

EFFICIENCY OF DIGITAL SIGNATURE IN DATABASE AS A SERVICE

KIMBO PRANESH SONGA



SUBMITTED AS PART OF THE REQUIREMENTS FOR THE DEGREE
OF MSc IN CLOUD COMPUTING
AT THE SCHOOL OF COMPUTING,
NATIONAL COLLEGE OF IRELAND
DUBLIN, IRELAND.

August 2014

Supervisor Dr. Horacio gonzalez-velez

Abstract

The aim of the thesis is to implement the digital signature on Database As a Service to ensure efficient data security. The implementation and validation of digital signature raises several challenges such as the selection of suitable algorithm for signing, hashing and storing the digital signature key pair and key exchange mechanism. We generate a key pair for every individual user and store the pair in centralized repository.

We conduct a complete evaluation on efficiency of digital signature with respect to many aspects comparing with other encryption techniques. During the evaluation we propose a banking system as a case study for the empirical validation of our approach. The main challenging part of the thesis is maintaining data integrity, and confidentiality even though its a time consuming process. The interesting feature of digital signature application is, it doesnt allow cloud database administrator to view the sensitive field data of the database.

Keywords:

Digital signature creation, DBAAS, FHE, Encryption, Decryption, Digital signature verification, Public key, Private key, RSA, SHA-1, CryptDB, Entity framework, LINQ2

Acknowledgements

I would like to take this opportunity to express my gratitude to everyone who have been helping me and guiding me through out this research work for my MSc Cloud computing. I whole heartedly thank my supervisor, Dr. Adriana Chis for the support she has given for the completion of the thesis. I appreciate her skill in various aspects and guidance in documentation part of the thesis. I sincerely thank Dr. Alina Madalina Popescu for her support and assistance.

I take this special privilege and honour to thank Dr. Horacio Gonzalez-Vlez for giving me this golden opportunity to pursue my MSc in Cloud computing in National College of Ireland.

I would like to thank my family specially my sister Ratna Joseph for their support through out my life.

Declaration

I confirm that the work contained in this MSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed

Date

Kimbo pranesh songa

Contents

Abstract	ii
Acknowledgements	iii
Declaration	iv
1 Introduction	1
2 Background	4
2.1 Database As a Service	4
2.2 Emergence of encryption techniques	5
2.2.1 Different types of encryption techniques	6
2.2.2 Evaluating encryption techniques based on data privacy and performance	7
2.2.3 Performance evaluation:	7
2.3 Efficiency of digital signature based on data confidentiality and data privacy	8
2.3.1 Data integrity	8
2.3.2 Anti deniability	8
2.3.3 Data protection and privacy	9
2.4 Working of digital Signature compared with different Encryption techniques in database as a service	9
2.4.1 Types of public key cryptography for digital signatures	11
3 Design	12
3.1 Communication between webapplication and AzureDataBase	15
3.2 AzureDBApplication implementation modules	16
3.2.1 Generation of digital signature	18
3.2.2 Verification of digital signature	19
3.2.3 Sequence diagram for digital signature creation	20

3.2.4	Sequence diagram for digital signature verification	21
3.3	Database Design and schema	21
3.3.1	Database schema	23
3.4	Decisions and Reasoning	23
4	Implementation	25
4.1	System specifications:	25
4.2	Digital signature implementation techniques	26
4.3	Class diagram for digital signature, encryption, decryption, verification	27
4.4	Experimental setup	27
4.4.1	Microsoft ADO.NET entity framework 5.0 from local machine to connect to SQL server 2012 database in azure	28
4.4.2	Experimental setup to migrate the web application from local to Virtual machine on windows azure	28
4.4.3	Deploying on existing ASP.NET AzureDBApplication to run on windows Azure website	29
4.5	Pseudo code implementation of digital signature	29
4.6	Software development life cycle:	32
4.7	Software testing methodology	33
5	Evaluation	34
5.1	Evaluation in azure instance	37
5.2	Experimental Analysis after deploying on azure website	39
6	Conclusions	45
6.1	Conclusion	45
6.2	Future Work	46
	Bibliography	47
A	How to create an Entity Data Model	49
B	How to Migrate the AzureDBApplication from local to Azure in- stance	53
C	How to Deploy an existing ASP.NET AzureDBApplication to run on Windows Azure website	55

List of Tables

2.1	Types of Encryption Techniques	6
5.1	Time taken for only digital signing in local machine	35
5.2	Time taken to only decrypt the fields randomly in local machine	36
5.3	Time taken to only verify in local machine	36
5.4	Time taken to only sign in azure instance	37
5.5	Time taken to only verify in azure instance	38
5.6	Time taken to Decrypt in Azure instance	38
5.7	Time taken to sign in azure site	39
5.8	Time taken to verify in azure site	40
5.9	Time taken to encrypt in azure site	40
5.10	Time taken to decrypt in azure site	41

List of Figures

2.1	Level of security Vs Database service providers capability graph	10
3.1	Design of digital signature implementation	13
3.2	Communication between application and Azure	15
3.3	workflow of digital signature	17
3.4	creation of digital signature	18
3.5	Verification of digital signature	19
3.6	sequence diagram	20
3.7	sequence diagram	21
3.8	Database schema	23
4.1	class diagram	27
4.2	Encrypted Data in Azure	30
4.3	Verification of Digital signed data	31
4.4	Decryption of Digital signed data	32
5.1	Exeuction time of sign Vs verification	41
5.2	Execution time of encryption Vs decryption	42
5.3	Execution time of sign Vs verification	43
5.4	Execution time of encrypt Vs decrypt	44
A.1	ADO.NET Entity Model	49
A.2	Entity Data Model	50
A.3	Database Connection	51
A.4	Version selection	51
A.5	Database Settings	52
B.1	Connecting to remote desktop of azure	54
C.1	Configuration setting to deploy the web application to azure site	56

Listings

4.1 Pseudo code of hashing and signing	29
--	----

Chapter 1

Introduction

Cloud computing is the advanced technology in this present era of networking and internet technologies. Buyya, Yeo, Venugopal, Broberg & Brandic (2009)) say that there is certainly seen vision that cloud computing will turn into the fifth utility in the world. Dubey & Wagle (2007) say that cloud is a platform where providers can deploy different services that will provide users flexibility to use services without installing the particular software on the individual systems. One of those services is DSaaS. In DSaaS clients are permitted to create, store and retrieve any kind of information from any corner of the world as long as they have internet connection. Since people can access data from anywhere in the world, there are major issues with data integrity and data privacy. The increase in access to the database storage has also increased frequency of the threats considerably increased. In the present situation threats are more for data assets. Therefore a database must be provided with more security features, when compared to business operations (online banking) i.e. to improve data privacy. There are many techniques to address data security issues, like CryptDB and Full homomorphic encryption, which are used to secure the data from the untrusted entities.

Kadhem, Amagasa & Kitagawa (2009, p.165) define data privacy as a "right to secure sensitive personal data in digital form to protect it against fraud, identity theft or unauthorised use". In many enterprises basic sensitive business information in databases is an obvious focus for attack. Subsequently, guaranteeing the confidentiality, security and integrity of information is a real issue for the security of database framework. The data transferred on a network must have the attributes of anti-deniability and integrity. In cloud computing there are issues like security of information, files management, issues in network and host security. The main functionalities of digital signature are introduced to protect the security of the data, non-repudiation and data integrity.

The technique of digital signature provides security to the entire application. Here the focus is not to lessen the requirement for different innovations, for example, encryption, verification, access control, firewalls and interruption discovery. Digital signature fit into the security fundamentals of an application. Encryption based on security components does not completely address some aspects like data integrity, anti-deniability and data privacy. These are purely addressed by using digital signature so occurring the fraud cannot be stopped but it can stop succeeding fault transaction by giving applications the capability to identify fault transactions.

The motivation of the thesis is to reduce the attacks that are occurring in enterprises, industries and organizations on sensitive business data. The aim is to secure sensitive information by applying digital signature on fields of database tables which are frequent exposed to threats. Digital signature proves the identity of the user and data can be secured from tampering. Therefore digital signature is more efficient when compared with other encryption techniques. In this thesis we propose a solution to increase the security of databases, namely we introduce the use of digital signature at the level of databases to ensure data security, and we implement this solution as a DSaaS. The main challenge of AzureDBApplication is securing the data from azure database administrator, in general digital signature verification process the original message, public key, and hash data must be sent to the receiver site to apply hash on the plain message and then to compare the hashes to prove the identity and to assure data integrity, and anti-deniability. AzureDBApplication is implemented in such way that the azure database administrator cannot view the plain message at the time of digital signature verification because the sensitive field is first encrypted with RSA algorithm and then sent to the azure database storage along with the digital signature. The other challenge is current encryption techniques may be efficient pertaining to time but cannot ensure the confidentiality and integrity. As digital signature is a block of byte array the main challenge is storing the signature in a separate database table to verify and decrypt the data as per the user requirement. This storage of digital signature has impact or affect on database storage space to store in database and time required to process the signature.

