting the previously homomorphically added constant. In order to prevent the analyzer from performing misused operations, the public key is not made accessible to him. Instead, we introduced the preprocessor who holds the public keys and is assumed not to collaborate with the analyzer for misuse operations.
Assuming the honest-but-curious adversarial model in the current mechanism design may not be suitable for very sensitive use cases. To serve more stringent adversarial models, the tasks of the stakeholders may be redistributed based on secure multiparty computation schemes [Go98]. This would ensure that an attacker cannot successfully misuse the mechanism to gain disallowed information unless all parties actively collaborate with him.

3 Related Work

Homomorphic encryption schemes are a well-studied field [RSA78] [Pa99] [Ge09]. Performing operations on encrypted data for privacy preservation is an actual research field. Mechanisms for machine learning [GLN13] [Bo15] and recommending [Er12] on encrypted data have been presented. Mechanisms for secure multiparty computations [Go98] are widely used and have been adapted to threshold homomorphic encryption [CDN01].

4 Conclusions and Future Work

We presented the design of a mechanism for privacy-preserving decryption of data analysis results on homomorphically encrypted data. We described the stakeholders and their capabilities, explained the roles of each stakeholder, how he data and metadata is distributed and how the mechanism performs a privacy-preserving decryption. We performed a brief

security analysis. In order to weaken the trust assumptions made, new parties and a secret sharing mechanism [Sh79] may be used.
Implementing and evaluating the mechanism would show the real-world applicability of our mechanism. One may investigate the applicability of the mechanism for different homomorphic encryption mechanisms. The use of threshold homomorphic encryption [CDN01] may lead to a simplified mechanism for data analysis. However, the design of practicable threshold homomorphic encryption is still an open research problem.

References

[Bo15] Bost, Raphael; Popa, Raluca Ada; Tu, Stephen; Goldwasser, Shafi: Machine Learning Classification over Encrypted Data. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014. 2015.

[CDN01] Cramer, Ronald; Damgård, Ivan; Nielsen, Jesper B.: Advances in Cryptology — EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, chapter Multiparty Computation from Threshold Homomorphic Encryption, pp. 280–300, 2001.

[Er12] Erkin, Z.; Veugen, T.; Toft, T.; Lagendijk, R.L.: Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing. Information Forensics and Security, IEEE Transactions on, 7(3):1053–1066, June 2012.

[Ge09] Gentry, Craig et al.: Fully homomorphic encryption using ideal lattices. In: STOC. volume 9, pp. 169–178, 2009.

[GLN13] Graepel, Thore; Lauter, Kristin; Naehrig, Michael: ML confidential: Machine learning on encrypted data. In: Information Security and Cryptology–ICISC 2012, pp. 1–21. Springer, 2013.

[Go98] Goldreich, Oded: Secure multi-party computation. Manuscript. Preliminary version, 1998.

[KMM15] Kasem-Madani, Saffija; Meier, Michael: Definition of Availability Policies for Data Pseudonymization Using XACML. IFIP Summer School on Privacy and Identity Management, August 2015.

[Pa99] Paillier, Pascal: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In (Stern, Jacques, ed.): Advances in Cryptology, EUROCRYPT 99, volume 1592 of Lecture Notes in Computer Science, pp. 223–238. Springer Berlin Heidelberg, 1999.

[PMB] Paverd, AJ; Martin, Andrew; Brown, Ian: Modelling and Automatically Analysing Privacy Properties for Honest-but-Curious Adversaries. Technical report.

[RSA78] Rivest, Ronald L; Shamir, Adi; Adleman, Len: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[Sh79] Shamir, Adi: How to Share a Secret. Commun. ACM, 22(11):612–613, November 1979.

[SY04] Slagell, Adam J.; Yurcik, William: Sharing Computer Network Logs for Security and Privacy: A Motivation for New Methodologies of Anonymization. CoRR, cs.CR/0409005, 2004.