Methods to Secure Services in an Untrusted Environment

Matthias Huber, Jörn Müller-Quade

matthias.huber@kit.edu, muellerq@kit.edu

Abstract: Software services offer many opportunities like reduced cost for IT infrastructure. They also introduce new risks, for example the clients lose control over their data. While data can be secured against external threats using standard techniques, the service providers themselves have to be trusted to ensure privacy. In this paper, we examine methods that can increase the level of privacy a service offers without the need to fully trust the service provider.
Keywords: services, cloud computing, security, privacy

1 Introduction

Due to advances in networking and virtualization technology, new paradigms of providing IT infrastructure and software have emerged – among them the so-called *Cloud Computing*. The National Institute of Standards and Technology [NIS09] defines Cloud Computing as "a model for enabling convenient, on-demand network access to a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud Computing enables clients to use *Software as a Service* and to outsource their data, thus cutting the cost of maintaining an own computing center.

Inherent to Software as a Service, however, are privacy problems [Pea09, HHK+09]: By using services, clients lose control over their data. The threat copy or misuse of the clients data on the server exists and keeps many potential customers from using Cloud Computing in critical or sensitive scenarios (e.g., scenarios comprising business secrets or customer data). Current security mechanisms focus on protecting the data transfer to and from the service provider.

Protecting a pure storage service against insider attacks is easy. Encrypting all data on the client before uploading it to the server provides a sufficient level of protection based on the used encryption [Bla93]. This, however, prevents the server from performing any meaningful operation on the data. Hence, more complex services require advanced techniques for providing privacy.

In this paper, we will examine techniques and methods that potentially can increase the security of services. In Section 2, we will describe the required properties of solutions. We will examine different methods in Section 3 and evaluate them according to the requirements described in Section 2. We will examine and evaluate applications of these methods in secure database outsourcing scenarios in Section 3.3. In Section 4, we will examine

potential candidates for formal security properties for services. Section 5 concludes.

2 Requirements

In this chapter we discuss the required properties of solutions for the privacy problems inherent to most services. Basically, solutions for the presented problem should provide *provable security* and *practicability*.

2.1 Requirement 1: Provable Security

Formal notions and proofs are provided for cryptographic methods. While providing security in the sense of classical cryptography is infeasible for nontrivial services (e.g. services more complex than pure storage services), solutions for the privacy problem described in Section 1 should provide provable security. This means that we need a formalization of the level of privacy provided. Furthermore, we need to prove that a solution provides this level of privacy (fulfills the formal notion). For a database service such a weaker, yet probably sufficient level of privacy may be that an adversary cannot learn the relations between the attribute values.

2.2 Requirement 2: Practicability

There are cryptographic solutions for problems involving parties that do not fully trust each other. In most cases, however, they are inapplicable due to their complexity. We strive for solutions that are practical and do not cancel the benefits of outsourcing. Consider a solution providing a high level of privacy, but due to its complexity it is more efficient to execute the service on the client. It will be hard to convince customers to use such a service even it provides privacy. Also the provided level of privacy has to be easy to understand, since this is vital to the acceptance of the solution, too.

3 Methods to Enhance the Security of Services

There are methods from different fields, that potentially can enhance the security of services. In this section, we will present methods from the field of software engineering and cryptography. In the field of secure database outsourcing there are applications of these methods. We will also present these applications in this section.

3.1 Software Architecture

There are approaches that enhance the security of services by separating it, and deploying the parts of different servers [HILM02, ABG+], assuming the servers do not cooperate maliciously. There are two different ways to separate a service, namely *serial* and *parallel* separation [Hub10]. For concrete services these separations have different performance implications depending on the usage profile. In most cases, however, a particular separation for itself does not yield security guarantees. Nevertheless, we believe a separation of a service in combination with cryptographic methods can yield provable security with feasible performance overhead.

