International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 27005-27016

http://www.ripublication.com

Evaluation Of Data Encryption Techniques In Cloud Database Services

Dr. A. Bhuvaneswari[1] and M. Pradeepa[2]

[1] Professor / CSE Adhiparasakthi Engineering College.
[2]PG Scholar, Adhiparasakthi Engineering College.
Email Address:
[1] bhuvan@adhiparasakthi.in (Dr.A. Bhuvaneswari),
[2]pradee027@gmail.com (M. Pradeepa)

Abstract

The user's perception that the confidentiality of their data is endangered by internal and external attacks is limiting the diffusion of public cloud database services. In this context, the use of cryptography is complicated by high computational costs and restrictions on supported SQL operations over encrypted data. Database as a Service paradigm (DBaaS) that poses several research challenges in terms of security and cost evaluation from a tenant's point of view. Here the proposed work implements and evaluate the Performance of symmetric and Asymmetric Encryption Techniques. Encryption schemes represent a feasible solution for achieving data confidentiality in public cloud databases even from a performance point of view. Additive Homomorphic Encryption Techniques Out Performance the Symmetric Encryption Techniques.

I Introduction

Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services, for example a chain of different services from different companies could calculate 1) the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services. Homomorphic encryption schemes are malleable by design. This enables their use in cloud computing environment for ensuring the confidentiality of processed data. In addition the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems, collision-resistant hash functions, private information retrieval schemes, and many ore. There are several partially homomorphic cryptosystems, and also a number of fully homomorphic cryptosystems. Although a cryptosystem which is unintentionally malleable can be subject to attacks on this basis, if treated carefully homomorphism can also be used to perform computations securely.

Fully homomorphic encryption theoretically supports any kind of computation over encrypted data but it is computationally unfeasible, because it increases the computational cost of any operation by many orders of magnitude. Other schemes which allow the execution of computations over encrypted data limit the type of allowed operations (e.g., order comparisons in , sums in, search).

II Related Work

Improving the confidentiality of information stored in cloud databases represents an important contribution to the adoption of the cloud as the fifth utility because it addresses most user concerns. Our proposal is characterized by two main contributions to the state of the art: architecture and cost model. Although data encryption seems the most intuitive solution for confidentiality, its application to cloud database services is not trivial, because the cloud database must be able to execute SQL operations directly over encrypted data without accessing any decryption key. The tenant has two alternatives for any SQL operation : downloading the entire database, decrypting it, executing the query and, if the operation modifies the databases, encrypting and uploading the new data; decrypting temporarily the cloud database, executing the query, and re-encrypting it. The former solution is affected by huge communication and computation overheads, and costs that would make the cloud database services quite inconvenient; the latter solution does not guarantee data confidentiality because the cloud provider obtains decryption keys. The right alternative is to execute SQL operations directly on the cloud database, but avoiding that the provider obtains the decryption key. This proposal is based on data aggregation techniques associate plaintext metadata to sets of encrypted data to allow data retrieval. However, plaintext metadata may leak sensitive information and data aggregation introduces unnecessary network overheads. The use of fully homomorphic encryption would guarantee the execution of any operation over encrypted cloud data, but existing implementations are affected by huge computational costs to the extent that they would make impractical the execution time of SQL operations over a cloud database. Other encryption algorithms characterized by acceptable computational complexity support a subset of SQL operator.

III Techniques and Algorithm

AES Encryption Advanced Encryption Standard is the new encryption standard. It has variable key length of 128, 192, or 256 bits; default 256. AES encryption is fast and flexible; it can be implemented on various platforms especially in small devices Brute force attack is the only effective attack known against it, in which the attacker tries to test all the characters combinations to unlock the encryption.

AES operates on a 4×4 column-major order matrix of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field. The Encryption and decryption process consists of a number of different transformations applied consecutively over the data block bits, in a fixed number of iterations, called rounds. The number of rounds depends on the length of the key used for the encryption process. For key length of 128 bits, the number of iteration required are 10. (Nr = 10). Each of the first Nr-1 rounds consist of 4 operations: SubBytes ( ), ShiftRows ( ), MixColumns ( ) & AddRoundKey ( ).