The main contribution of the thesis is applying digital signature on DSaaS on SQL queries to provide efficient data security. As mentioned earlier digital signature is applied not on plain field, instead we apply digital signature to encrypted field data (see Design chapter for details). Consequently the cloud provider cannot view sensitive fields data, thereby confidentiality of the data is maintained throughout the application. This approach may be time consuming when compared with encryption, because the digital signature is applied on an encrypted field, but the availability, non-repudiation, data

integrity, confidentiality and authentication are maintained. Furthermore, we conduct empirical evaluation to evaluate the performance of encryption, decryption, signing, and verification under three different environments. We implement our approach by applying digital signature on Microsoft Azure DB, in particular we propose digital signature on AzureDBApplication. In this thesis we discuss the encryption and digital signature considering different parameters and security issues involved.

Our thesis is organized as follows.

- Chapter 2 : In particular, we discuss Database as a Service (DBaaS) and review different encryption techniques.
- Chapter 3: We describe the design and specifications of AzureDBApplication.
- Chapter 4 : Presents the implementation of AzureDBApplication on digital signature.
- Chapter 5 : Shows the evaluation of digital signature and encryption on different environments.
- Chapter 6 : We conclude our research and present future work.

Chapter 2

Background

2.1 Database As a Service

Hacigumus, Iyer & Mehrotra (2002) say in traditional approach for fulfillment of different data processing requests business organization focus more on installing and managing database. To purchase the necessary hardware and deploy database products over a connected network and to run the system by hosting a professional is really very expensive to manage. This process led to the emergence of Database As A Service. DBAAS reduces the financial burden on an organization so that it can focus on the business logic of the project. Service providers take care of database backup, administration and also restoration. This approach helps to eradicate the issue of installing, upgrading the software by actualizing the above methodology will utilize the instant database framework kept up by the administrator for its database needs. Popa, Redfield, Zeldovich & Balakrishnan (2012) say that DBAAS shifts the burden of scaling, provisioning, performance tuning, backup, privacy and access control from the database user to the service provider at very low cost. The main reason that made Database As A Service more attractive is due to the economic scaling of hardware and cost of resources is affordable by users. The main thing that attracted users towards Database As A Service is pay for the service that is utilized. Lehner & Sattler (2010) say PNUTS, HBase, SimpleDB, Google BigTable are current cloud DBS allows the user to submit queries to DBs with generic schemas. Basically, these systems consist of simple container l with put/get semantics for data blocks of unknown structure. DBAAS will be successful only if security and confidentiality is ensured by outsourced database. In next section of the paper we will explain the emergence of encryption techniques in a database.

2.2 Emergence of encryption techniques

Kadhem et al. (2009, p.163) argue that in private and government sectors the need for information security has increased tremendously because of the sensitive data. They identified a high rate of threat for the sensitive data in specific as mentioned " In May 2008, researchers at security vendors uncovered a server containing the sensitive email and web-based data of thousands of people, including healthcare information, credit card numbers and business personnel documents and other sensitive data", This shows that sensitive data like healthcare information, credit card numbers and personal business documents have lost, this proves that traditional techniques need to be improved still further. Kadhem et al. (2009) state that preexisting traditional security measures like access control and authentication cannot prevent intrusion and unauthorized access to occur for a database. This led to the development of the encryption techniques, which will secure a database to certain extent even if attacker attacks it. Curino, Jones, Popa, Malviya, Wu, Madden, Balakrishnan & Zeldovich (2011) say that there are two types of threats are addressed; one is preventing Database administrator (DBA) from seeing the private data. The other is cryptDB system that provides security-using authentication, which improve data privacy from the attackers. Gahi, Guennoun & El-Khatib (2011) propose that the client could encrypt the data and then deploy on a cloud providers site, so that it will be secured from the cloud provider. So that only authorized client can decrypt the data whenever required on processing the query to the provider site. Kadhem et al. (2009) propose a new system named three-tier environment depending upon data of the owner. Responsibilities of encryption are divided among client data, trusted third party data, and server data. They propose system that adds a encryption/decryption layers while data is sent and retrieved from to the database. Mani, Shah & Gunda (2013) did not introduce any layered type architecture for data encryption. Kadhem et al. (2009) state to implement encrypted storage to satisfy data privacy and data integrity. While deploying a database on the cloud the major obstacle is data privacy, to resolve this issue there are many encryption techniques that are briefly explained in the later sections of this paper. Mani et al. (2013) discuss Data integrity; as assuring authorized person stored data is unchanged or not modified on the service provider site . This also includes protecting the database from the other unauthorized users from thefts. Kadhem et al. (2009) say a clients data that is stored in the database of the service provider site must be encrypted this eliminates the issue of data privacy to a larger extent. Using CryptDB and other encryption techniques data privacy is increased, the next section discuss different type of encryption techniques and critically analyze the techniques based on few security components. The below (Fig. 2.1) explains the two different types of encryption techniques.

Table 2.1: Types of Encryption Techniques

Encryption	Methodology	Algorithm	Data Integrity
CryptDB	Onion Layered Encryption.	AES and Blowfish	Medium
Full Homomorphic	Circuits for Encryption	Evaluative algorithm	Medium

2.2.1 Different types of encryption techniques

Curino et al. (2011) propose a design named cryptDB (which prevents the administrator to see the user data) to ensure the privacy of the stored data by encrypting all the tuples, example employees salaries, which is sensitive and must be secured. Encrypting the entire database ensures sensitive data is secured from outside threats. The main idea of this technique is to encrypt each value of each row into onion. Onion is like wrapping a value into a number of layers (encryption schema layers). Each value in the row is independently dressed for strong encryption with every integer value stored three times. The integer is stored two times as an onion to allow database queries and once for homomorphic integers. Popa et al. (2012) give their support to the onion encryption technique, adding additional extra functionality to each onion layer. Curino et al. (2011) say CryptDB provides different encryption levels for data, depending upon the sensitivity of the data, and also based upon the database queries that user executes. Mani et al. (2013) say Full homomorphic encryption algorithm to encrypt and decrypt the user requested data. The general ground level of encryption techniques is the one, where queries are evaluated on the encrypted data by the server at provider site and sent back to the client side for further decryption. But as explained above in Curino et al. (2011) say onion based encryption level of evaluation is a bit high while evaluating a particular query.

A fully homomorphic encryption technique (FHE) consists of an evaluating algorithm that will encrypt and decrypt the data which is part of encryption schema. The evaluating algorithm of fully homomorphic encryption technique uses a circuit for input queries and then produces the output in cipher text, which is the encrypted format of data that is not understood or can't be decrypted by an unauthorized person. Hence FHE can process any query and evaluate any function. The major disadvantage of FHE technique is increase in computing time of user query. Gahi et al. (2011) say that inspite of the fact that this type of processing may increase the amount of computing time the benefits associated with it are worth the processing overheads.

When a client processes a query on the encrypted data, the query enters the circuit in the algorithm then the query is evaluated on encrypted data and the requested data will be retrieved as per the given input query. This might be a difficult process for querying

because a query would have to enter the particular circuit of the FHE, which is very complex and computing time also increases. Basically in database indexing helps to search the user data fast. But when we use FHE technique in database indexing does not work as data is encrypted. This will reduce the performance and efficiency of the system, which can be resolved by digital signature that is being explained in the next sections of the paper. Similar to Mani et al. (2013) say in relation to FHE outline that data must enter into the decryption circuit to remove the inner layer of encrypted data using a private key. Gahi et al. (2011) say that a medical application was built on non-secure homomorphic schema because its not possible to use test cases for a fully homomorphic cryptosystem which can be considered as one of the major disadvantages of encryption. The next sections of the paper explain encryption techniques based on different parameters like data privacy and data performance.

2.2.2 Evaluating encryption techniques based on data privacy and performance

Kadhem et al. (2009) say that there are two major privacy issues. First for the trusted servers, where the service provider must assure the owner that data on the cloud is protected against data thefts. The second issue is owner must be assured that data stored in the service provider is safe i.e. even service providers cannot be trusted. The main issue is that sensitive data must be protected because there are many disadvantages with the encryption techniques used for particular SQL queries. Gahi et al. (2011) outline that if an unauthorized user try to decrypt the encrypted data, the original data is not reveled because encryption is a key based access control system. Basically encryption technique is applied to only to the columns of database. Further section explains about the process of encryption in detail.

2.2.3 Performance evaluation:

Gahi et al. (2011) implement a system known as secure database system, the main functionality of system is a client sends an encrypted query to the database cloud provider and the provider evaluates the request and sends the encrypted data from the database. This will affect the scalability, performance and also increase complexity of system. Popa et al. (2012) implement different types of measures to improve security and performance of the database. Gahi et al. (2011) has not implemented any measures to optimize the performance of database. Popa et al. (2012) discuss to optimize the performance of the database by encrypting sensitive data fields and remaining other

fields will be in plain text. This will to some extent optimize the database performance and reduce the complexity.

2.3 Efficiency of digital signature based on data confidentiality and data privacy

2.3.1 Data integrity

Ngai & Wat (2002) say if there is any slight modification in the message or text the abstract will change a lot for hash functions peculiarity, so this avoid the message being distorted and assures data integrity because of its cryptographic nature. Aki (1983) say when sender can send the message using private key. Furthermore validation is additionally feasible fact that the receiver can confirm that the message has not been altered while it was on the network. Rewagad & Pawar (2013) propose architecture to encrypt and decrypt the user data file using AES encryption algorithm which also includes level of authentication. This is implemented to provide trusted computing environment in order to avoid data modification at the server end. So to avoid the data modification digital signature is best solution. Because digital signature will not stop the modification of data but it proves that the data has been altered. This mechanism includes Die Hellman algorithm to generate keys for key exchange step. Meijer & Aki (1982) say the complication in key exchange is "man in middle attack". In private key cryptography both encryption and decryption is done using same key, so when the session keys is sent over the network attacker can intercept the keys and alter the message. Which cannot be predicted by the receiver or authorized user.

