Towards Secure Cloud Database with Fine-Grained Access Control

Michael G. Solomon, Vaidy Sunderam, Li Xiong

Towards Secure Cloud Database with Fine-Grained Access Control

Michael G. Solomon, Vaidy Sunderam*, and Li Xiong**

Department of Mathematics & Computer Science
Emory University
Atlanta, Georgia 30322, USA
{msolo01,vss,lxiong}@emory.edu

Abstract. Outsourcing data to cloud environments can offer ease of access, provisioning, and cost benefits, but makes the data more vulnerable to disclosure. Loss of complete control over the data can be offset through encryption, but this approach requires an omniscient third party key authority to handle key management, increasing overhead complexity. We present the ZeroVis framework that provides confidentiality for data stored in a cloud environment without requiring a third party key manager. It combines fine-grained access control with the ability to search over encrypted data to allow existing applications to migrate to cloud environments with very minimal software changes, while maintaining data provider control over who can consume that data.

Keywords: Confidentiality, Searchable Encryption, Ciphertext Policy, Fine-grained Access Control, Cloud

1 Introduction

An agreement with a Cloud Service Provider (CSP) [13] to store data in a public, community, or hybrid cloud environment can provide the benefits of outsourced maintenance and capability to alter capacity based on demand [3]. However, the cost of outsourcing data storage is diminished control over data security [25, 16]. CSP environments are untrusted [10] in which local levels of control cannot be attained [17, 16]. Traditional access control methods are often insufficient for CSP [17, 19] hosted databases. Lacking sufficient confidentiality controls not only exposes the data to additional vulnerabilities, but is also possibly a violation of laws, regulations, or contract terms [22].

* Research supported in part by NSF grant OCI-1124418 and AFOSR DDDAS grant FA9550-12-1-0240
** Research supported in part by NSF grant CNS-1117763 and AFOSR DDDAS grant FA9550-12-1-0240

The primary challenge is to extend confidentiality assurances into untrusted domains [19]. Since different data consumers have different privileges, data access must be individualized and restricted to authorized consumers. And to be functionally effective, the protected data must be searchable without incurring excessive overhead or exposing any of the protected data to any entities in the untrusted environment [5].

A common method to protect data in any untrusted environment is to encrypt data before sending it outside the trusted domain [9]. In multi-user database scenarios, solutions using most traditional encryption implementations are suboptimal, requiring an additional key-management layer thereby degrading performance and scalability [32, 20, 8].

Traditional data encryption techniques require a single key, or a pair of keys, to encrypt and decrypt each data item. The most fine-grained approach to using encryption for data stored in a database requires a separate key for each cell (each column within a row), and a trusted key authority to store keys and manage access to them based on predetermined access criteria. The opposite extreme approach would be to use a single key or key pair to encrypt and decrypt all protected cells in the database, an approach similar to various transparent data encryption schemes [11, 7]. This approach makes it easier to manage keys but introduces a single point of compromise. A balance between the two extremes is to define partitions of encrypted data (the set of encrypted cells in a database that share the same encryption/decryption key), and are often implemented as roles [30]. While this approach is a good compromise between minimum and maximum granularity, the common use of key access managers does still grant access control authority to the key access manager, instead of giving the authority to the data provider.

Table 1. Data Access Example: Research teams and contexts of interest

Team	**Treatment**	**Research Context**	**Description**
A	Z51.11	Cancer	Effectiveness of different treatment
B	E66.09	Nutritional health	Impact of obesity on heart disease
C	Z51.11	Tobacco use AND Heart disease	Impact of lifestyle and heart disease on cancer treatment effectiveness
D	Modified Diet	Nutrition	Measurable benefits of various diets

Running Example. Difficulties with balancing data protection and ease of access are common in medical data collection. Consider the following scenario. Four research teams, A, B, C, and D, need patient data. Table 1 shows four research teams, along with the specific treatments they

are studying and the general context of their work. The primary challenge to be addressed is to obtain current data that is pertinent to their research, while complying with HIPAA rules and patient constraints. Patient constraints allow a patient to control who can access her data, such as researchers, medical service providers, and next of kin to access her data. A patient can submit her data with constraints, such as "allow authorized cancer researchers to access my data".