The four different transformations are described in detail below:

1) **SubBytes** ( ): It is a non-linear substitution of bytes that operates independently on each byte of the State using a substitution table (S box). This S-box which is invertible is constructed by first taking the multiplicative inverse in the finite field GF (28) with irreducible polynomial m(x) = x8 + x4+ x3 + x + 1. The element {00} is mapped to itself. Then affine transformation is applied (over GF (2)).
2) **ShiftRows** (): Cyclically shifts the rows of the State over different offsets. The operation is almost the same in the decryption process except for the fact that the shifting offsets have different values.
3) **MixColumns ()** : This transformation operates on the State column-by-column, treating each column as a fourterm polynomial. The columns are considered as polynomials over GF (28) and multiplied by modulo x4 + 1 with a fixed polynomial a(x) = {03} x3+ {01} x2+ {02} x.
4) **AddRoundKey** (): In this transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key expansion. Those Nb words are each added into the columns of the State. Key Addition is the same for the decryption process.
5) **Key Expansion**: Each round key is a 4-word (128-bit) array generated as a product of the previous round key, a constant that changes each round, and a series of S-Box lookups for each 32-bit word of the key. The Key schedule Expansion generates a total of Nb (Nr + 1) words. The decryption process is direct inverse of the encryption process. All the transformations applied in encryption process are inversely applied to this process. Hence the last round values of both the data and key are first round inputs for the decryption process and follows in decreasing order.

BLOWFISH ENCRYPTION

Blowfish encryption is a fast and alternative to existing encryption algorithms such AES, DES and 3 DES etc.

Blowfish is a symmetric block encryption algorithm designed in consideration with,

- **Fast :** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.
- **Compact:** It can run in less than 5K of memory.
- **Simple:** It uses addition, XOR, lookup table with 32-bit operands.
- **Secure:** The key length is variable ,it can be in the range of 32~448 bits: default 128 bits key length.

the left shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two sub key arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XOR with one of the two remaining unused P-entries and the upper right shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 2^{32} and XOR to produce the final 32-bit output.

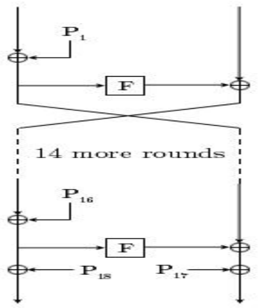

Figure 1 : The Feistel structure of Blowfish

Description of Algorithm

Blowfish symmetric block cipher algorithm encrypts block data of 64-bits at a time.it will follows the feistel network and this algorithm is divided into two parts.

1. Key-expansion
2. Data Encryption

Key-expansion

It will converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. Blowfish uses large number of subkeys.

These keys are generate earlier to any data encryption or decryption.

The p-array consists of 18, 32-bit subkeys:

P1,P2,…………,P18

Four 32-bit S-Boxes consists of 256 entries each:

S1,0, S1,1,………. S1,255

S2,0, S2,1,……….. S2,255

S3,0, S3,1,……….. S3,255

S4,0, S4,1,.............S4,255

Generating the Subkeys

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

Data Encryption:

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

Algorithm:Blowfish Encryption

Divide x into two 32-bit halves: xL, xR

For i = 1to 16:

xL = XL XOR Pi

xR = F(XL) XOR xR

Swap XL and xR

Swap XL and xR (Undo the last swap.)

xR = xR XOR P17

xL = xL XOR P18

Recombine xL and xR

PAILLIER ENCRYPTION

1. Choose two large prime numbers p and q randomly and independently of each other such that **gcd(pq,(p-1)(q-1))=1** . This property is assured if both primes are of equal length.
2. Compute **n=pq and λ-lcm(p-1,q-1)** .
3. Select random integer **g** where $\mathbf{g \in Z^*_{n2}}$
4. Ensure **n** divides the order of **g** by checking the existence of the following modular multiplicative inverse: $\mathbf{\mu=(L(g^{\lambda}\ mod\ n^2))^{-1} mod\ n}$ where function L is defined as **L(u)=u-1⁄n** .