2.3.2 Anti deniability

Ngai & Wat (2002) say that Anti-deniability can be solved by using public key cryptography algorithm; the sender cannot deny that he has sent the message for he has the private key. This avoid receivers forging message that is claiming to be from the sender. Anti-deniability can be achieved perfectly only by using digital signature. Digital signature makes encryption and decryption are irretrievable decrypting the data is done using the pubic key of the sender as it is shared between all the authorized users. Zhang (2010) say in order to show the identity of the sender digital certificates are generated by the receiver using his private key this methodology of certificates proves anti deniability and proves repudiation.

2.3.3 Data protection and privacy

Somani, Lakhani & Mundra (2010) mention the concept of encrypting the data before it reaches to the receiver's end this is implemented using RSA algorithm. This technique solves the dual problem of authentication and security this implies data privacy is assured. The strength of their work is the framework proposed to address security and privacy issue. Aki (1983) explain it is clear and necessary that sensitive information must to verified and authenticated between sender and receiver in a secure manner. Certifying the content of the message is validated using the concept named verification and authorization by using methods like `verifyHash()` in the concept of digital signature. So every message is appended with a digital signature this assures data privacy and data validation for a larger extent.

2.4 Working of digital Signature compared with different Encryption techniques in database as a service

Sklavos, Kitsos, Papadomanolakis & Koufopavlou (2002) say digital signature is a string of binary data or a byte of characters. Identity of the signatory and integrity of the data is computed using a set of parameters in the process of implementing digital signature, signature generation and verification is done using private and public key pair. A signatory sign the data using private key and verification is done using public key which is shared by all the authorized users of the application but always the private key is kept in secret. Each user of the application has a unique private and public key pair. Aki (1983) say a database signature function integrates a digital signature on the data and then stores it in the database. A database uses a command to execute a stored procedure on which digitally signs and then stores back in the database. To verify a digital signature query, commands are used. These measures will make data more confident when compared with different encryption techniques. Mani et al. (2013) explains a graph comparing different encryption techniques pros and cons of each other taking security in the y-axis and different database providers capability in x-axis .

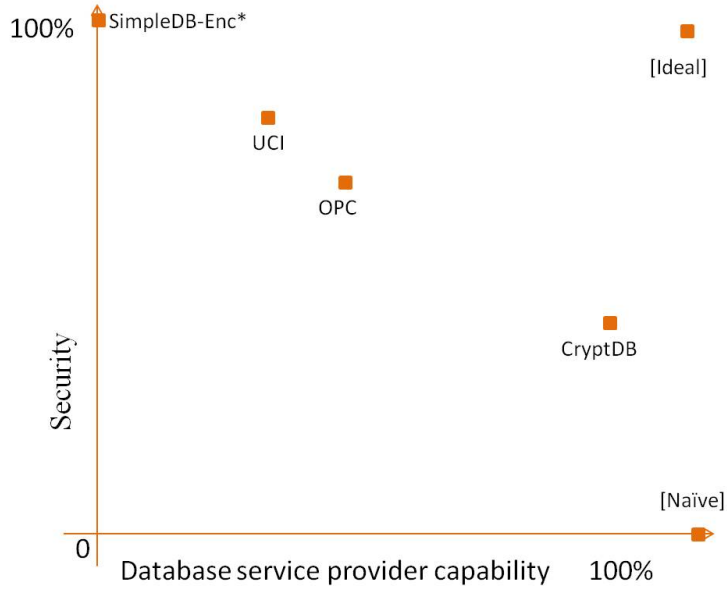


Figure 2.1: Level of security Vs Database service providers capability graph

The (Fig. 2.1) is explained below comparing the efficiency of encryption techniques with security on y-axis and database providers on x-axis.

1. Simple DB provides security very efficiently, but service provider does not provide any type of query processing.
2. Naïve which is one of the encryption technique and does not provide security stores the record in the plain text, but service provider provides all the query processing.
3. The main idea behind cryptDB is adjustable security, but this doesn't provide effective query processing.
4. Efficient security with no query processing.

By this graph we can prove that encryption techniques lack in few aspects that may be security or query processing. Digital signature has solved these issues effectively. Narasimha & Tsudik (2005) say an alternative method or traditional approach is to use digital signature at granularity to the individual tuples or fields in the database table. The database owner should sign each tuple before storing it in the outsourced database in the server site. So server will store each tuple along with the signed tuple. When a query is processed the server will send the matching tuples along with the signature to prove the authenticity and integrity of the system.

2.4.1 Types of public key cryptography for digital signatures

huang Wu (2010) propose traditional approach firstly authenticating a user with his public key along with a certification is known public key cryptography were user identity is known by the public key which is signed by the certification authority. Storage, certificate verification, and distribution are the problems that must be taken into consideration. Identity-Based Public key Cryptography (IB-PKC) is other type of cryptography were the public key authentication of the uses are entirely different from the traditional approach. There is no need for the authentication of users public key with a certificate which is a third style of cryptography known as certificate less public key cryptography (CL-PKC). The main disadvantage of this style is that in case of denial of service attack the attacker as well as authorized user cannot decrypt the message this is known as Denial of Decryption Finally Self-Generated-Certificate Public Key Cryptography (SGC-PKC) .

Chapter 3

Design

The design shows the architecture of AzureDBApplication, a Asp.Net web application that maintains any type of data pertaining to any type organization or Industry like Hospital database, Banking system database. In the thesis the AzureDBApplication is banking database-backed application consisting of azure SQL server and separate client application. Banking system must be secured from threats as it has sensitive data of accounts, customers which must be authenticated, authorized. While transactions are made on network data integrity, anti-deniability must be maintained throughout the transactions. In order to secure the data from these threats digital signature is applied to sensitive fields of database tables. AzureDBApplication provides high level of confidentiality and integrity throughout the transactions.

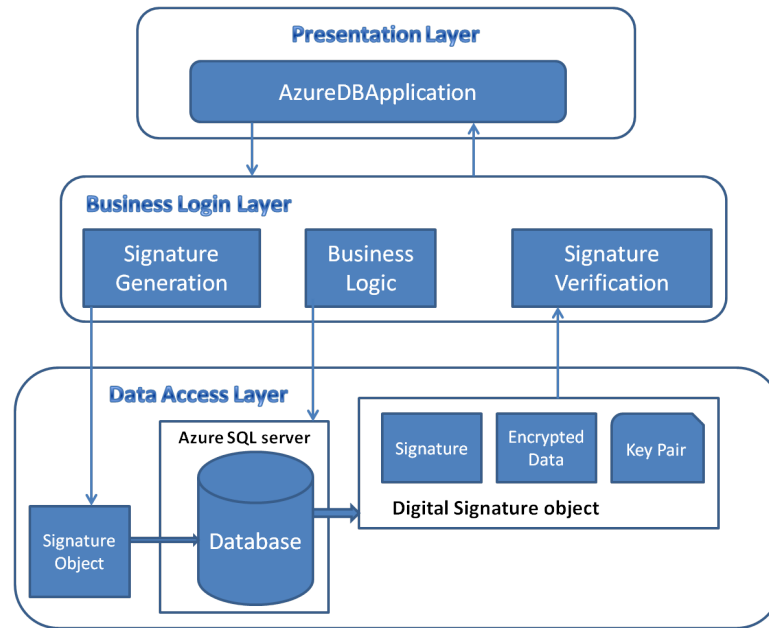


Figure 3.1: Design of digital signature implementation

In above (Fig. 3.1) architecture implementon is designed as layered architecture, which comprise of three layers.

- Presentation layer: AzureDBApplication may reside anywhere at three different enviroments local machine, azure vm instance and azure website. In both the scenarios user will have user name and password to access the database and application. The web application provides all database operations (create, retrieve, update, delete) to users and all the operations are database transactions which access database tables.
- Bussiness layer: The business logic layer provides interaction or communication between presentation layer and data layer. Creation of digital signature objects for the signed data are handled by the business layer and persisted into database for every individual user for security reasons. The digital signature object comprises signature, encrypted data and primary key of table being signed. The web application queries to store signed data and encrypted data of message into the database. In response to the query commands, SQL server retrieves result sets from the database tables. This system includes business logic, as part of the digital signature module; the business logic is executed based on digital signature of the user and identifies records in the database tables. For example, a query command may request all data signed by particular user business logic identifies all data from database signed by that particular user. Signature verification is

also part of business logic, authenticates the data, the users key pair used to verify the sign. For the above example, after the database system identifies all records signed by that particular user, signature verification verifies the authenticity of the data as well as the authenticity of particular user. Only the data is returned as a response to the query. The signature verification functions are transparent to the application users. However, if the verification procedure fails, then the user is informed that either the data is corrupted or the users key pair is corrupted. Business logic interprets the command regarding requests based on digital signatures. Based on records identified, the key pair for the corresponding signatories of the data is extracted as signature object for the records identified is also extracted. The signature verification utilizes the key pair and signature object to generate a response to the query.

- **Data Access Layer:** The database will reside in cloud as a service (DBAAS), which provides authentication and authorization to database application users. The individual user will have digital signatures based on their queries and will be identified, verified using the key pair stored in the centralized repository. Actually the digital signature and encrypted data stored into database table to provide digital signature functionality during verification of digital signature.