Contributions. We propose a framework that addresses the need of confidentiality in an untrusted environment along with maintaining *data provider control* over *data consumer access* without an omniscient key manager. Our framework, termed ZeroVis[1], combines the ability to search across encrypted data [24] with fine-grained access control [1] to provide confidentiality protection, searchability for efficient access, and data owner initiated access control, all in an untrusted storage environment. Our framework will provide a one-to-many (one data provider to many data consumers) data confidentiality layer that can be accessed by existing legacy applications to allow current host-bound applications to migrate to a cloud storage environment and maintain confidentiality.

Our framework does not require a trusted third party to manage encryption keys for data providers and consumers. Nor does it require specific permission for each new data consumer (e.g. research team). In essence, each data provider (patient) specifies an access policy (based on attributes rather than identities) for her data that determines who can access protected portions of her data. Traditional key management schemes require a key manager to associate authorized data providers and authorized data consumers with keys (a many-to-many relationship). Our framework assumes the existence of an attribute manager that maintains valid attributes for authorized data consumers (instead of many keys), regardless how many partitions they can access.

Given our running example, assume that a patient received treatment at an oncologist's office. The patient specified that the data to describe and record the visit is saved with the following access policy:
"treatment='Z51.11' AND (context=cancer OR context=tobacco use)" (i.e. only data consumers that possess the treatment attribute with a value of Z51.11[2] and the context attribute with either the values of cancer or tobacco can access her data.)

[1] Like flying an instrument approach with limited or no visibility - only pilots with proper equipment, clearance, and the current local frequencies can land.

[2] Z51.11 is the ICD10 code for "Encounter for antineoplastic chemotherapy", which also corresponds to the ICD9 code V58.11.

Our framework proposes the use of CP-ABE (Ciphertext Policy Attribute Based Encryption) [1] to control access to data based on the data consumer's attributes. Only consumers who possess attributes that satisfy the ciphertext's access policy can decrypt. In the running example, all research teams can retrieve any encrypted database row. However, only A and C can decrypt the data since their attributes (treatment and context) satisfy the CP-ABE access policy. Our framework also utilizes layered encryption in combination with CP-ABE to support efficient query processing on encrypted data. In this paper we present an implementation of our framework and a performance study with different database sizes the demonstrate the feasibility of our proposed approach.

2 Related Work

The ZeroVis framework most closely relates to searchable encryption and distributed/federated encryption key management. Broadcast encryption [12] was first proposed as a solution to the problem of sending secure transmissions from one site to an arbitrary number of recipients. This scheme is similar to ours, but differs in its reliance on a known hierarchical distribution pattern and set of privileged users. Later work based on [12] increases scalability [23] [18] and even integrate Attribute Based Encryption techniques for greater utility [31].

Attribute Based Encryption (ABE) [14] addressed the problem of encrypting data for an arbitrary number of recipients. Unlike broadcast encryption, ABE keys are derived instead of simply shared. Goyal proposed an extension to Identity Based Encryption (IBE) [2] that uses attributes and access policies, not distinct identities, to encrypt and decrypt data. The two primary forms of ABE are Ciphertext Policy ABE (CP-ABE) and Key Policy ABE (KP-ABE). KP-ABE embeds the access policy in the user's private key [14], while CP-ABE embeds the access policy in the ciphertext [1]. KP-ABE gives control over who can decrypt data to the key generator, while CP-ABE ensures that the encryptor (data owner) retains control over who can decrypt her data [1]. ABE solves the problem of providing access to private data for specified recipient without traditional key management issues, and is proposed in several outsourcing secure data schemes [27, 15, 28], but the technique alone does not map well to encrypting data for storage in a database due to its lack of a mechanism to efficiently search encrypted data. Li et. al. [21] uses both CP-ABE and KP-ABE schemes to store personal health record (PHR) data in a semi-trusted environment. Their proposed framework extends

the basic ABE notion to include Multi-Authority ABE (MA-ABE) [4] to allow different attribute authorities with different data needs to collectively generate users' secret keys based on distinct sets of user attributes. This approach of securing PHR data focuses primarily on storing documents and does not address the problem of efficiently searching across many PHR data items.