3.2 Cryptographic Methods

Aside from *encryption* which is uses in many scenarios to prevent eavesdroppers from learning the anything about the information sent over a channel, cryptography offers additional methods and protocols to engage privacy and security issues. In the remainder of this section, we will discuss these methods.

There are encryption schemes that produce ciphertexts with homomorphic properties: Consider for example Textbook-RSA [RSA78]. Multiplying two ciphertexts and decrypting the result yields the same result as decrypting the two ciphertexts and multiplying the plaintexts. However, Textbook-RSA is not considered secure [BJN00]. In 2009, Craig Gentry discovered a *fully homomorphic encryption* that supports multiplication as well as addition [Gen09], and theoretically solves our privacy problem for scenarios involvin only one client: The client could simply use the proposed encryption scheme, and the service provider could adapt its service to work on encrypted data using this scheme. However, this is not feasible since the size of the key scales with the size of the circuit of the algorithm which the service calculates. It is more efficient to execute the service on the client than to use homomorphic encryption.

Secure Multiparty Computations [GMW87, CCD88] are cryptographic solutions for problems where two or more parties cooperatively want to compute a certain function over a set of data without any party learning anything about the input of other parties except what is learned by the output. The problem is that for each party, the computation cost is higher than computing the whole function on the complete input without any other party. This makes the concept of multiparty computation for outsourcing services too expensive and in fact pointless if the client is the only one with private input.

There are methods to retrieve information from an outsourced database without the server learning anything about the information queried [Gas04]. However, these methods are also infeasible in most cases: If the database server must not learn anything about a query, the query issued to the database must contain every cell. Otherwise the server learns which cells do not contribute to the result of the query, and thus learns something about the result set, if no special-purpose hardware is involved [KC04]. The need to iterate over every cell for every query execution makes private information retrieval impractical. If for every

query, the database service has to touch every cell, the service does not scale.

For many scenarios pure software solutions do not provide enough protection. It is believed, that hardware can be made more tamper resistant than software. Therefore many solutions to security problems involving secrets incorporate dedicated hardware [DL07, BS03]. Assuming the hardware can be trusted, there are solutions to security problems, that are otherwise infeasible or impossible. In the context of secure service outsourcing, the *Trusted Computing* approach [TCG] is very interesting. In theory a lightweight tamper resistant piece of hardware the so-called *trusted computing module* can be used for ensuring that the software running on the machine is certified. This, in principle, can solve many problems such as software protection and digital rights management. This, however, does not solve the problem of building a service that provides privacy for the clients data, but the problem of verifying that the desired software is actually running on the server.

3.3 Approaches for Secure Database Outsourcing

Since many services rely on databases, the problem of secure database outsourcing emerged early in 2002. Classical cryptographic notions are not applicable to encrypted databases under practical constraints because in general it is infeasible to realize a database that complies with these notions [KC04]. However, there are approaches that try to provide some level of privacy. We believe that the methods developed in this context are the most advanced ones that try to solve the privacy problem of outsourcing. Most of them combine cryptography with special architectures in order to achieve a higher level of privacy. They implicitly introduce an adapter deployed on the client, that encapsulates en- and decryption as well as distribution of data according to the proposed approach. From an architectural point of view, you can distinguish between two classes of approaches, which we term *coarse indices approach* and *distribution approach*.

The idea of the coarse indices approach [HILM02, DVJ+03, CDV+05, HMT04, BBO07] is to encrypt the database on a tuple level. In order to support efficient query processing indices are created. These indices are coarse-grained. They contain keywords, ranges, or other identifiers and encrypted row ids where the keywords occur in the database. In most approaches, these indices are kept in the adapter. An adapter deployed on the client handles query transformation and sorts out false positives. Figure 1 depicts the architecture of the coarse indices approach. When issued a query, the adapter first queries the coarse-index tables and decrypts the results. Then, it queries the encrypted database with the results from the index tables and recieves encrypted tuples. These tuples potentially contain false positives due to the coarse indices. The adapter decrypts these rows, sorts out the false positives, and retuns an exact anwser to the clients query. Since the adapter handels (en- and) decryption and query transformation, this is transparent to the client application. This approach supports efficient execution of exact match and range queries. It is unclear, however, what level of privacy this approach provide for realistic scenarios. Since we need solutions that provide provable security (c.f. Section 2), these approaches for itself are insufficient.