The database engine executes the query command received from the business logic and returns the results set to users. As discussed above, business logic includes any logic in a query that includes digital signature related information as a parameter (e.g., retrieve all data signed by signatories). If no records in the database are identified in response to the query, then a query response, with no records are provided. Alternatively if, at least one record satisfies the query criteria, then the matched record(s) are extracted, including the signature object, from the database table. All the query requests that update or insert data into significant or sensitive data columns have to be decided. All queries affect the values of sensitive data fields need to be used in the digital signature.

3.1 Communication between webapplication and Azure-DataBase

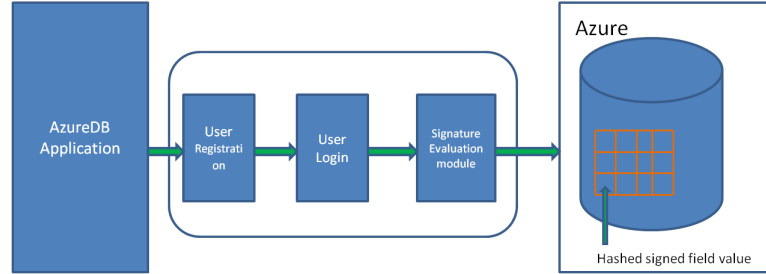


Figure 3.2: Communication between application and Azure

The AzureDBApplication is designed to digital sign the banking database system. Actually the system is designed to digital sign any database, but now the application is using the banking system. The main reason behind choosing the banking system is banking database have lots of confidential sensitive bussiness data that cant be shared or understandable to any user even the azure database provider. The second reason is , if the banking data need to migrate to database in azure cloud in this case the security , integrity and anti-denialability has to be maintained. The banking database has AccountTable, Loan, Branch, Customer tables as user data and DigitalField table to store the signature and encrypted data. The user data tables are separated from digital signed table. The AzureDBApplication has separate screens or web pages for each table, so that the user can perform SQL operations on data. Every table mapped with Data model class to update the user interface changes with database and to get the data from database.

First the user has to register into AzureDBApplication to access the application and login into system using crdentials like Username and password. Login into the application enables the user to access database tables, insert into tables, digital sign the fields of database tables, decrypt data, verify the digital signature. The AzureDBApplication allows user to take the required tablename , columnname, and no of records to sign sensitive fields as per the requirement. To communicate with the database entity framework and LINQ queries are used because with entity framework the operations are automated and mapped with database model classes, LINQ queries are efficient and fast in retrieving the database.

In digital signature design SHA1 algorithm is used to hash message and RSA (Rivest, Shamir and Adlemen) algorithm is used for digital signing. RSA algorithm is chosen as it is the most commonly used algorithm for encryption and digital signing. The

key length used for AzureDBApplication is 1024 bit long so it has got high level of security and faster verification of signature. To achieve all the specified functionality the RSA and SHA1 algorithms are provided by .Net framework namespace (System.Security.Cryptography.RSACryptoServiceProvider) has the helper classes for the specified algorithm functionality. The public and private keys are created using the helper class RSACryptoServiceProvider from .Net framework and after creating, the key pair is serialized in X.509 format using AsnKeyBuilder and AsnKeyParser classes. This X.509 format based key pair (public, private) files are stored in specified centralized repository. From Digital Evaluation web page when user clicks on DigitalSign button the specified number of records in particular table for the given field will be hashed, signed and inserted into DigitalField table present in the azure storage.

3.2 AzureDBApplication implementation modules

- Digital Signature module

Digital signature module comprises generation of RSA based public and private key pair, loading the key pair in X.509 format based files, encrypt and decrypt the data, hash and sign data and finally verifying hash of the data. The class DigitalKeys of AzureDBApplication implements the specified digital signature functionality using System.Security.Cryptography C#.Net namespace helper classes. The C# System.Security.Cryptography namespace implements digital signing, encryption, decryption, verification and key exchange functionality for RSA algorithms. The complete functionality of DigitalKeys class can be categorized into three functions namely encryption with RSA, Hashing with SHA-1, and finally signed with RSA shown in (Fig. 3.3)

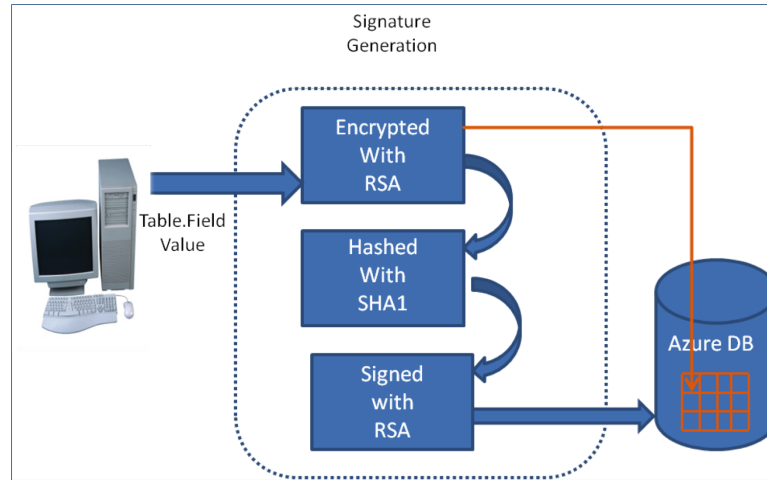


Figure 3.3: workflow of digital signature

1. Generate public and private keys of type RSAPParameters as per the RSA algorithm. It generates 1024 bit size key pair by default CspParameter.KeyNumber = AT_KEYEXCHANGE.
2. Load the private and public key pair individually into X.509 format based files. The file name is named as user name logged into the application at that instance.
3. The class has methods implemented to encrypt, decrypt, hash and sign, and verify hash functionality. These methods can be invoked as per the requirement.

- User modules

AzureDBApplication is web based application, the user interactions are provided by web pages for individual functionality. The web application has Login page, registration page, Account page, Loan page, Branch page, Customer page. These web pages are provided to the user to insert, update and delete data records from the database tables.

- Database Module

The AzureDBApplication communicates with database using entity framework as steps involved are already specified in 4.6(Experimental Setup). Each database table has to be mapped with separate domain specific data model objects. These mapped objects are generated automatically by entity framework and used to interact with Database as CRUD operations. EFModel.edmx and EF-Model.Designer.cs are automatically generated by performing the steps specified

in the session in 4.6 (Experimental Setup). The developer can simply update the model from database and vice versa using entity framework in VS 2013.

3.2.1 Generation of digital signature

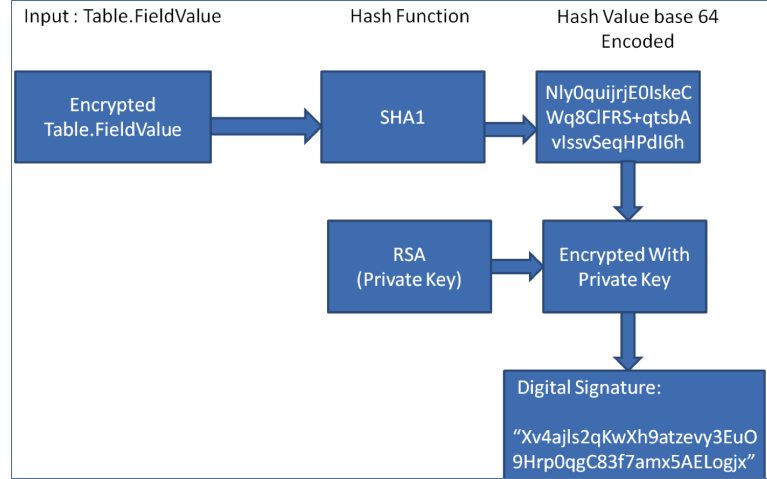


Figure 3.4: creation of digital signature

The (Fig. 3.4) explains in detail creation of digital signature. Firstly the particular sensitive field of a table is encrypted using RSA algorithm and sent as input. Next hash value for encrypted field is computed using SHA-1 algorithm. Then the hashed data is encrypted with the private key of the user to generate signature. Finally this digital signature, encrypted field values are sent to azure database table named DigitalFieldTable. The process of sending will be explained in the next section in detail.

3.2.2 Verification of digital signature

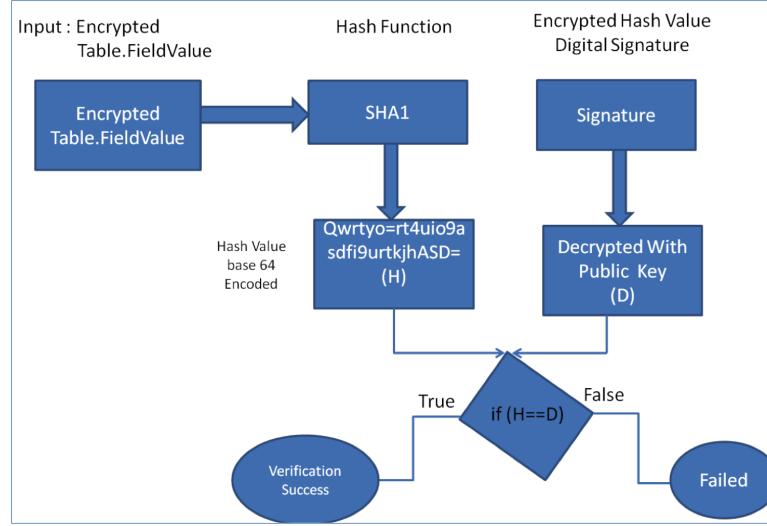


Figure 3.5: Verification of digital signature

The (Fig. 3.5) explains the verification of digital signature. First as encrypted field and digital signature values are retrieved from azure database Digitalfield table. Digital signature is decrypted with the public key using RSA algorithm to get a hashed field data. Next SHA-1 function is applied to encrypted field data to compute hash for that data. If both the hashed field data and decrypted signature are matching then verification of digital signature is successful. Data integrity is not compromised and identity of the user is authenticated.