CryptDB [24] is research software that addresses the performance limitations of accessing encrypted data stored in a database. Multiple copies of each encrypted column are stored, using different encryption algorithms, to support many requirements of common application queries. Although CryptDB does solve the access performance issue, it relies on distinct keys that are bound to user identities. Further, CryptDB focuses primarily on transaction related queries. The Monomi [26] project uses many of CryptDB's techniques to address analytical queries, extending the CryptDB concept by splitting query processing between the server and the client. While more scalable than CryptDB for analytical queries, it still does not provide a scalable method for one-to-many encryption.

Verifiable Attribute-based Keyword Search over Outsourced Encrypted Data (VABKS) [29] uses ABE to provide access control and solves the problem of searching across encrypted data in the cloud by adding encrypted keyword indexes to the ABE payload. While VABKS does provide searchability for ABE encrypted data, the technique is document-centric, requiring a defined list of searchable keywords for each ABE item, limiting its usefulness for searching across many database items.

Our approach builds on selected concepts from each of the above, and adds data provider controlled access control to better address efficient encrypted data access and overcome difficulties associated with distributed access control.

3 Problem Definition and Building Blocks

Consider a database D, with tables T_1 .. T_i. Each Table contains rows R_1 .. R_j, each with columns C_1 .. C_k. Clients access the database contents as data providers (DP), data consumers (DC), or as both roles. Data providers store data in the database (INSERT, UPDATE), and data consumers retrieve data from the database (SELECT). In a database that uses client-based encryption to protect stored data, clients access data in one or more columns (C_1, C_2, ..., C_k) from one or more rows (R_1, R_2, ..., R_j) from one or more tables (T_1, T_2, ..., T_i). Data providers encrypt data before storing it in the database and data consumers must decrypt data after retrieving it from the database. In this model, the database only

stores encrypted versions of protected cells and never sees the plaintext version of the data. The primary problem with this approach is in the difficulty of generating and managing the keys to encrypt and decrypt data. Data providers and data consumers must share keys to access data, and the number of keys grows with a higher level of desired fine-grained access (i.e. a need for more encryption partitions.)

We built the ZeroVis framework on two primary building blocks, Ciphertex Policy Attribute Based Encryption (CP-ABE), and CryptDB. Each component brings desirable features to ZeroVis, but neither one solves our problem alone.

Ciphertex Policy Attribute Based Encryption. A CP-ABE scheme provides fine-grained access control over data [1]. CP-ABE associates a user with a set of descriptive attributes to generate the user's secret key, SK. Data are encrypted under an access policy such that only users whose attributes match the access policy can decrypt the data. To encrypt a message M using CP-ABE, the encryptor provides an access policy which is expressed as a boolean expression containing selected attributes and values for M. Figure 1 shows the access policy presented earlier in a tree structure. The message is then encrypted based on the access structure, T. Decryptors generate SK based on their attributes. A decryptor is only able to decrypt ciphertext, CT, when her SK satisfies the access policy used to encrypt the message. Unauthorized users cannot decrypt CT even if they collude and combine their disjoint attributes.

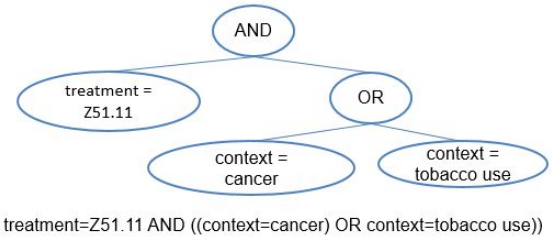

Fig. 1. CP-ABE Access Tree)

CP-ABE defines the following four essential functions:

1. Setup(): Input security parameter, output public parameter (PK), for encryption, and master key (MK), to generate user secret keys.
2. Encrypt: Input message M, access structure T, public parameter PK, output ciphertext CT.
3. KenGen: Input set of user's attributes SX and MK, output secret key SK for SX.

