A Database Adapter for Secure Outsourcing to the Cloud

Rafael Dowsley, Matthias Gabel, Kateryna Yurchenko and Valentin Zipf

Karlsruhe Institute of Technology
Emails: {rafael.dowsley,matthias.gabel,kateryna.yurchenko}@kit.edu

***Abstract*—The advent of cloud computing and storage provides numerous opportunities for better management of resources, with the potential of drastically reducing costs. However, when data is outsourced to the cloud, new security vulnerabilities emerge, as the cloud provider (and its employees) are normally not completely trusted by the party that is outsourcing the data. Therefore additional security mechanisms are needed in order to prevent against internal attacks in the cloud provider. Nonetheless, the performance and functionality should be impacted as less as possible. This work presents a database adapter for the secure outsourcing of data that aims at achieving a good performance-security trade-off.**

***Index Terms*—Database Adapter, Secure Outsourcing, Side-Channel Attacks, Data Distribution.**

I. Introduction

Cloud computing and cloud storage provide a myriad of possibilities for companies to promptly adjust their computing resources to meet their ever-changing requirements in a cost-effective way. Accordingly, there is a rapidly growing interest in outsourcing databases to the cloud. However, when a database is outsourced to a cloud, new security issues appear as the cloud provider (and its administrators) are not always completely trusted by the database owner. Therefore there is an inherent need to protect the database against internal adversaries (e.g., cloud administrators). This work deals with the problem of enhancing the security of outsourced databases while preserving their functionality. On one hand, the security level should be increased. On the other hand, the functionality and performance of the database should not suffer a big impact. Given the current state of affairs in the field of cryptography, this requires us to choose a good trade-off between performance and security. This work describes the approach adopted in EU research project PaaSword[1].

From a theoretical point of view, there are very powerful cryptographic solutions for this problem which can achieve very strong security levels while preserving the functionality. Fully homomorphic encryption (FHE [1]) is one example of a cryptography primitive that could be useful in such context since it provides the ability to perform operations over the encrypted data. However, from a practical point of view, it is an unacceptable solution as its current performance is by far insufficient for practical use. Another possible approach would be to encrypt all the data before sending it to the storage server, and then downloading all the data for local decryption and processing whenever some operation is needed. But it is obvious that this approach scales very badly with large data sets and therefore cannot be used in real world deployments. This work is focused on a more practical approach that aims at a reasonable trade-off between security and efficiency. Cryptographic primitives such as deterministic encryption can be used to obtain a better performance-security balance. Another possible direction for improving security without a significant degradation in performance is the usage of algorithms for data distribution among servers. Ideally, each type of data would be distributed to a different server in order to decrease the impact of internal attacks. However, normally, there are a limited number of servers and this method does not scale. Therefore more than one type of data need to be stored in each server and so data distribution schemes that respect privacy constraints should be used instead.

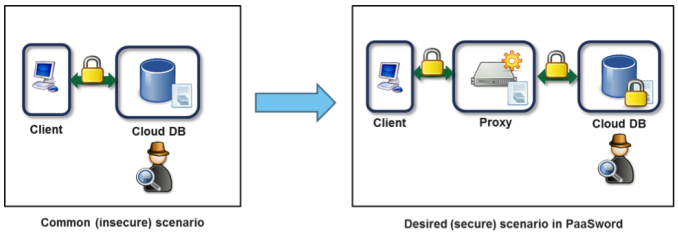

Fig. 1. A Database Adapter for Secure Outsourcing.

From a security viewpoint, the idea used in PaaSword is to introduce an entity to automatically distribute and encrypt data before it is uploaded into the cloud as depicted in Figure 1. We call this entity the DB proxy or just proxy. It acts as an adapter to give an application transparent access to a secure and distributed database. We focus on internal passive adversaries (e.g., cloud administrators) as we assume that cloud service providers offer high level of security against external adversaries by employing standard cryptographic mechanisms to mitigate this threat.