3.2.3 Sequence diagram for digital signature creation

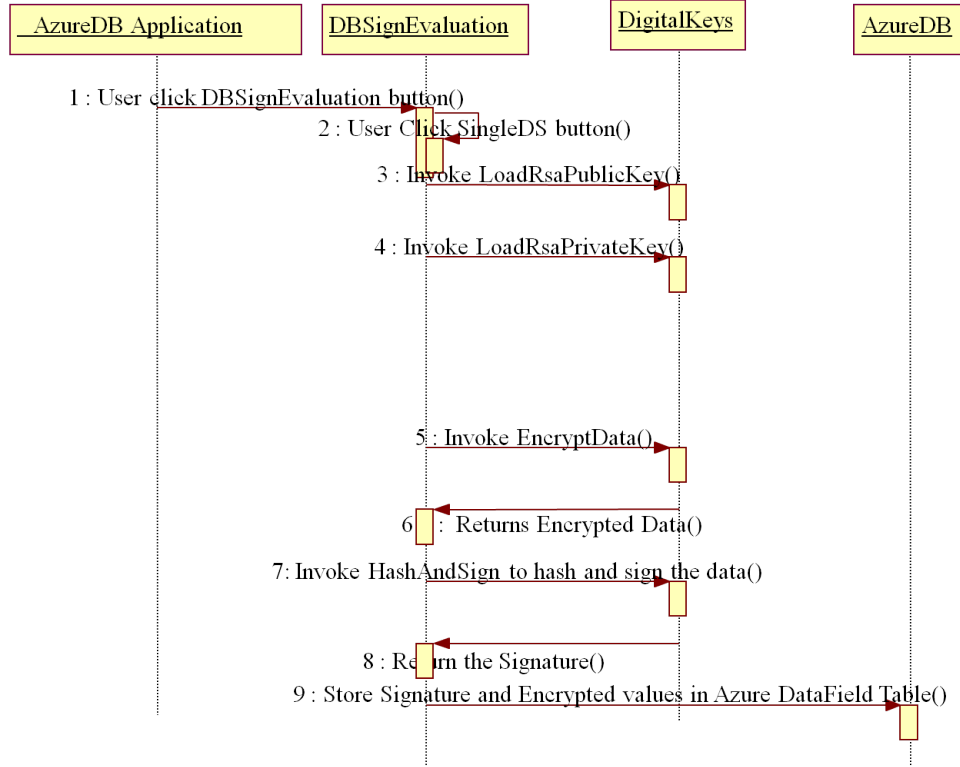


Figure 3.6: sequence diagram

In the digital signature creation sequence diagram (Fig. 3.6), first the user has to login into the AzureDBApplication, Click DBSignEvaluation button on main page of the application. Once the DB sign evaluation screen is popped up, user has to enter table name, field name and no of records to be digital signed and click on SingleDS button. After this invoke LoadRsaPublickey and LoadRsaPrivatekey, the methods load public and private keys into RSACryptoServiceProvider class. Next invoke EncryptData() method to encrypt the field value and returns encrypted data so that data will not be available in plain text in azure database. So the digital signature is created for encrypted field value by invoking the HashAndSign() method , which applies SHA1 Hash function to encrypted field and encrypt the hashed value with RSA private key, so that the filed value is digital signed. Once the digital signing processing completes the encrypted data and signature values are stored in Azure database DataField table.

3.2.4 Sequence diagram for digital signature verification

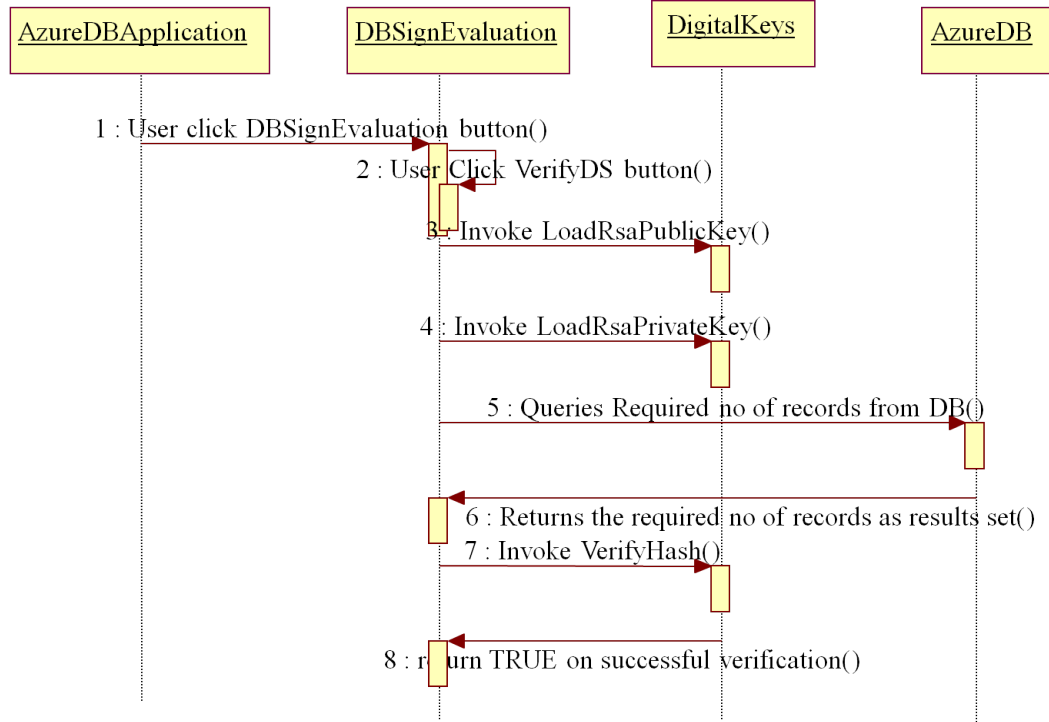


Figure 3.7: sequence diagram

In Digital signature verification sequence diagram (Fig. 3.7) , user provides table name, field name and no of records and clicks on button VerifyDS to initiate the signature verification. To the VerifyHash method, the input parameters are public key, signature and encrypted field value. Again SHA1 hash value is computed on encrypted field value and signature value is decrypted using public key. The hash computed on encrypted field value is compared with decrypted signature, if both the values match then the digital signature verification is successful by returning true, otherwise failed.

3.3 Database Design and schema

The azure SQL server 2012 database is used for the implementation of the banking management system. The banking management system database has seven tables namely User table, AccountTable table, Branch table, Customer table, Loan table, Digital-FieldTable table and DigitalDatatable table.

1. User (Username, Password)
 2. AccountTable (Account_No, Account_name, Branch_name, CustomerID, First_name, Second_name, City, Street)
 3. Branch (Branch_ ID, Branch_ name, Branch_ city)
 4. CustomerTable (CustomerID, CustomerName, CustomerAge, CustomerAddress, Gender, DOB, BranchID, AccountType, AccountNO)
 5. Loan (Loan_id, Branch_name, Loan_Amount)
 6. DigitalFieldTable (SNO, UserName, TableDetails, TablePK, Signature, EncryptedData)
 7. DigitalDataTable (SNO, Username, EncryptedData, Signature, TableDetails)
- Primary key in the User is Username
 - Primary key in the AccountTable is AccountNo .
 - Primary key in the Branch is Branch_ID.
 - Primary key in the Customer is Customer_ID.
 - Primary key in the Loan is Loan_ID.
 - Primary key in the DigitalFieldData is SNO.
 - Primary key in the DigitalDataTable is SNO.