4. Decrypt: Input CT, SK. If SK satisfies access structure in CT, return M, else return NULL.

CP-ABE works well for encrypting individual shared data where the file's name or identifier is known, but there is no provision for searching ciphertext, thereby making CP-ABE alone insufficient for database queries.

CryptDB. CryptDB is a DBMS that provides confidentiality for data stored on an untrusted database server [24]. The system provides near-transparent confidentiality by intercepting database queries and rewriting them in such a way as to execute over encrypted data. Decryption for consumption never occurs on the server, only at the trusted proxy. CryptDB also incorporates an encryption strategy that can adjust the encryption level of each column based on user queries. At runtime, the CryptDB proxy analyzes each query and determines the encryption needs based on the query components. The proxy will either then map each query component to an encrypted data item or request an encryption layer adjustment. All data is initially stored by CryptDB encrypted into several layers, with each layer encrypted with one of six encryption methods. The resulting value is called "encryption onion". CryptDB will only "peel" an onion layer (decrypt the outer layer) if a query requires an inner layer to successfully complete. This dynamic ability to alter encryption layers gives CryptDB the flexibility to maintain confidentiality while still responding to query requirements. The database server peels onion layers with user defined functions, and will never remove the innermost layer that would expose the original plaintext. Although CryptDB does provide the ability to select and search encrypted data on an untrusted sever, it still requires user-based encryption keys. CryptDB must rely on an external authority to enforce key management, including authorizing multiple consumers to decrypt a provider's data.

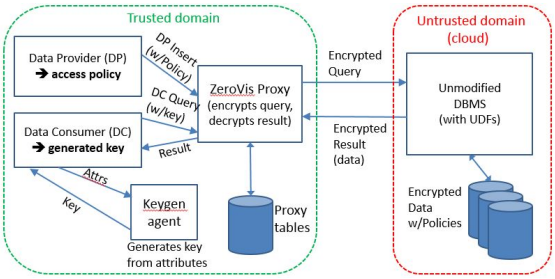

Fig. 2. ZeroVisibility Cloud Framework.

4 ZeroVis Framework

4.1 Framework Overview

To overcome the problems described in the previous section, our approach integrates CP-ABE with the ability to search across encrypted data, e.g. as provided in CryptDB, to synthesize a solution that supports single data provider encryption accessible by multiple data consumers for data stored in an untrusted environment, along with the ability to efficiently retrieve the data without decrypting in the cloud.

Figure 2 shows the ZeroVis framework. The core of our framework is the ZeroVis proxy which is responsible for encrypting data and queries and decrypting query results. The data provider submits data along with access policies through ZeroVis Proxy which encrypts the data via CP-ABE and searchable encryption and stores the encrypted data through an unmodified DBMS. A data consumer submits a query along with a pre-generated secret key, SK, (generated from the data consumer's attributes) through the ZeroVis proxy which encrypts the query. The DBMS returns encrypted results of the query to the ZeroVis proxy, which decrypts the results and returns the plaintext to the data consumer.

One additional requirement of a complete framework in a production environment is an Attribute Authority (AA). The AA is responsible for authorizing users, and managing attributes associated with those users. The framework depends on the AA to supply authenticated attributes for each authenticated user, and to prevent unauthorized users from submitting queries through the framework. Users can submit queries directly to the untrusted DBMS, but without the necessary master key from the AA, decryption attempts are unsuccessful.

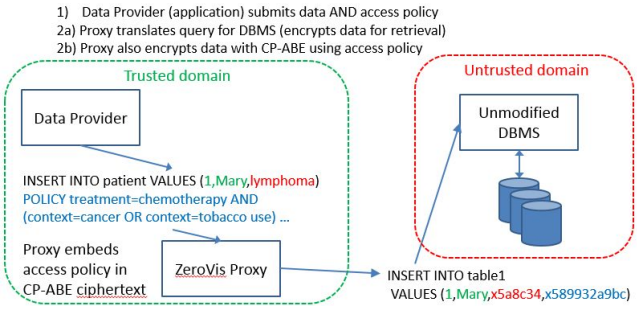

Fig. 3. Submitting Data (INSERT)

4.2 Data Insertion and Encryption

To encrypt data, the DP provides the trusted proxy with the plaintext data and an access policy. Figure 3 shows the data flow with an example INSERT query. The trusted proxy encrypts the plaintext data, translates the query components into their encrypted counterparts (for query elements that are stored encrypted in the DBMS), and submits the encrypted payload, along with the embedded access policy, to the DBMS. Notice in Figure 3 there are 2 ciphertext values. The first represents existing CryptDB encryption and the second depicts the new CP-ABE CT added by ZeroVis.