Note that the notation **a⁄b** does not denote the modular multiplication of **a** times the modular multiplicative inverse of **b** but rather the quotient of **a** divided by **b**, i.e., the largest integer value **v≥0** to satisfy the relation **a≥b**.

- The public (encryption) key is (**n,g)**
- The private (decryption) key is **(λ,μ**)

If using p,q of equivalent length, a simpler variant of the above key generation steps would be to set **g = n+1, λ=** □ **(n)** and **μ=** □ **(n)-1 mod n**, where □ **(n)=(p-1)(q-1) .**

ENCRYPTION

1. Let m be a message to be encrypted where $\mathbf{m \in Z_n}$
2. Select random **r** where $\mathbf{r \in Z_n^*}$
3. Compute ciphertext as: $\mathbf{c=g^m.r^n\ mod\ n^2}$

DECRYPTION

1. Let c be the ciphertext to decrypt, where $\mathbf{c \in Z^*_n{}^2}$
2. Compute the plaintext message as: $\mathbf{m = L(c^{\lambda} mod\ n^2).\mu\ mod\ n}$

As the original paper points out, decryption is essentially one exponentiation modulo $\mathbf{n^2}$

Mode of Operation
1.CBC

Cipher Block Chaining
Cipher block chaining (CBC) is a mode of operation for a block cipher (one in which a sequence of bits are encrypted as a single unit or block with a cipher key applied to the entire block). Cipher block chaining uses what is known as an initialization vector (IV) of a certain length. One of its key characteristics is that it uses a chaining mechanism that causes the decryption of a block of cipher text to depend on all the preceding cipher text blocks. As a result, the entire validity of all preceding blocks is contained in the immediately previous cipher text block. A single bit error in a cipher text block affects the decryption of all subsequent blocks.
INPUT: plaintext x, key K
OUTPUT: cipher text y =
ASSUMED: round function g, last round h, key
scheduling procedure giving
for i = 1 to e K(x)
Nr −1
Ki w0= x
wi= g(wi−1,Ki), y = g(wNr−1,KNr−1)

IV Architecture
The architecture is proposed to guarantee data confidentiality through adaptive encryption methods in cloud database environments.

Architecture Model
The distributed architecture represented in fig.2 where it is assumed that independent and distributed clients (Client 1 to N) access a public cloud database service. All information (i.e., data and metadata) is stored after encryption in the cloud database. The proposed architecture manages five types of information.

- **Plain data**: The informative content provided by the client users.
- **Encrypted data**: The encrypted data that is stored in the cloud database.
- **Plain metadata**: All the information required by the clients to manage encrypted data on the cloud database.
- **Encrypted metadata**: The encrypted metadata that are stored in the cloud database.
- **Master key**: The encryption key of the encrypted metadata. Data transferred between the user application and the encryption engine. Data's are plain format. Encrypted database interface contacts the encryption engine that retrieves the encrypted metadata and decrypts it through the master key. The SQL operation on encrypted data and then to decrypt the results. Each plain database column is encrypted into one or more encrypted columns, each one corresponding to a different Onion, depending on the SQL operations and must be supported on that column .The results are returned to the user

application through the encrypted database interface.