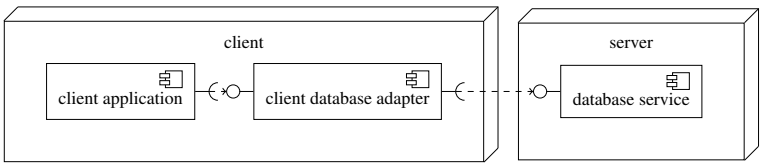

Figure 1: The architecture of a secure outsourced database proposed for example in [HILM02, DVJ+03]. The adapter provides and requires a standard SQL interface. The overall database service consists of the database service on the server and the adapter on the client.

The idea of the *distribution approach* [ABG+, CdVF+] is to distribute the database to several servers in order to fulfill previously defined *privacy constraints*. When these contraints cannot be fulfilled, encryption can be used. A database is called secure, of all privacy constraints for this individual database are met. Consequently this approach does not provide a general but rather a individual security guarantee. The classes of queries this approach supports to execute efficiently depend on the chosen distribution scheme and encryption methods used. In the worst case, it is either not possible to execute any class of queries efficiently, or the approach does not provide provable security. Therefore this approach is not a solution that fulfills out requirements (c.f. Section 2).

4 Anonymity Properties and Notions for Databases

In the previous section, we have seen different approaches that try to solve the secure database outsourcing problem. They, however, either do not provide security guarantees, or have strong assumptions about the input data or hardware, and are therefore not applicable in most cases. In order to provably provide a service that provides privacy for its clients data, we need a formalization of the level of privacy provided. In this section, we will examine different anonymity and security notions for databases with respect to their security and applicability to service outsourcing scenarios.

The anonymity properties k-*anonymity*, l-*diversity*, t-*closeness* and *differential privacy* are motivated from privacy preserving database disclosure. In this scenario, a trusted instance wants to disclose a database that contains data about people whose privacy must be protected. Therefore the database has to be transformed into a database fulfilling a particular anonymity property (offline). In the case of differential privacy, the results of queries are altered (online). Figure 2 depicts the architecture of the offline scenario of privacy preserving database disclosure, where an anonymizer transforms a database d into d'. Although, these properties are originated from privacy preserving database disclosure outsourcing, they might be applicable to secure database outsourcing.

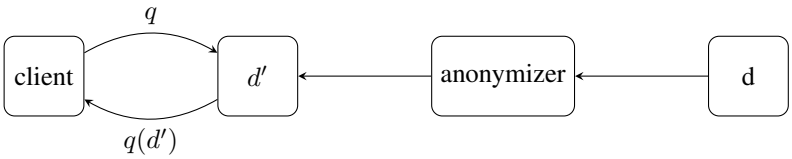

Figure 2: Offline scenario for privacy preserving database disclosure. An anonymizer transforms a database d, and discloses the result d'. Clients then can query d'

4.1 k-anonymity, l-diversity, and t-closeness

In 2002, Latanya Sweeney [Swe02] defined the property k-anonymity. The intention of k-anonymity is that any individual in the database can only be mapped to at least k rows. Therefore attributes of datasets are classified into *identifiers*, so-called *quasi-identifiers*, and *sensitive information*. A anonymized table must not contain attributes classified as identifiers. The property k-anonymity is defined as follows:

Definition 1 *k-anonymity*
A database adheres k-anonymity if for each row, there are at least $k-1$ other rows with the same attribute values of the quasi-identifiers. Rows with the same attribute values of their quasi-identifiers are called a equivalence class.