4.3 Data Retrieval and Decryption

To decrypt data, a DC must first generate a secret key, SK, based on her attributes. In most implementations, a trusted AA will generate a key for each identity upon new user registration. The DC provides a set of descriptive attributes, SX, such as treatment and context interest areas (for our running example). Attributes can describe an entity's state, status, or authorized interest areas. The AA generates SK based on the supplied SX and returns SK to the DC on demand. For example, a research team member may possess attributes "treatment=Z51.11, context=cancer".

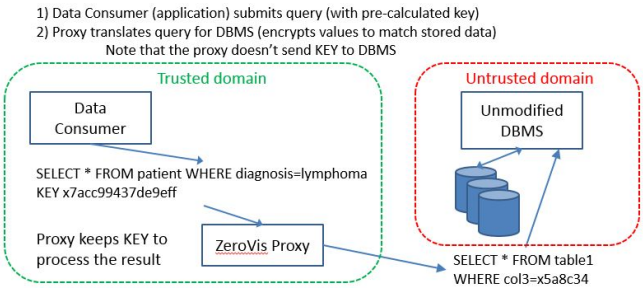

Fig. 4. Retrieving Data (SELECT request)

The DC then presents SK (generated by the AA) to the trusted proxy when attempting to access encrypted data. The trusted proxy translates the supplied query elements into their encrypted counterparts (for query elements that are stored encrypted in the DBMS), and submits the query, depicted in figure 3. The proxy then translates the returned data from the encrypted state, CT, as stored in the DBMS, depicted in figure 4, into plaintext state, M, for the application. The CP-ABE decryption algorithm will only return plaintext message, M, when the supplied SK satisfies the data's embedded access policy that was provided by the DP. If the

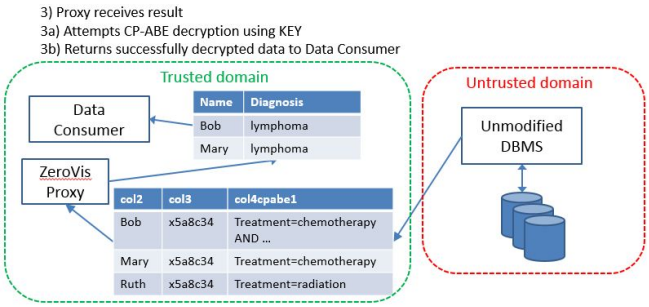

Fig. 5. Retrieving Data (SELECT response)

supplied key does not satisfy the access policy the proxy simply returns a null value.

The process of modifying data (UPDATE) is essentially a combination of a data retrieval operation followed by a data submission operation. While the process of updating data is straightforward, the implementation of the framework would need to ensure updates are well-behaved and do not allow unauthorized data or policy modifications. Users updating data must possess SK to retrieve data and an access policy to encrypt changes. Traditional access controls would be necessary to limit data and policy updates to authorized users.

4.4 Implementation

Our test implementation of ZeroVis was built on the architecture described above. The data consumer issues queries to the ZeroVis proxy. The ZeroVis proxy re-writes each query and submits it to the mySQL database server. We built the ZeroVis proxy by modifying the CryptDB proxy, which was built by modifying mysql-proxy. Both CryptDB and ZeroVis can be implemented using other proxy software and any DBMS the chosen proxy supports. Both the ZeroVis proxy and the MySQL database server run on computers running Linux. ZeroVis supports both interactive clients through a shell prompt and existing applications through a connection to the proxy. Both client types require that users register with an AA.