All data and metadata stored in the cloud database are encrypted. Any application running on a legitimate client can transparently issue SQL operations (e.g., SELECT, INSERT, UPDATE and DELETE) to the encrypted cloud database through the encrypted database interface. Data transferred between the user application and the encryption engine are in plain format, whereas information is always encrypted before sending it to the cloud database. When an application issues a new SQL operation,

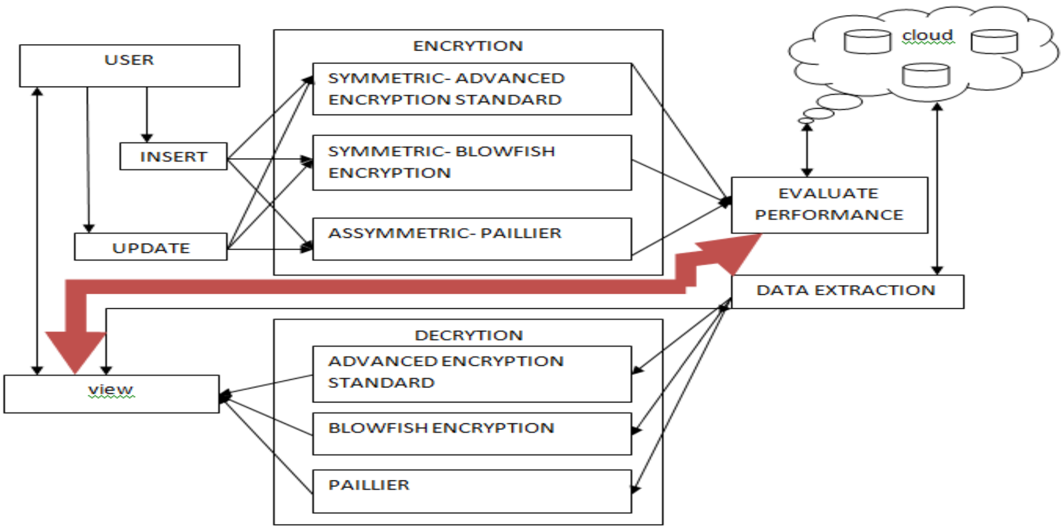

Figure 2: System Architecture design for Homomorphic Encryption

the encrypted database interface contacts the encryption engine that retrieves the encrypted metadata and decrypts it through the master key.

In order to improve performance, the plain metadata are cached locally by the client as a volatile information. After obtaining the metadata, the encryption engine is able to execute the SQL operation on encrypted data, and then to decrypt the results. The results are returned to the user application through the encrypted database interface. If each column of the database was encrypted through only one algorithm, then the database administrator would have to decide at design time which operations must be supported on each database column. However, this solution is impractical for scenarios in which the database workload changes over time.

V System Modules

A. Create Cloud Environment

Create a Cloud environment for cloud users using Cloud Data centers, Virtual Machines. Registration, login process is used to validate the clients in this module. Ubuntu operating system is used as cloud server and apache as web services and

MYSQL as database services. Ubuntu is the most popular operating system on the leading public clouds and the fastest growing scale-out platform in the world. Ubuntu's goal is to secure "out-of-the box". By default user's programs run with low privileges and cannot corrupt the operating system or other user's files.

B. Homomorphic Encryption

Develop and Implement the cloud environment with homomorphic Encryption scheme for Virtual Machines. Dynamic resource allocation will help to allocate the resources with Encrypted data. Homomorphic encryption theoretically supports any kind of computation over encrypted data, but it is computationally infeasible, because it increases the computational cost of any operation by many orders of magnitude. Homomorphic encryption would guarantee the execution of any operation over encrypted cloud data, but existing implementations are affected by huge computational costs to the extent that they would make the execution time of SQL operations over a cloud database impractical the execution time of SQL operations over a cloud database.

PERFORMANCE AND COST EVALUATION

The feasibility of the proposed cost model by applying it in the case of PLAIN, ENC and ADAPT configurations for real cloud database services. Then analyze the variations of costs for different cloud providers and resource usages. We finally evaluate tenant's costs over a mid-term period equal to three years by considering realistic resource usage increments and price reductions.

The cost is evaluated through the Performance. The cost of a cloud database service can be estimated as a function of three main parameters: **Cost = f(T ime, Pricing,Usage)** where:

- Time: identifies the time interval T for which the tenant requires the service.
- Pricing: It refers to the prices of the cloud provider for subscription and resource usage; they typically tend to diminish during T.
- Usage: denotes the total amount of resources used by the tenant; it typically increases during T .