3.3.1 Database schema

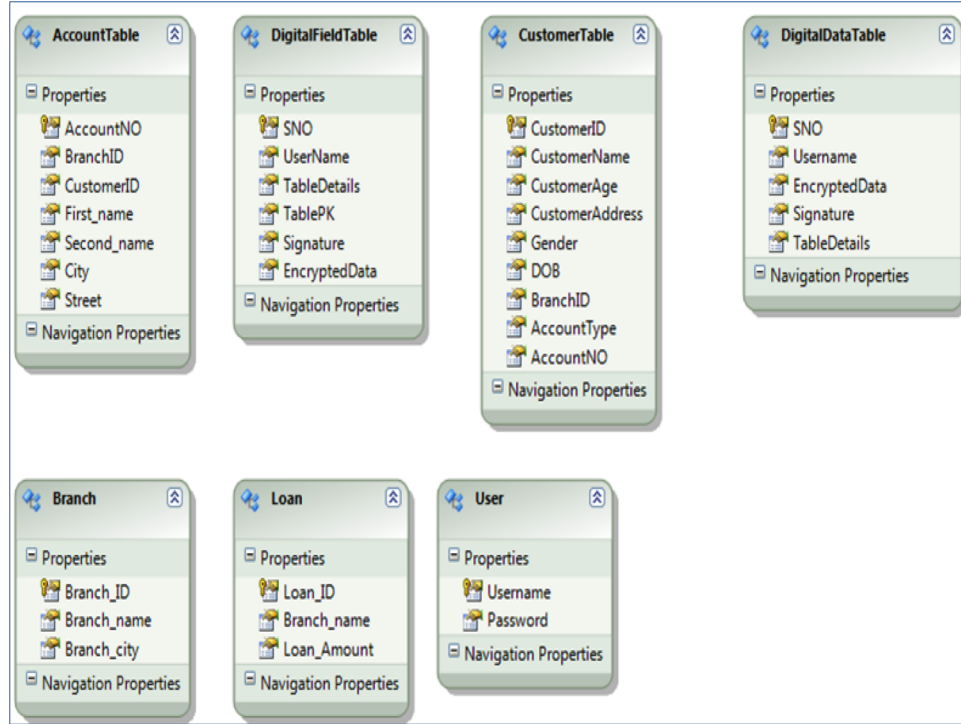


Figure 3.8: Database schema

3.4 Decisions and Reasoning

- The application is based on database and it resides on remote server as a service like DBAAS. Here the constraints and parameters that applies to DBAAS is not considered. The assumption is DBAAS in cloud is just like database in remote server.
- The digital signature is processed and is persisted as object and stored in database, because the users will not be able to see the digital signature has a common public, private key pair.
- Performance evaluation of database connections, network latencies, network traffic are not considered.
- The focus is inserting digital signature into database as persistent signature object and retrieving the signature object while returning the results to get verified by verification logic has to be considered.

- The reason for choosing digital signature is its impossible to imitate or regenerate digital signature and its implementation is not dependent any proprietary OS or platform or language. In this application the assumption is key generation and verification process must happen on local client machine, so that implementation and verification will not fail due to restricted accessing of the remote server or network failure and latency.
- There is possibility to provide role based check on database through the registration process, that means only particular role only can sign particular data.
- In this architecture the digital signature is serialized as object model not like a file and only while processing the object model is used so there is no much disk memory wasted.
- When a signature fails to create or verify, the processing of database transaction must be stopped until the discrepancy is resolved. Using error handling mechanism will ensure that all errors are handled and users are presented with the some error message.

Chapter 4

Implementation

Today as cloud has being advanced to great extent in all aspects like Information technology and government organization. There is a need to migrate database into cloud which is known as Database as a Service (DSaaS). Therefore there will be a tremendous increase of threats this can be solved by implementing digital signature while inserting and retrieving data from the database. Many research papers gave solution to reduce the threats by implementing different encryption techniques using symmetric algorithms. The aim is not to decline encryption techniques but to increase the efficiency of security using digital signature in the DSaaS, which maintains data integrity, and confidentiality. The aim is to insert digital signature in the query and how it is stored and verified in the database and the algorithm used for the implementation of the digital signature this will be explained in this chapter. The main issue is how well the random number of request queries can access the database server without breaking and handling the exceptions incase digital signature is not generated. In this chapter will address these issues to some extent.

This chapter discusses about the implementation details of the AzureDBApplication. The functionality implementation can be divided into three vital parts. One is digital signature module and second module implements user modules includes user interactions, user interfaces, mainly digital signature evaluation methodology and third is the database operations (CRUD) and LINQ queries using entity framework.

4.1 System specifications:

The thesis is implemented based on client server model, where the database application is on the client machine and database is deployed on the cloud.

The application comprises DB application interface, data model (like digital signature object) and database transactions in entire application all modules are implemented using object oriented technologies.

4.2 Digital signature implementation techniques

Digital signatures is based on asymmetric cryptography, by having two key pairs (public, private), private key to encrypt messages (fields) and a public key to decrypt the message. The RSA algorithm is chosen because it provides digital signature and also encryption and decryption with high level of data security than DSA. The application digital signature module is developed using SHA1CryptoServiceProvider (Secure hash algorithm with key size 160-bits) for hashing and RSACryptoServiceProvider (Rivest, Shamir and Adleman) for digital signing provide by the .Net framework (System.Security.Cryptography.RSACryptoServiceProvider namespace has helper classes for digital signature . (<http://www.codeproject.com/Articles/25590/Cryptographic-Interoperability-Digital-Signatures>)). Actually the digital signing process is the original database column value hashed using SHA1 algorithm. The hashed column value is encrypted using private key and digitally signs the message and store it in database. To validate the signature compute the hash and compare with decrypted value , on successful matching the signature is valid. The basic level of authentication is provided by user credentials; onlt the registered users can access the application. All the users sensitive data can be secured by digital signing, so sentive data is not avalable in plain text in azure database storage.

4.3 Class diagram for digital signature, encryption, decryption, verification

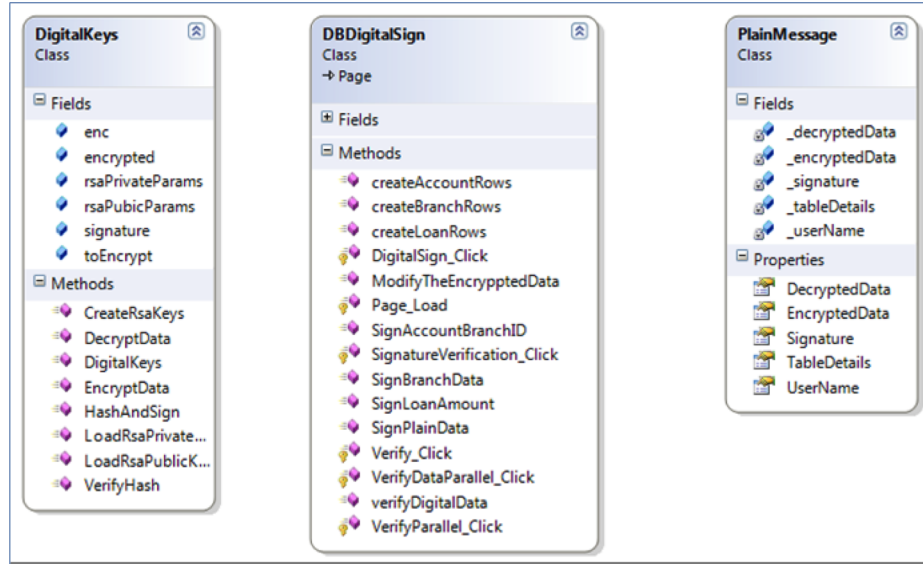


Figure 4.1: class diagram

The class diagram (Fig. 4.1) describes digital signature generation , signature verification, message encryption and decryption functionality. The diagram consists of Digital Keys, DBDigitalSign, PlainMessage, classes. Digital Keys class creates the RSA based key pair (private / public key), load the key pair, hash and sign message and verify the signed message. DBDigitalSign class uses the DigitalKeys class to sign and verify the data columns selected by the user. The PlainMessage class is used to store the plain message once the encrypted data is decrypted after the verification process. The plain message is provided to users to verify the correctness of the data as and when required.

4.4 Experimental setup

Digital signature is implemented using RSA algorithm (1024 bit) and SHA-1 as the hash algorithm to compute hash value on table fields of the database. The RSA algorithm and hash algorithms used in research for taken from third party library (Walton 2009). The application is developed in visual studio 2013 IDE, ASP.Net using C# as programming language. The application uses System.Security.Cryptography APIs for digital signing, verifying, encryption, decryption of the data and SQL server 2012 as database server on Microsoft Azure. The client machine communicates with the database server

using LINQ queries and Entity Framework as database model using .Net framework environment.

4.4.1 Microsoft ADO.NET entity framework 5.0 from local machine to connect to SQL server 2012 database in azure

This is an automated framework to achieve Object/Relational Mapping (ORM) which facilitates to work with relational data (CRUD operations) and removes the lots of coding overhead on users to access the database. LINQ queries are used to retrieve, insert and delete the data model objects in entity framework. All database modifications can be performed easily and without coding, so that user can concentrate more on application and domain specific development.

Entity Data Model (EDM) includes conceptual model objects, storage model objects and mapping information to map conceptual and relational database objects. This framework provides separation between the database design and application domain class model design. (Framework.net 2014) (<http://www.entityframeworktutorial.net/EntityFramework5/create-dbcontext-in-entity-framework5.aspx>).

Please refer to the appendix A for detailed information of how to create an Entity Data Model.

4.4.2 Experimental setup to migrate the web application from local to Virtual machine on windows azure

We evaluate the efficiency of digital signature using AzureDBApplication as test application on test database banking database system. The AzureDBApplication deployed in azure cloud to evaluate efficiency, so to achieve AzureDBApplication is deploying as VM. Create VM (kimbopranesh22) remote desktop connection, create windows server 2012 operating system in azure, visual studio 2013 and web browser (google chrome). Finally migrate AzureDBApplication in the created VM namely kimbopranesh22.

Please refer to the appendix B for details on how to migrate the AzureDBApplication from local to Microsoft Azure instance.

4.4.3 Deploying on existing ASP.NET AzureDBApplication to run on windows Azure website

The AzureDBApplication is deployed in azure cloud as web site to evaluate efficiency, so as to achieve this AzureDBApplication is deployed as cloud service. So that complete solution can be deployed in cloud as web site which more optimized solution. Create a web site mydb and deploy the AzureDBApplication as .Net cloud service. This scenario definitely has effect on efficiency in good way.