We implemented the ZeroVis framework by integrating CP-ABE into CryptDB proxy. CryptDB provides query re-writing and capability to search across encrypted data. The addition of CP-ABE as a new encryption method within CryptDB gives the framework one-to-many encryption capability. The first change to CryptDB was to create a new column

for each protected column. CryptDB normally creates 2 or 3 columns to store encrypted data using different methods to support different types of queries. The new column for each plaintext column stores the CP-ABE CT. We added a new encryption layer, ABE, to each onion definition, added a new ABE security level, and added a new class to handle CP-ABE encryption and decryption operations. The new class uses cpabe-toolkit functions to encrypt and decrypt data. We modified the CryptDB proxy query re-writing code to replace requested columns with CP-ABE columns. We retain the CryptDB obfuscated column names in the queries to allow the database to select data using searchable data. The database then returns only CP-ABE encrypted data. The proxy attempts to decrypt each column and returns successfully decrypted data to the client.

With the new functionality in place to handle CP-ABE, we extended the proxy to fetch the user's CP-ABE SK, based on the MySQL database user id. The private key will be provided by the AA in more robust implementations. Additional modifications to the proxy also fetch and store the current user's default access policy for CP-ABE encryption operations. The ZeroVis system currently creates CP-ABE CT for every encrypted column. The CP-ABE encryption uses the current user's access policy. Decryption uses the current user's SK, previously generated using the CP-ABE keygen() function. Future work will extend the supported SQL syntax to allow users to optionally provide access policies with every query.

5 Performance Results

Experiment Setup. Our performance assessment is based on a straightforward CP-ABE addition to CryptDB as described above. Our goal was to determine the additional overhead CP-ABE added to the existing CryptDB implementation. We created multiple copies of test databases, all based on subsets of the TPC-C[6] benchmark database. Test databases of different sizes were built by altering the number of rows in the item, warehouse, and district tables, all based on cardinality relationships defined in the TPC-C specification. The resulting 5 test databases DB-a, DB-b, DB-c, DB-d, DB-e have row cardinality of 1912, 2975, 7156, 18622, 35756 respectively, which are approximately increasing in a logarithmic scale. We created sets of queries, both single row and multiple row returned sets, to assess the general performance of the ZeroVis framework. The queries were simple, single table INSERT statements to load varying size subsets of the TPC-C database, and single table SELECT statements

to retrieve 1 row (150 SELECTs) and sets (150 SELECTs) from the item, stock, and customer tables. The SELECT queries to retrieve sets of rows were randomly generated to select a range from the domain of each table. The test sever had an Intel Core 2 2.0 GHz(x2) processor with 3GB RAM running Ubuntu 13.10. The client/proxy computer had an Intel Core i7 2.4 GHz(x8) processor with 16GB RAM running Ubuntu 13.10. The two computers were connected via a 100Mbit/s Ethernet connection.

Results. Adding CP-ABE results in an additional encryption operation for each protected column, adding substantial observed space and computation time overhead. CryptDB, without CP-ABE, is approximately 26% slower (throughput loss) than native MySQL [24] when running the TPC-C benchmark. Encryption and decryption times are linearly related to the number of leaf nodes in the CP-ABE access policy. According to Bethenourt et al, [1], their implementation of CP-ABE took approximately 0.5 seconds to encrypt a payload with 20 policy leaf nodes, while only taking 0.04 seconds to decrypt. One reason why the encryption operation is so much slower is that it includes parsing and processing the provided policy. The decrypt operation does not directly interact with attributes. Generating the key, based on supplied attributes, is a separate function that must be completed prior to any decryption attempt.