A. Database Adapter for Secure Outsourcing

The idea of using a database adapter in order to provide security guarantees was previously used in the MimoSecco project [2]. The PaaSword database adapter is an enhanced version of the MimoSecco one that deals with many of the

[1]https://www.paasword.eu

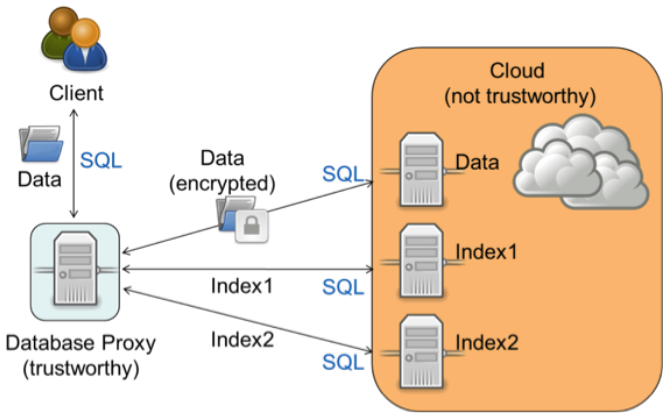

Fig. 2. Operation of the MimoSecco database adapter.

TABLE I
ORIGINAL DATA BEFORE THE TRANSFORMATION.

Row	Name	Surname	City
1	Jacob	Anderson	New York
2	Sophia	Anderson	Chicago
3	Emma	Connor	Los Angeles

previous security shortcuts and improves the performance and functionality. In order to better contextualize the improvements we present here a short overview of the MimoSecco database adapter.

MimoSecco [2] is a cloud-storage technique for securing relational databases that considered a model with three different security zones: the zone in which the client is located is assumed to be completely trusted. The zone in which the cloud storage is located is untrusted and the cloud provider is considered to be an honest-but-curious adversary; i.e., it follows the protocol instructions correctly, but tries to learn additional information. This captures for instance attacks launched by the cloud administrator. In the middle there is a semi-trusted zone, in which there is an adapter that provides access to the cloud storage.

The SQL queries made by the client to the database proxy are converted to SQL queries that the proxy forwards to the cloud storage, see Figure 2. The main goal of the MimoSecco adapter is to hide the relation between the attributes, rather than the attributes themselves. This is done by creating two different classes of tables: encrypted data tables (which store the data encrypted) and index tables (which are used to perform the SQL queries). The behavior of the adapter as well as an example of the table's division is presented below: Table I presents the original data before the transformation is applied; Table II depicts the index tables after the transformation and Table III shows the encrypted data table after the transformation. The resulting data table contains the same data, but encrypted row-wise using a randomized encryption scheme (all the attributes are concatenated into a single ciphertext). The index tables contain encrypted links to positions in the data table and are responsible for allowing fast look-up.

The MimoSecco scheme achieves the security notion called

TABLE II
INDEX TABLES AFTER THE TRANSFORMATION.

Keyword	Rowlist
Emma	Enc(3)
Jacob	Enc(1)
Sophia	Enc(2)

Keyword	Rowlist
Anderson	Enc(1‖2)
Connor	Enc(3)

TABLE III
ENCRYPTED DATA TABLE AFTER THE TRANSFORMATION.

Row	Encrypted Data
1	Enc(Jacob‖Anderson‖New York)
2	Enc(Sophia‖Anderson‖Chicago)
3	Enc(Emma‖Connor‖Los Angeles)

IND-ICP [3], [4] that guarantees that the relation among the attributes is not leaked. However, there is data that is sensitive even when considered out of context, such as credit card numbers for instance. Such data needs to be protected by not being stored in clear text. Additionally, the MimoSecco architecture also have challenges dealing with side-channel attacks, which exploit aspects of the real deployment that are not modeled by the security model (examples of such attacks will be described in Section III). The PaaSword database adapter presents security improvements that deal with the above mentioned problems. To tackle this problem we introduced countermeasures based on the encryption of the index tables as well as on the distribution of the tables among many servers.

II. Related Work

There are several approaches to securely outsource data to potentially untrusted environments such as external servers or public clouds. The requirements for such solutions are not only protection against loss of sensitive data but also the ability to perform searches and other kinds of computations on the outsourced data in a efficient manner.

For example, homomorphic encryption, which was introduced by Rivest et al. [5], allows to do mathematical calculations on ciphertexts such as either addition [6] or multiplication [7], [8], [9]. A fully homomorphic encryption scheme (i.e., a scheme allowing both addition and multiplication, and thus allowing the computation of any algebraic circuit) was first introduced by Gentry [1]. Despite being a very powerful primitive and representing one of the hottest research topics in cryptography recently, the state-of-the-art fully homomorphic encryption schemes are not practical enough for real world deployment.