PERFORMANCE EVALUATION

Performance Evaluation is a multi-purpose tool used to Measure actual performance against expected performance, Provide an opportunity for the employee and the supervisor to discuss job performance, Identify employee training and development needs, and plan for career growth, Identify skills and abilities for purposes of promotion, transfer, and reduction in force, Support alignment of organization and employee goals, Provide the basis for determining eligibility for compensation adjustments based on merit, Provide legal protection against lawsuits for wrongful termination, The primary purpose of Performance Evaluation is to provide an opportunity for open communication about performance expectations and feedback. Most employees want feedback to understand the expectations of their employer and to improve their own performance for personal satisfaction. Employees prefer feedback that is timely and given in a manner that is not threatening.

There are Many benefits result from the Performance Evaluation process Enhancement of employee motivation, commitment, and productivity, Identification of goals and objectives for the employee, Satisfaction of the basic human need for recognition, Identification of process improvement opportunities, Identification of employee development opportunities.

In this system the performance is evaluated by analyzing the process that are taken place in the cloud computation to complete the task. The task that are taken place in the computation are user creating new data, time taken by three algorithms to encrypt the data, time taken to decrypt the encrypted data and the total size consumed after the encryption. This process of analysis in this system takes place in evaluating all the data calculated at the each event takes place is the evidence for the performance over the algorithms on the data. This after performing an analysis the conclusion has been drawn based on this analytic reporting that which algorithm is good performance on which parameter and evaluates the brief conformation about each of the algorithms.

VI Experimental Results

SQL result

Host: localhost
Database: protrolo_test
Generation Time: Apr 08, 2015 at 09:54 AM
Generated by: phpMyAdmin 4.0.10.7 / MySQL 5.5.41-cll-lve
SQL query: SELECT `user`, `fname`, `feta1`, `feta2`, `feta3`, `fdta1`, `fdta2`, `fdta3`, `ol`, `el1`, `el2`, `el3` FROM `sakthi_vpall` LIMIT 0, 30 ;
Rows: 11

This table does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

user	fname	feta1	feta2	feta3	fdta1	fdta2	fdta3	ol	el1	el2	el3
testuser	sample2	5	4	3	3	1	1	18	65	65	65
pradee	omsakthi	2	1	2	1	4	1	38	89	89	89
pradee	amma	14	8	5	6	4	2	21	65	65	65
pradee	om	16	8	5	6	3	2	29	65	65	65
pradee	vp	14	9	5	5	4	2	18	65	65	65
lee	sample1	6	6	3	3	2	1	27	65	65	65
omsakthi	pradeepa	8	5	3	4	2	2	21	65	65	65
omsakthi	pradeepa	8	5	3	4	2	2	21	65	65	65
omsakthi	pradeepa	8	5	3	4	2	2	21	65	65	65
pradee	cloud	1	0	1	1	0	0	17	65	65	65
Pradeepa	abstract	14	11	7	12	5	3	1300	1808	1808	1808

Figure 3 Represents the Report of sql in Analyzing Performance.

REGISTRATION FORM: This form is an initialization to auto configuration on server for creating new users and authenticating the user. This is an form which contains text box for entering new username and password.

LOGIN FORM: This page is the gateway for accessing this service. The user should be pre registered on the signup registration form to access this service. This page consisted of username and password to authenticate the user and allow login.

HOME PAGE : This page is the home page which contains links to do activities of creating new file and encrypt wit all three algorithms before upload with user defined keyvalues And link to download them and decrypt them to view.

UPLOAD NEW DATA: This page is the contains activities of creating new file and encrypt with three algorithms before upload with user defined keyvalues.