In order to achieve k-anonymity, attribute values can be generalized and rows can be suppressed.

Different weaknesses of and attacks on k-anonymity have been identified [MKGV07]. So-called *Homogeneity Attacks* as well as *Background Knowledge Attack* can be used to break k-anonymity. If an attacker knows that a particular individual is in the disclosed database and all sensitive information cells of the k corresponding rows have the same value (alternatively: the attacker can rule out particular values by background knowledge), she can infer information that intuitively should be protected by the property of k-anonymity. Consider for example the table in Figure 3. This database adheres 2-anonymity. If an attacker knows an individual that is 30 years old and lives in area code 3524 and knows that he is in the database, she can infer that the individual has the flu.

Because these attacks the property of l-diversity [MKGV07] has been defined. l-diversity constrains the attribute values of the sensitive information column in an equivalence class:

Definition 2 *l-diversity*
A database adheres l-diversity if in each equivalence class, there are at least l "well-represented" values in the sensitive information column.

There are different interpretations of "well-represented" [MKGV07]. Although l-diversity has been defined in order to eliminate homogeneity and background knowledge attacks, they are still possible. For example if the used interpretation of "well-represented" does not consider semantic similarities as for example "gastritis" and "stomach pain" (c.f. Figure 3).

quasi-identifier		*sensitive information*
age	area code	disease
15-34	352*	flu
15-34	352*	flu
35-40	345*	gastritis
35-40	345*	stomach pain

Figure 3: A medical database adhering 2-anonymity the attributes age and area code have been generalized in order to achieve 2-anonymity.

Because of the weaknesses of l-diversity, t-closeness has been defined [LL07]:

Definition 3 *t-closeness*
An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

t-closeness shifts the problem of defining a resilient anonymity property to the problem of finding a resilient distance measurement for distributions. Moreover, if t is small, the table contains not more information than the distributions of attribute values, which is in contrast to the initial intention of disclosing whole databases instead of just summaries.

4.2 General Problem of k-anonymity, l-diversity, and t-closeness

Besides the weaknesses and attacks described in Section 4.1, k-anonymity, l-diversity, and t-closeness have another fundamental weakness.They describe the result of an anonymization process, not the process itself. An attacker can, knowing the anonymization process, learn information from the disclosed database that should intuitively not be learned from a database adhering k-anonymity, l-diversity, or t-closeness. In the remainder of this section, we will provide two examples. Consider for example the database depicted in Figure 4. In general, a database has more than one versions adhering k-anonymity. Figure 5, for

name	age	area code	disease
Alice	31	3524	flu
Bob	31	3456	flu
Carol	35	3524	gastritis
X	35	3456	stomach pain

Figure 4: A medical database

example, depics two versions of the database in Figure 4 adhering 2-anonymity.

age	area code	disease
3*	3524	flu
3*	3456	flu
3*	3524	gastritis
3*	3456	stomach pain

(a)

age	area code	disease
31	3*	flu
31	3*	flu
35	3*	gastritis
35	3*	stomach pain

(b)

Figure 5: Databases adhering 2-anonymity derived from the table in Figure 4

Consider an anonymization process as follows. If "X" = "Eve", disclose Table 5(a), otherwise disclose Table 5(b). Now, although the results adhere 2-anonymity, an attacker learns if Eve is in the database just by knowing the anonymization process and looking at which attribute was generalized.

Consider as another example a database about a city comprised of three city parts A, B, and C with 7, 9 and 20 inhabitants, respectively. The database consists of a quasi-identifier the city part, and a binary sensitive information attribute p. Let the anonymization process merge city parts as long as the database does not adhere 2-diversity and try to minimize the number of merges. If all city parts are merged in the disclosed database, an attacker can infer information based on the number of people with p according to the table in Figure 6. Consider for example 8 people with p living in the city. If anybody in city part A

# people with property p	information attacker can infer
1	(*nothing to infer*)
2 - 7	all inhabitants with p live in the same city part
8,9	nobody living in part A has p
10 - 15	all inhabitants with p live in part C
16 - 20	all inhabitants with p live in part C or all inhabitants of A and B have p
21	(*impossible*)

Figure 6: Information an attacker can infer depending on the number of people with attribute p in the disclosed database adhering 2-diversity.

had p and without loss of generality nobody in C had p the anonymization process would have stopped after merging A and C. In the other case (people in B and C have p) the anonymization process would have stopped immediately.