Order-preserving encryption (OPE) schemes have the goal of allowing to efficiently compare the order based only on the corresponding ciphertexts. Agrawal et al. [10] proposed such a scheme in which numerical plaintexts are mapped to randomly selected values (the ciphertexts) from a chosen statistical distribution. Obviously this approach can only be secure if the domain of the distribution is big enough. Therefore the suggestion [11], [12] is to use hyper-geometrical distributions with the "Lazy Sampling" technique which allows to find

ciphertexts more securely and comfortably. A different hybrid approach to realize order-preserving encryption was proposed by Popa et al. [13]. Here (non order-preserving) ciphertexts are organized in a binary tree so that the order of the nodes fits the order of the plaintexts.

Another special type of encryption scheme that has the goal of allowing a particular type of operation over the ciphertexts is searchable encryption (SE). Here the goal is allowing to efficiently search keywords in encrypted documents. Song et al. [14] proposed the first encryption scheme, with a two-fold scheme in which a ciphertext contains, next to the encryption of the plaintext, a specially generated hash value. If a search on outsourced ciphertexts is made, the client can generate a trapdoor for a keyword. Using both the hash value of the ciphertexts and the trapdoor of the client, the server is able to find those ciphertexts. A lot of works have further developed the field of searchable encryption schemes, e.g. [15], [16]. One of the main applications of searchable encryption is in cloud solutions, see e.g. [17], [18]. Cash et al. [19] showed how to extend the data structures of SSE schemes that allow single-keyword searches in order to permit more expressive queries such as conjunctive search and general Boolean queries. The research in the area of searchable encryption schemes is still very active nowadays. For a recent survey about searchable encryption and its relevance for cloud computing, see [20].

Beside the cryptographic schemes, there are other hybrid approaches that combine the use of normal encryption schemes and techniques for data management. One approach published by Hacigümüs et al. [21] suggested to encrypt database entries row-wise and use an adapter to translate queries to the database as well as to take care of encryption and decryption. Hore et al. [22] proceeded the work of Hacigümüs et al. [21] and suggested performance improvements by indexing the encrypted database entries. Another interesting approach is the onion encryption that was made public by the CryptDB project of Popa et al. [23]. In this approach a mixture of common, SE, OPE and homomorphic encryption schemes are applied to a plaintext in a specific order so that a plaintext is wrapped in multiple encryption layers. If, for example, a range query is executed and the ciphertexts have to be compared by their order, all encryption layers of the ciphertexts are removed (decrypted) until the layer of OPE is revealed and the query can be executed on the ciphertexts. The downside of this approach is that the security guarantees for the user are unclear because the goal is to give the server the least amount of knowledge needed to process the queries. As a result a single query might lead to a permanent decryption of onions hence sensitive information cannot be encrypted again.

There are also approaches that do not implement security by encryption but by the distribution of data. Aggarwal et al. [24] suggested so-called privacy constraints to separate columns of databases that together have a high information leakage to an attacker. They suggested separating and distributing those columns that should not be stored together to different non-communicating servers so that the possibility for an attacker to learn sensitive data is decreased. Ciriani et al. [25] afterwards published an algorithm to detect and apply such privacy constraints on databases.

The PaaSword project itself is a hybrid and index-based approach that is a derivative of prior research projects such as Cumulus4j [3] and MimoSecco [2]. In PaaSword the encryption key is not stored locally on the database adapter; instead a distributed key management mechanism is used (see [26] for more details).

III. Side-channel Attacks

As already mentioned in the introduction, the MimoSecco adapter (the predecessor of the PaaSword adapter) achieves the IND-ICP security notion [3], [4], which guarantees that the relation among the attributes is not leaked. However, in real world deployments there are normally side-channels attacks that are not covered by the security model. And, in fact, Huber and Hartung [4] showed some side-channel attacks against the MimoSecco adapter. We should emphasize that vulnerabilities to side-channel attacks is not a specific problem of the MimoSecco scheme, it is a general problem.

The following examples illustrate the concept of side-channel attacks:

- Background knowledge: An adversary with background knowledge, for instance that men cannot be pregnant, can from the permuted database already exclude some possibilities for the original database. This attack is not captured by the IND-ICP security notion.
- Access statistics: By observing the time and order of the accesses to the index tables, the adversary can learn valuable information as correlated data is normally accessed in short time intervals.
- Database updates: Upon updates of the database, an adversary can learn some relation among the attributes by observing the updates in the index tables. This attack is also not modeled in the IND-ICP security notion.
- Order of physical storage: The order of the physical storage can be different from that of the model, and this can help the adversary. The physical order can possibly reflect the inclusion order.