Please refer to the appendix C for details on how to deploy the AzureDBApplication on Microsoft Azure WebSite.

4.5 Pseudo code implementation of digital signature

We enter the (Fig. 4.5) uenters the table name, field name, number of queries and click on DigitalSign button. The specified username based public and private files are loaded into RSACryptoServiceProvider class by using a third party library (Walton 2009)(<http://www.codeproject.com/Articles/25590/Cryptographic-Interoperability-Digital-Signatures>) which has different helper classes to implemtent digital signatures. The specified number of AccountTable objects are added to AccountTable , sensitive field of a particular table get encrypted by invoking the DigitalKeys object EncryptData() method, hashed and signed by invoking HashAndSign() method. The signature and encrypted data are stored in Azure in a new table called DigitalFiledTable.

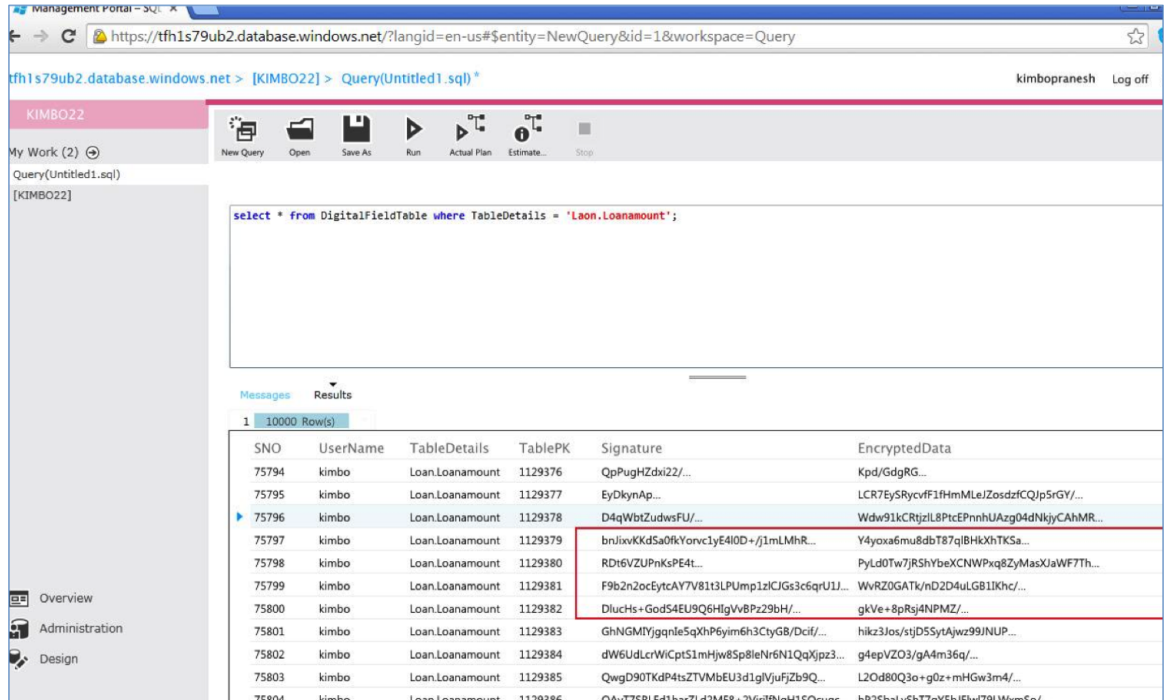
```
1 dk.LoadRsaPrivateKey(file);
2 dk.LoadPublicKey(file);
3 if ( Columnname_ID == "CustomerID")
4 {
5     var watch = new Stopwatch();
6     toEncrypt = enc.GetBytes(objAcct.CustomerID);
7     watch.start();
8     encrypted = dk.EncryptedData(dk.rsaPublicParams, toEncrypt);
9     signature = dk.hashAndSign(encrypted);
10    watch.Stop();
11 }
12
13 DigitalFieldTable dft = new DigitalFieldTable();
14 dft.SNO = obj.SNO+1+i;
15 dft.TableDeatils = TableDetails_ID.Text + "." + Columnname_ID.text;
16 dft.UserName+file;
```

```

17 dft.EncryptedData = covert.ToBase64String(encrypted);
18 dft.Signature = convert.ToBase64String(signature);
19 dft.TablePK = convert.ToString(objAcct.AccountNO);
20 objContextAddToDigitalFieldTables(dft);

```

Listing 4.1: Psudo code of hashing and signing



management Portal - SQL

https://tfh1s79ub2.database.windows.net/?langid=en-us#\$entity=NewQuery&id=1&workspace=Query

tfh1s79ub2.database.windows.net > [KIMBO22] > Query(Untitled1.sql) *

kimbopraneish Log off

KIMBO22

My Work (2)

Query(Untitled1.sql)

[KIMBO22]

select * from DigitalFieldTable where TableDetails = 'Loan.Loanamount';

Messages Results

1 10000 Row(s)

SNO	UserName	TableDetails	TablePK	Signature	EncryptedData
75794	kimbo	Loan.Loanamount	1129376	QpPugHZdx122/...	Kpd/GdgRG...
75795	kimbo	Loan.Loanamount	1129377	EyDkynAp...	LCR7EysRycvF1fHmMLe/ZosdzfCQjpSrGY/...
75796	kimbo	Loan.Loanamount	1129378	D4qWbtZudwsFU/...	Wdw91kCRtjzL8PtcEPnnhUAz904dNkjcCAhMR...
75797	kimbo	Loan.Loanamount	1129379	bnJixvKKdSa0fkYorvc1yE4l0D+/j1mLMhR...	Y4yoxa6mu8dbT87qIBHkXhTKSa...
75798	kimbo	Loan.Loanamount	1129380	RDt6VZUPnKsPE4t...	PyLd0Tw7JRShYbeXCNWpXq8ZyMasXJaWF7Th...
75799	kimbo	Loan.Loanamount	1129381	F9b2n2ocEytAY7V81t3LPUMP1zICjG63c6qrU1J...	WvRZOGATk/nD2D4uLG81IKhc/...
75800	kimbo	Loan.Loanamount	1129382	DlucHs+GodS4EU9Q6HlgVv8Pz296h/...	gkVe+8pRsj4NPMZ/...
75801	kimbo	Loan.Loanamount	1129383	GhNGMYjgqnl5qXhP6yim6h3CtyG8/Dcif/...	hikz3Jos/stjD5SytAjwz99INUP...
75802	kimbo	Loan.Loanamount	1129384	dW6UdLcrW/CptS1mHjw8Sp8leNr6N1QqXjPz3...	g4epVZO3/gA4m36q/...
75803	kimbo	Loan.Loanamount	1129385	QwgD90TKdP4tsZTVMbEU3d1glVjufjZb9Q...	L2Od80Q3o+g0z+mHGw3m4/...
75804	kimbo	Loan.Loanamount	1129386	CAuT3F8LEd1kx71d3Mf8+76u3Bh4H3CQw...	hD3ShuL6hT7xV5h3Eh79LWm5e/...

Overview Administration Design

Figure 4.2: Encrypted Data in Azure

KIMBO22Entities is database object for the database in azure. Once the selected data table field get encrypted and signed, the encrypted data and signature data are stored in azure database table called DigitalField table as separate fields. (Fig. 4.2). Each subscription of microsoft windows azure has access to create a storage or database which has a secret key inorder to control the entire data which is connected to a particular application. This shows that the data in the azure is secure and authenticanted.

```

public List<PlainText> VerifyDigitalData()
{
    int rowcount = Convert.ToInt16(Rowcount_ID.Text);
    string tabd = TableName_ID.Text + "." + Columnname_ID.Text;

    // Retrieve the records from azure database
    var DigSigList = (from d in objContext.DigitalFieldTables where ((d.TableDetails == tabd)) select d).Take(rowcount);

    // Load public and Private keys

    dk.LoadRsaPrivateKey(file);
    dk.LoadRsaPublicKey(file);

    verifytime = 0;

    foreach (DigitalFieldTable d in DigSigList)
    {
        PlainMessage pMsgs = new PlainMessage();
        encrypted = Convert.FromBase64String(d.EncryptedData);
        signature = Convert.FromBase64String(d.Signature);

        string strSig = Convert.ToBase64String(signature);

        var watch = new Stopwatch();
        watch.Start();

        // Verify the decrypted signature with encrypted data hash values
        if (dk.VerifyHash(dk.rsaPubParams, encrypted, signature))
        {
            watch.Stop();
            verifytime = verifytime + watch.ElapsedMilliseconds;

            // storing into msg list
            pMsgs.DecryptedData = dk.DecryptData(encrypted);
            pMsgs.UserName = d.UserName;
            pMsgs.TableDetails = d.TableDetails;
            pMsgs.SNumber = d.SNO;
            msgList.Add(pMsgs);
        }
    }
    return msgList;
}