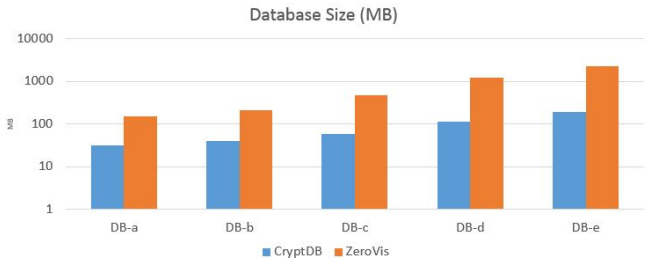

Fig. 6. Resulting DB Sizes for test DBs (of logarithmically increasing row cardinality)

Figure 6 shows the resulting database size (in MB) of the 5 test databases with varying row cardinality (approximately increasing in a logarithmic scale) for CryptDB (without CP-ABE) and ZeroVis (with CP-ABE) respectively. As the figure illustrates, the overhead incurred by ZeroVis increases linearly with the row cardinality. The current implementation stores a complete CP-ABE ciphertext payload for every protected database column, which includes the access policy and the encrypted data. Our future work will explore reducing redundancy through consolidating CP-ABE access policies which we expect will significantly decrease the overhead.

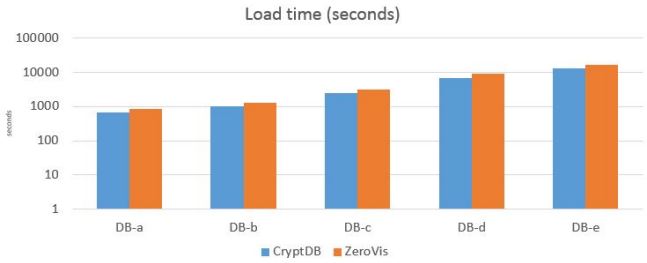

Fig. 7. Database load time

Figure 7 shows the load time for each database instance. The ZeroVis computational overhead is a result of the additional CP-ABE calculations. As mentioned above, the current test ZeroVis implementation constructs the access policy tree for each column, even if all columns share the same policy. It is expected that reducing CP-ABE access policy redundancy will also reduce computational overhead for future ZeroVis framework versions and result in ZeroVis performing more closely to CryptDB.

Fig. 8. Database Query time

Figure 8 shows times for queries that return single rows, and sets of rows (range queries). We submitted 300 SELECT queries for each database instance, 150 distinct queries and 150 range queries. The queries were scaled to consider the range of data stored in each database (randomly generated to exercise the full range of data in each table). Queries use both indexed and non-indexed criteria. The disparity between CryptDB and ZeroVis performance for range queries is due to ZeroVis' current larger data storage requirements. Additional tests with the proxy and sever running on a single machine showed that network costs were not responsible for the higher overhead of ZeroVis. The queries in our test returned most of the columns from each table, requiring CP-ABE decryption operations for each column. While decrypting multiple columns is normal expected behavior, the redundancy of storing and transporting

multiple copies of the access policy for each column increases the workload.We believe reducing redundant CP-ABE operations and normalizing the access policy storage technique will reduce ZeroVis' computational overhead and additional costs of the framework, resulting in performance closer to CryptDB than the current ZeroVis implementation.

6 Conclusions and Future Work

In this paper we showed how combining CP-ABE with encrypted data searching solves the problem of storing and retrieving confidential data from an untrusted environment, while giving the data provider control over who accesses her data. While other frameworks provide some of these capabilities, ours is the only one to our knowledge that accomplishes this without relying on traditional key management techniques. Our framework is the first to specifically address the need for one-to-many encryption in a database environment, which requires support for efficient queries across encrypted data.

This paper describes the initial ZeroVis framework implementation. Future framework changes are necessary to create a more production viable framework.A specific requirement for a trusted AA needs to be included. Although we only generally described the need for the AA,the AA will be an integral component of a completed framework. It will be responsible for authorizing users, securely storing their attributes, and providing the ZeroVis proxy with sufficient authentication information and attributes to properly handle encryption and decryption operations for authorized users. The AA will act as the layer of protection that stops attackers from arbitrarily providing unauthorized attributes to the CP-ABE encryption/decryption functions. The AA will also manage the master key required for encryption/decryption operations.

Additional work is necessary to reduce the storage and computational overhead of CP-ABE. Others have already studied this problem, including Constant-size CP-ABE (CP-ABE) [31] and techniques discussed in [18] and [1]. We will also explore normalizing the CP-ABE ciphertext,which is currently a concatenation of the access policy and the encrypted payload. The access policy comprises over 90% of the ciphertext size. Denormalizing the CP-ABE ciphertext will reduce the storage (and network transmission) requirements for multiple columns that share the same access policy.