A. Counter-measures

Some of the counter-measures that can be used to mitigate side-channel attacks include:

- Deterministic encryption of the indexes: By encrypting the keywords of the indexes tables using a deterministic encryption scheme it is possible to reduce the effectiveness of side-channels attacks as the cloud does not get plaintext anymore. And, since the encryption is deterministic, it is still possible to perform exact match queries efficiently. The PaaSword database adapter enables the use of deterministic encryption of the indexes. More information is presented in Section IV. Searchable encryption schemes can even provide a higher level of security, at the cost lower performance (see [20] for a recent survey of that area).

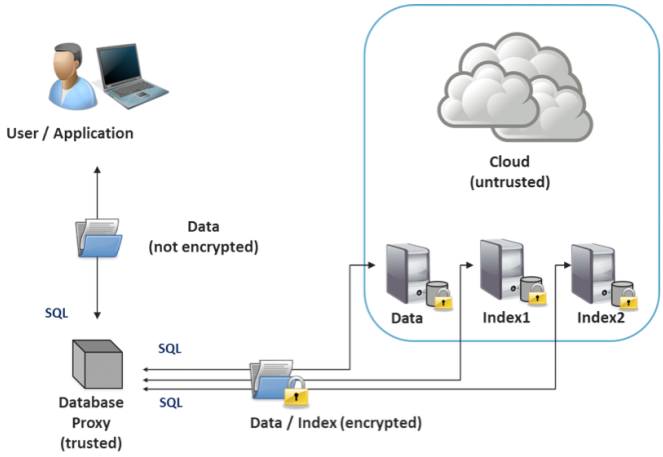

Fig. 3. The PaaSword Database Adapter.

- Order-preserving encryption: Similarly, by using order-preserving encryption, it is possible to reduce the effectiveness of side-channels attacks while keeping efficient range queries.
- Delayed updates: By caching and permuting the updates in the database adapter and performing many updates simultaneously, the effectiveness of the side-channels based on database updates can be reduced.
- Distributed storage: By storing the different index tables in different servers that do not communicate with each other it is possible to drastically reduce the effectiveness of side-channels attacks. Ideally, each index table would be stored in a different server, but in practice there are limitations on the number of non-communicating servers that are available. Therefore one needs to adopt solutions based on data distribution that complies with privacy constraints established by the database owner. The PaaSword adapter enables the distributed storage and also implements an algorithm to, given the privacy constrains, determine the way that the data should be distributed among the available servers.

IV. Encrypted Index

As already mentioned in the introduction, the PaaSword database adapter follows the same design paradigm as MimoSecco [2]. However, one of the countermeasures against side-channel attacks of the PaaSword adapter is enabling the deterministic encryption of the keywords in the index tables (in addition to randomized encryption of the links in those tables). See Figure 3 for the depiction of the PaaSword database adapter. For the same original data as presented before in Table I, the index tables in PaaSword would look like Table IV (compare that with MimoSecco index tables in Table II). The names (Emma, Jacob, Sophia) and surnames (Anderson and Connor) are encrypted using a deterministic scheme.

Keywords stored in clear in the index tables can help an adversary to gain some information about the original database if he has background knowledge, or if he has the opportunity

TABLE IV
INDEX TABLES IN PAASWORD.

Keyword	Rowlist
0x0B34 = DetEnc(Sophia)	Enc(2)
0x78AB = DetEnc(Emma)	Enc(3)
0xC134 = DetEnc(Jacob)	Enc(1)

Keyword	Rowlist
0x5AFE = DetEnc(Anderson)	Enc(1‖2)
0x7732 = DetEnc(Connor)	Enc(3)

to observe database updates, access statistics or physical order of data entries on the storage. Deterministic encryption of keyword values in the index tables is a good approach that can be applied in order to come down to a trade-off between security and providing efficient support for as many different query types as possible. If the keywords values are encrypted, it is more difficult for the adversary to perform side-channel attacks. At the same time, since the encryption is deterministic, a user can still execute exact-match queries.

For example, when the database adapter gets a query for the name Emma, it encrypts Emma with the deterministic encryption scheme in order to obtain 0x78AB and then search for this value in the index table. As a result he gets Enc(3), the encrypted link (using a randomized encryption scheme) to the row corresponding to Emma. The adapter then decrypts the value to 3 and requests the third row from the data table. Upon receiving the (randomized) encryption Enc(Emma‖Connor‖Los Angeles), the adapter can decrypt it to get Emma's attributes. Negation and IN queries can be supported in a similar way using standard set operations.