These examples show, that, knowing an anonymization process, an attacker can infer information that should be protected by l-diversity.

4.3 differential privacy

In contrast to k-anonymity, l-diversity, and t-closeness that describe the result of the anonymization process, *differential privacy* [Dwo06] describes the anonymization process itself. This fixes the problem described in the previous section. The intention is to minimize the risk of an individual joining a database. Formally, differential privacy is defined as follows [Dwo08]:

Definition 4 *differential privacy*
A randomized function $\mathcal{K}$ gives ε-differential privacy if for all data sets D_1 and D_2 differing on at most one element and all $S \subseteq Range(\mathcal{K})$:

$$Pr[\mathcal{K}(D_1) \in S] \leq exp(\varepsilon) \times Pr[\mathcal{K}(D_2) \in S]$$

Informal, this means that if a function adheres diffential privacy, the probability of getting different results from data sets differing on at most one row should be less than a certain threshold.

In practice this notion is achieved by adding noise to results. Therefore a trusted instance called "curator" is placed between clients and the database (cf. Figure 7). The curator forward the clients' queries and adds noise according to assumptions about the universe of all possible databases.

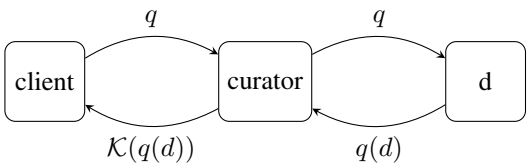

Figure 7: An architecture for differential privacy. A trusted curator is placed between the clients and the database d and transforms query results in order to achieve differential privacy

This is also the biggest drawback of this notion. In order to achieve differential privacy by adding noise, you have to know every entity that possibly can be added to your database.

4.4 k-Ind-IPC

Another notion, that describes the anonymization process instead of the result is k-Ind-IPC [HHK+10]. Intuitively, k-Ind-IPC hides relations within an equivalence class:

Definition 5 *k-Ind-IPC*
Let d and d' be databases, where d' can be gain by partitioning d into parts of at least k-rows, permuting the attribute values indepentdently within each column and each part and

uniting the resulting. A anonymization function f provides k-Ind-IPC if for all databases d and d'. $f(d)$ is indistinguishable from $f(d')$

For a more formal definition please refer to [HHK+10].

Since this notion is motivated from secure database outsourcing, there are methods for outsourcing databases that provide k-Ind-IPC [Hub10]. These methods combine the approaches presented in Section 3.3 and involve separating the database service, deploying the parts on different servers and deploying an adapter on the client. Only a limited set of queries, however, can be executed efficiently and, due to the separations of duties approach, different servers are needed.

5 Conclusion

In this paper, we presented methods that potentially can solve the privacy problems inherent to service outsourcing. We evaluated these methods with respect to performance and provable security. Then, we presented approaches for secure database outsourcing, that apply these methods. We have seen that thees approaches imply special architectures. Most these concrete approaches do not provide a practical security guarantee. Therefore we examined potential candidates for formalizations of security notions for outsourced databases. Differential privacy and k-Ind-ICP seem viable candidates. Since differential privacy, however, is originated from a different scenario, it is unclear how this notion can be applied in a secure database outsourcing scenario.

For future work we want to examine how to build databases that provide k-Ind-ICP or differential privacy, and what architectures these notions imply. We also want to examine if and how differential privacy and k-Ind-ICP relate. This can lead to a better understanding of these notions and also to new methods that support them.