COST AND PERFORMANCE EVALUATION: This page is an part in uploading data. This involves in evaluating the cost and performance of an individual data on different aspect with three algorithm as mentioned previously in this report

DECRYPT DATA AND DISPLAY: This page is an part in downloading data. This involves decrypting and downloading data with the input as file name and key. This makes the encrypted data to be decrypted at the client side.

EVALUATED REPORT: This is an evaluated result comparison on set of all data processed by all users and provides a brief comparison result to judge the algorithm which is working fine in processing data for encryption and decryption.

VII Discussion and Conclusion

Data confidentiality architecture allows to leverage adaptive encryption mechanisms while avoiding the use of any intermediate (trusted) proxy server to manage encryption details. There are several benefits characterizing the proposed architecture. It guarantees confidentiality of information stored in the cloud database, while allowing the execution of SQL operations over encrypted data. It simplifies database configuration, because it does not require to define manually at design time which operations should be allowed on each column. It guarantees best level of data confidentiality for any SQL workload. It does not require any intermediate (trusted) proxy to manage encryption details. Adaptive encryption is also affected by two major drawbacks. The first problem is that each plain column must be encrypted into one or more encrypted columns (Onions), thus increasing the overall database size up to one order of magnitude. This cost may be considered acceptable, or it can be reduced by the database administrator through a suitable tuning of the confidentiality parameters. The second problem is the performance overhead characterizing adaptive encryption that has to encrypt all parameters and decrypt the results of every SQL operation through all the Encryption Layers of each involved Onion. These costs prevent the use of adaptive encryption methods on most real contexts. However, that this overhead becomes less significant when an encrypted database is used in the cloud, because in these scenarios realistic network latencies tend to mask the CPU time of expensive operations.

The Proposed architecture supports adaptive data confidentiality in cloud database environments without requiring any intermediate trusted proxy. Adaptive encryption mechanisms have two main benefits: they guarantee at runtime the maximum level of data confidentiality for any SQL workload, and they simplify database configuration at design time. If the workload is characterized by many similar operations, the present alternative is to accept this cost when data confidentiality is more important than performance. From the research point of view, it would be also inteesting to evaluate the proposed or alternative architectures under different threat model hypotheses.

References

1. H.-L. Truong and S. Dustdar, "Composable cost estimation and monitoring for computational applications in cloud computing environments," Procedia Computer Science vol. 1, no. 1, pp. 2175 –2184, 2010, iCCS 2010.
2. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41,no. 1, pp. 23–50, 2011.
3. L. Ferretti, F. Pierazzi, M. Colajanni, and M. Marchetti, "Security and confidentiality solutions for public cloud database services,"in Proc. Seventh Int'l Conf. Emerging Security Information, Systems and Technologies, Aug. 2013.
4. T. Mather, S. Kumaraswamy, and S. Latif," Cloud security and privacy: an enterprise perspective on risks and compliance". O'Reilly Media, Incorporated, 2009.
5. O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications. Cambridge university press, 2004.
6. J. Daemen and V. Rijmen, The design of Rijndael: AES – the advanced encryption standard. Springer, 2002.
7. A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making snapshot isolation serializable," ACM Transactions on Database Systems, vol. 30, no. 2, June 2005, pp. 492–528.
8. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in Proc. of the 23rd ACM Symposium on Operating Systems Principles, October 2011, pp. 85–100.
10. Amazon RDS Pricing, "Amazon Relational DatabasePricing,"http://aws.amazon.com/rds/pricing, Mar. 2014
9. B. Moeller (May 20, 2004), Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures.
10. Amazon RDS Pricing, "Amazon Relational DatabasePricing,"http://aws.amazon.com/rds/pricing, Mar. 2014.
11. B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in Proc. Fifth USENIX Conf. Operating Systems Design and Implementation, Dec. 2002.
12. TPC, "Tpc-c on-line transaction processing benchmark." http://www.tpc.org/tpcc, Mar. 2014.
13. Amazon, "Amazon Web Services Blog," http://aws.typepad. com/aws/price-reduction/, Mar. 2014.