V. Preserving Functionality when using an Encrypted Index

The deterministic encryption of the keywords in the index tables comes with some disadvantages regarding the query support and the performance of the database adapter. Foremost it is to be said that the database adapter is capable of handling all types of queries. But with encrypted keywords for some queries there is a significant loss in performance. Firstly there is a small overhead in analyzing and rewriting the query because the plaintext values in it have to be encrypted before executing the query on the tables. As already mentioned exact-match queries come with this minor performance loss because matching plaintexts is equal to matching ciphertexts if they are encrypted with a deterministic scheme. But other kinds of queries face a much larger disadvantage. For range queries that use $<, \leq, \geq, >$ or the BETWEEN operator in WHERE statements, *all* keyword values related to the queried column have to be decrypted before the actual query is executed. This is because the ciphertexts do not preserve order. Also LIKE statements that use the wildcard operator % are very difficult to handle because there is no way to tell for a ciphertext which plaintext characters are mapped to characters of the ciphertext. Here the solution is also the decryption of all relevant keywords. For aggregates like SUM or AVG

the encryption of keywords has a minor effect since these are calculated based on the entries of the data table. These entries are encrypted by a randomized encryption scheme and thus have to be completely decrypted if an aggregate function should be applied to them. The COUNT function is also not affected since it just reflects the number of returned rows of the data table. If a query that contains aggregates uses a WHERE statement, all above mentioned disadvantages apply to it. The same applies for INSERT, UPDATE and DELETE operations.

For a sufficiently small database this overhead of possibly decrypting all keywords tends to be negligible. But outsourced databases tend to be huge and grow over time and thus the amount of decryption cycles will rise as well. This might lead to a bad performance in big-data scenarios if many non-exact match queries are issued. A solution to this problem might be to distribute the keywords among multiple non-communicating servers (see Aggarwal et al. [24]), so that each server only contains the keywords for a subset of columns of the database tables. The keywords then can be left unencrypted if privacy constraints are applied so that a possible attacker does not learn relevant and related data. With plaintext keywords the actual search can be done like in a normal database but in a distributed fashion. Now there is an overhead in building up secure connections (TLS) to transfer the identifiers from the servers, but this overhead does not directly correlate with the amount of keywords. This scenario would allow even more performance and security improvements because queries can be split up and processed in parallel and thus possible attackers (on the servers) only learn a part of the entire query.

VI. Implementation and Benchmarks

The implementation uses Java as the programming language. For the database connection to PostgreSQL we use the corresponding JDBC driver. The Advanced Encryption Standard (AES) is the encryption algorithm and it is currently used in the Cipher Block Chaining (CBC) mode of operation. For future work employing Galois Counter Mode (GCM) is an option. The PaaSword database adapter supports a wider range of SQL query types than its predecessors.

Since deterministic encryption of keywords does not allow for efficient execution of all query types, we compare the performance of the improved middleware with the original prototype without deterministic encryption of the keywords in the index tables. We applied performance analysis methods to the improved middleware and compared the results with the original prototype. Figure 4 illustrates the execution time of four different query types on the original middleware in comparison to the middleware with encrypted indexes.

The comparison showed that our security improvement did not lead to a big overhead. In the cases where the queries can be executed efficiently (exact-match queries), our version of the prototype computes results almost as fast as the original version. In all of these cases, the execution time was maximum of 12% longer. For the cases where queries could not be effectively executed, our system with encrypted indices is maximum of 40% slower, which is also a good result when considering that the privacy of the attribute values associations is preserved and an adversary is not able to learn any sensitive data.

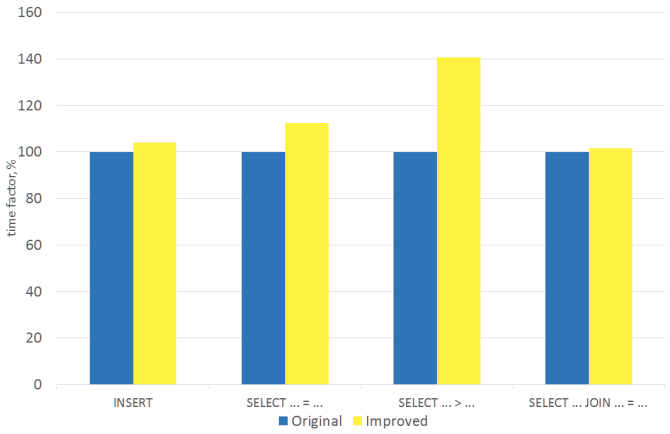

Fig. 4. Performance comparison of different query types on both the original and the improved middleware.