References

[ABG+] Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas, and Ying Xu. Two Can Keep a Secret: A Distributed Architecture for Secure Database Services. *CIDR 2005*.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and Efficiently Searchable Encryption. In *CRYPTO*, pages 535–552, 2007.

[BJN00] Dan Boneh, Antoine Joux, and Phong Nguyen. Why Textbook ElGamal and RSA Encryption are Insecure (Extended Abstract), 2000.

[Bla93] Matt Blaze. A cryptographic file system for UNIX. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 9–16, New York, NY, USA, 1993. ACM.

[BS03] Adrian Baldwin and Simon Shiu. Hardware Encapsulation of Security Services. In Einar Snekkenes and Dieter Gollmann, editors, *Computer Security â ESORICS 2003*, volume 2808 of *Lecture Notes in Computer Science*, pages 201–216. Springer Berlin / Heidelberg, 2003.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19, New York, NY, USA, 1988. ACM.

[CDV+05] Alberto Ceselli, Ernesto Damiani, Sabrina De Capitani Di Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Modeling and assessing inference exposure in encrypted databases, 2005.

[CdVF+] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, SUSHIL JAJODIA, Stefano Paraboschi, and PIERANGELA SAMARATI. Combining Fragmentation and Encryption to Protect Privacy in Data Storage.

[DL07] Jeffrey S. Dwoskin and Ruby B. Lee. Hardware-rooted trust for secure key management and transient trust. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 389–400, New York, NY, USA, 2007. ACM.

[DVJ+03] Ernesto Damiani, S. De Capitani Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs, 2003.

[Dwo06] Cynthia Dwork. Differential Privacy. *Automata, Languages and Programming*, pages 1–12, 2006.

[Dwo08] Cynthia Dwork. Differential Privacy: A Survey of Results. *Theory and Applications of Models of Computation*, pages 1–19, 2008.

[Gas04] William Gasarch. A Survey on Private Information Retrieval. *Bulletin of the EATCS*, 82:72–107, 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA, 2009. ACM.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM.

[HHK+09] Christian Henrich, Matthias Huber, Carmen Kempka, Jörn Müller-Quade, and Mario Strefler. Towards Secure Cloud Computing. In *Proceedings of the 11th International Symposium on Stabilisation, Safety, and Security of Distributed Systems (SSS 2009)*, 2009.

[HHK+10] Christian Henrich, Matthias Huber, Carmen Kempka, Jeorn Mueller-Quade, and Ralf Reussner. Technical Report: Secure Cloud Computing through a Separation of Duties. `https://sdqweb.ipd.kit.edu/huber/reports/sod/technical_report_sod.pdf`, 2010.

[HILM02] Hakan Hacigümüs, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227. ACM, 2002.

[HMT04] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 720–731. VLDB Endowment, 2004.

[Hub10] Matthias Huber. Towards Secure Services in an Untrusted Environment. In Barbora Bühnová, Ralf H. Reussner, Clemens Szyperski, and Wolfgang Weck, editors, *Proceedings of the Fifteenth International Workshop on Component-Oriented Programming (WCOP) 2010*, volume 2010-14 of *Interne Berichte*, pages 39–46, Karlsruhe, Germany, June 2010. Karlsruhe Institue of Technology, Faculty of Informatics.

[KC04] Murat Kantarcioglu and Chris Clifton. Security Issues in Querying Encrypted Data. Technical report, 2004.

[LL07] Ninghui Li and Tiancheng Li. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. 2007.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy beyound k-Anonymity. *Cornell University*, page 52, March 2007.

[NIS09] NIST. NIST - Cloud Computing. `http://csrc.nist.gov/groups/SNS/cloud-computing/`, 2009.

[Pea09] Sinai Pearson. Taking Account of Privacy when Designing Cloud Computing Services. *HP Laboratories*, 2009.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Swe02] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[TCG] TCG. Trusted Compting Group. `http://www.trustedcomputinggroup.org/`.