Bibliography

[1] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[3] M. Carroll, A. van der Merwe, and P. Kotze. Secure cloud computing: Benefits, risks and controls. In *Information Security South Africa (ISSA), 2011*, pages 1–9, Aug 2011.

[4] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 121–130, New York, NY, USA, 2009. ACM.

[5] Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, and Jesus Molina. Controlling data in the cloud: Outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 85–90, New York, NY, USA, 2009. ACM.

[6] Transaction Processing Performance Council. Tpc benchmark c, standard specification version 5, 2001.

[7] Dr Deshmukh, Anwar Pasha, Dr Qureshi, et al. Transparent data encryption–solution for security of database contents. *arXiv preprint arXiv:1303.0418*, 2013.

[8] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Over-encryption: Management of access control evolution on outsourced data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 123–134. VLDB Endowment, 2007.

[9] R.A. Elmasri and S.B. Navathe. *Fundamentals of Database Systems [With Access Code]*. ADDISON WESLEY Publishing Company Incorporated, 2011.

[10] M.R. Farcasescu. Trust model engines in cloud computing. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 465–470, Sept 2012.

[11] Luca Ferretti, Michele Colajanni, Mirco Marchetti, and Adriano Enrico Scaruffi. Transparent access on encrypted data distributed over multiple cloud infrastructures. In *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 201–207, 2013.

[12] Amos Fiat and Moni Naor. Broadcast encryption. In DouglasR. Stinson, editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin Heidelberg, 1994.

[13] B. Gowrigolla, S. Sivaji, and M.R. Masillamani. Design and auditing of cloud computing security. In *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*, pages 292–297, Dec 2010.

[14] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[15] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes. *Univeristy of Twente, Tech. Rep*, 2009.

[16] Wayne Jansen, Timothy Grance, et al. Guidelines on security and privacy in public cloud computing. *NIST special publication*, 800:144, 2011.

[17] K.M. Khan and Q. Malluhi. Establishing trust in cloud computing. *IT Professional*, 12(5):20–27, Sept 2010.

[18] Jongkil Kim, Willy Susilo, ManHo Au, and Jennifer Seberry. Efficient semi-static secure broadcast encryption scheme. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography Pairing 2013*, volume 8365 of *Lecture Notes in Computer Science*, pages 62–76. Springer International Publishing, 2014.

[19] G. Kulkarni, N. Chavan, R. Chandorkar, R. Waghmare, and R. Palwe. Cloud security challenges. In *Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on*, pages 88–91, Oct 2012.

[20] Wei-Bin Lee and Chien-Ding Lee. A cryptographic key management solution for hipaa privacy/security regulations. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):34–41, Jan 2008.

[21] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):131–143, Jan 2013.

[22] T. Mather, S. Kumaraswamy, and S. Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. Theory in practice. O'Reilly Media, 2009.

[23] Duong-Hieu Phan, David Pointcheval, SiamakF. Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. *International Journal of Information Security*, 12(4):251–265, 2013.

[24] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

[25] Zhidong Shen and Qiang Tong. The security of cloud computing system enabled by trusted computing technology. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–11–V2–15, July 2010.

[26] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. *Proc. VLDB Endow.*, 6(5):289–300, March 2013.

[27] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[28] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 261–270, New York, NY, USA, 2010. ACM.

[29] Qingji Zheng, Shouhuai Xu, and Giuseppe Ateniese. Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. Cryptology ePrint Archive, Report 2013/462, 2013. `http://eprint.iacr.org/`.

[30] Lan Zhou, Vijay Varadharajan, and Michael Hitchens. Enforcing role-based access control for secure data storage in the cloud. *The Computer Journal*, 54(10):1675–1687, 2011.

[31] Zhibin Zhou and Dijiang Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption: Extended abstract. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 753–755, New York, NY, USA, 2010. ACM.

[32] Xukai Zou, Yuan-Shun Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 538–546, April 2008.