VII. Conclusion and Future Directions

In this paper we introduced PaaSword data outsourcing scheme that encrypts sensitive information before uploading it to untrusted cloud providers. Encryption, decryption, query analysis, processing and execution is handled by a trusted adapter that we call the database proxy. It offers secure and transparent access to a relational cloud database and uses a reverse index to increase performance. We pointed out some security issues, mainly due to side channels like access patterns and background knowledge. These attacks can be mitigated by deterministic index encryption and data distribution to non-communicating servers. We showed how different query classes can still be executed efficiently in most cases when working with the more secure encrypted index. To prove that our improved approach is working, we implemented the adapter with index encryption and successfully validated the results. Depending on the query type the overhead of index encryption is negligible or at most 40% while a significant gain privacy is achieved.

Future work will address a more efficient handling of specific queries. Range queries ($a < x < b$) can be speeded up by using order-preserving encryption schemes thus enabling for sub-linear search time. For wildcard queries (name LIKE 'Emil%'), there exist sophisticated and complex approaches that can be combined with our scheme. Another direction is to support random addition, where a sensitive value v is split into two values a and b such that $v = a + b$. Then, a and b are distributed to different non-communicating servers such that the adapter can still perform some operations like average and mean efficiently by combining the computation results of the two servers.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation

programme under grant agreement No 644814, the PaaSword project (www.paasword.eu) within the ICT Programme ICT-07-2014: Advanced Cloud Infrastructures and Services.

REFERENCES

[1] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, http://crypto.stanford.edu/craig.

[2] M. Gabel and G. Hübsch, "Secure Database Outsourcing to the Cloud Using the MimoSecco Middleware," in *Trusted Cloud Computing*, H. Krcmar, R. Reussner, and B. Rumpe, Eds. Cham: Springer International Publishing, 2014, pp. 187–202.

[3] M. Huber, M. Gabel, M. Schulze, and A. Bieber, "Cumulus4j: A provably secure database abstraction layer," in *International Conference on Availability, Reliability, and Security*. Springer, 2013, pp. 180–193.

[4] M. Huber and G. Hartung, "Side Channels in Secure Database Outsourcing on the Example of the MimoSecco Scheme," in *Trusted Cloud Computing*, H. Krcmar, R. Reussner, and B. Rumpe, Eds. Cham: Springer International Publishing, 2014, pp. 35–48.

[5] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[6] P. Paillier and D. Pointcheval, "Efficient public-key cryptosystems provably secure against active adversaries," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 1999, pp. 165–179.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[8] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '82. New York, NY, USA: ACM, 1982, pp. 365–377.

[9] E. Taher, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 10–18.

[10] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 2004, pp. 563–574.

[11] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 224–241.

[12] A. Boldyreva, N. Chenette, and A. ONeill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Annual Cryptology Conference*. Springer, 2011, pp. 578–595.

[13] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 463–477.

[14] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44–55.

[15] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/216, 2003, available at http://eprint.iacr.org/2003/216.

[16] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *ACM CCS 06: 13th Conference on Computer and Communications Security*, A. Juels, R. N. Wright, and S. Vimercati, Eds. Alexandria, Virginia, USA: ACM Press, Oct. 30 – Nov. 3, 2006, pp. 79–88.

[17] S. Kamara and K. Lauter, *Cryptographic Cloud Storage*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 136–149.

[18] A. Michalas and R. Dowsley, "Towards trusted ehealth services in the cloud," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, Dec 2015, pp. 618–623.

[19] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology – CRYPTO 2013, Part I*, ser. Lecture Notes in Computer Science, R. Canetti and J. A. Garay, Eds., vol. 8042. Santa Barbara, CA, USA: Springer, Berlin, Germany, Aug. 18–22, 2013, pp. 353–373.

[20] R. Dowsley, A. Michalas, and M. Nagel, "A report on design and implementation of protected searchable data in iaas," Technical Report, 2016, available at http://soda.swedishict.se/5921/1/T2016_01.pdf.

[21] H. Hacigümüs, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 29–38.

[22] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 720–731.

[23] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 85–100.

[24] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services," *CIDR 2005*, 2005.

[25] V. Ciriani, S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Combining fragmentation and encryption to protect privacy in data storage," *ACM Transactions on Information and System Security*, vol. 13, no. 3, pp. 1–33, Jul. 2010.

[26] R. Dowsley, M. Gabel, G. Hübsch, G. Schiefer, and A. Schwichtenberg, "A distributed key management approach," Manuscript, 2016.