```

Figure 4.3: Verification of Digital signed data

when the (Fig. 4.3) user enters the Table name, field name, no. of records and click on the verify button. All the fields of the given records are retrieved from the azure database and the private, public keys of the admin are also loaded. Then the verifyhash() method is invoked on each record as per the count. Actually here when the record is retrieved the encrypted field and digital signature of the records are retrieved to azure VM and digital signature is decrypted, Hash algorithm SHA-1 is applied on the encrypted field if (verify hash) of these hashes is same then the verification is successful. This shows that data integrity is maintained.


```

var DigSigList = ( from d in objContext.DigitalFieldTables
                    where ((d.TableDetails== tabd ) ) select d).Take(count);

dk.LoadRsaPublicKey(file);
dk.LoadRsaPrivateKey(file);

foreach (DigitalFieldTable d in DigSigList)
{
    PlainMessage pMsgs = new PlainMessage();

    encrypted = Convert.FromBase64String(d.EncryptedData);
    signature = Convert.FromBase64String(d.Signature);

    string strSig = Convert.ToBase64String(signature);

    pMsgs.DecryptedData = dk.DecryptData(encrypted);

    pMsgs.UserName = d.UserName;
    pMsgs.TableDetails = d.TableDetails;
    pMsgs.SNumber = d.SNO;

    msgList.Add(pMsgs);
}

```

Figure 4.4: Decryption of Digital signed data

when the user provides the table name , field name, no of records to decrypt, and clicks on Decrypt button. The specified no of records are retrieved using LINQ query and stored in list. The specified username based public and private files are loaded into RSACryptoServiceProvider object. After that the specified field get decrypted by invoking DigitalKeys object DecryptData() method. (Fig. 4.4)

4.6 Software development life cycle:

SDLC methodology chosen for the application is spiral model because of its iterative nature and waterfall model nature. This approach is suitable for developing research thesis as proof of concept or prototype. In this methodology all phases of this approach are covered for every iteration with more focus on risk and at end a prototype is produced. The software is developed in engineering phase and testing at the end of the phase. The spiral model suits for the R &D project, so the fast prototype or POC can be implemented iteratively by evaluating each phase.

4.7 Software testing methodology

In the testing phase of the thesis: Unit testing, Integration Testing, System Testing, Performance Testing and Security Testing are executed.

- Functional testing: : Testing of application modules and classes at the functional (object) level. In this all modules in the system are tested individually. All the data model class implementation is tested for its functionality verification. For example signature generation, verification modules can be tested whether the signatures are getting generated and verified can test individually. Classes like signature object, error handling can be tested.

Chapter 5

Evaluation

We evaluate the efficiency of digital signature on different database table columns. The main focus of the empirical validation is to evaluate the performance of securing the data from database using the digital signature. We evaluate the efficiency of digital signature by applying it to different tables and columns. we evaluate the following scenarios:

1. The signing of data.

In this case the execution time taken for insertion and signing the fields are counted and tabulated.

2. The encryption of data.

In this case execution time is taken only for encrypting the selected fields are counted and tabulated.

3. The decryption of data.

In this case execution time is taken only for decrypting the selected fields are counted and tabulated.

4. The Verification of data.

In this case execution time is taken only for verifying the selected fields are counted and tabulated.

Our approach uses four important tables Loan, AccountTable, Branch, and DigitalFieldTable for evaluation. In which the fourth table is DigitalFieldTable used to store user data, encrypted data, signed data.

We use a banking database (i.e) AzureDBApplication as a test case for the evaluation. we conduct evaluation :

1. In local machine. ¹
2. In Azure cloud environment:
 - Evaluation on azure instance.
 - Evaluation after deploying on azure website.

The Table. 5.1 summarizes the total time taken for executing the operation to digital sign.

We conduct digital signing evaluation on fields like street, Loanamount, Branchname on tables Accounttable, Loan, Branch. Table 5.1 shows that performance of execution time for digital signing the fields degrades due to large volumes of data.

Table 5.1: Time taken for only digital signing in local machine

TableName	ColumnName	No of Records	Time taken to sign in local (Sec)
Loan	Loanamount	100	0.688
Loan	Loanamount	500	3.2
Loan	Loanamount	1000	6.53
Loan	Loanamount	10000	184.833
Branch	Branchname	100	0.722
Branch	Branchname	500	3.37
Branch	Branchname	1000	6.70
Branch	Branchname	10000	180
AccountTable	CustomerID	100	0.655
AccountTable	CustomerID	500	3.73
AccountTable	CustomerID	1000	7.64
AccountTable	CustomerID	10000	96.815

The Table. 5.2 summarizes the time taken for executing the operation to decrypt the corresponding fields of database tables in local machine. The time taken for decrypting the fields is better in local machine compared with azure site.

¹ RAM 4 GB, Processor 2.4GHz Intel Core i5, HDD 500GB

Table 5.2: Time taken to only decrypt the fields randomly in local machine

TableName	ColumnName	No of Records	Time taken to decrypt in local (MillSec)
AccountTable	Street	100	236
AccountTable	Street	500	1090
AccountTable	Street	1000	2265
AccountTable	Street	10000	22984
Loan	Loanamount	100	223
Loan	Loanamount	500	1172
Loan	Loanamount	1000	2324
Loan	Loanamount	10000	22605
Branch	Branchname	100	234
Branch	Branchname	500	1159
Branch	Branchname	1000	2372
Branch	Branchname	10000	21997

Jansma & Arrendondo (2004) mention signature verification of RSA is efficient than signature creation. Table. 5.3 summarizes the time taken for verifying the corresponding fields of database tables is better than signature generation i.e (signature verification is efficient than signature creation). So the evaluation in this case done is correct.

Table 5.3: Time taken to only verify in local machine

TableName	ColumnName	No of Records	Time taken to verify in local (MillSec)
AccountTable	CustomerID	100	672
AccountTable	CustomerID	500	3631
AccountTable	CustomerID	1000	8021
AccountTable	CustomerID	10000	77873
Loan	Loanamount	100	630
Loan	Loanamount	500	4708
Loan	Loanamount	1000	7775
Loan	Loanamount	10000	72290
Branch	Branchname	100	506
Branch	Branchname	500	2784
Branch	Branchname	1000	6491
Branch	Branchname	10000	85432

5.1 Evaluation in azure instance

The Table. 5.4 summarizes the time taken for executing the operation to sign the corresponding fields of database tables in azure virtual machine .²

We conduct the same evaluation in azure instance, the signing time, decryption time is more when compared with local machine because of the configuration of virtual machine.

Table 5.4: Time taken to only sign in azure instance

TableName	ColumnName	No of Records	Time taken to sign in azure (MillSec)
AccountTable	CustomerID	100	767
AccountTable	CustomerID	500	3791
AccountTable	CustomerID	1000	7763
AccountTable	CustomerID	10000	76301
Loan	Loanamount	100	758
Loan	Loanamount	500	3885
Loan	Loanamount	1000	7655
Loan	Loanamount	10000	76623
Branch	Branchname	100	765
Branch	Branchname	500	3836
Branch	Branchname	1000	7815
Branch	Branchname	10000	77023

The Table. 5.5 summarizes the time taken to verify the corresponding fields of database tables in azure virtual machine. The performance is better because RSA algorithm supports efficient verification than signature generation.

² RAM 3.27GB, Cores: 2

Table 5.5: Time taken to only verify in azure instance

TableName	ColumnName	No of Records	Time taken to verify in azure (MillSec)
AccountTable	CustomerID	100	1
AccountTable	CustomerID	500	8
AccountTable	CustomerID	1000	39
AccountTable	CustomerID	10000	663
Loan	Loanamount	100	2
Loan	Loanamount	500	11
Loan	Loanamount	1000	64
Loan	Loanamount	10000	658
Branch	Branchname	100	4
Branch	Branchname	500	20
Branch	Branchname	1000	23
Branch	Branchname	10000	535

The Table. 5.6 summarizes the time taken for executing the operation to decrypt the corresponding fields of database tables in azure virtual machine.

Table 5.6: Time taken to Decrypt in Azure instance

TableName	ColumnName	No of Records	Time taken to decrypt in instance (MillSec)
AccountTable	CustomerID	100	768
AccountTable	CustomerID	500	3852
AccountTable	CustomerID	1000	7605
AccountTable	CustomerID	10000	76277
Loan	Loanamount	100	745
Loan	Loanamount	500	3678
Loan	Loanamount	1000	7625
Loan	Loanamount	10000	76764
Branch	Branchname	100	757
Branch	Branchname	500	3877
Branch	Branchname	1000	7598
Branch	Branchname	10000	76375

5.2 Experimental Analysis after deploying on azure web-site

For above specified scenarios digital signing, verifying, encryption, decryption execution time is evaluated and analysed.

There is considerable reduction in execution time of digital signing and verification. When the evaluation is performed on azure web site is very much efficient than on local and virtual machine because web application deployed on azure is an optimized solution as set of virtual machines run on azure websites. Digital signature not only limited to security but also includes authentication and reliability of data transfer. Certain techniques like maintaining key pair, key exchange mechanism involves lots of analysis and challenges. Encryption provides security and privacy to data; however digital signature is quite promising in providing security in all aspects.

The Table. 5.7 summarizes the time taken for executing the operation to sign the corresponding fields of database tables in azure web site.

Table 5.7: Time taken to sign in azure site

TableName	ColumnName	No of Records	Time taken to sign in azure site (MillSec)
AccountTable	CustomerID	100	229
AccountTable	CustomerID	500	1171
AccountTable	CustomerID	1000	2195
AccountTable	CustomerID	10000	22551
Loan	Loanamount	100	228
Loan	Loanamount	500	1156
Loan	Loanamount	1000	2374
Loan	Loanamount	10000	22939
Branch	Branchname	100	208
Branch	Branchname	500	1153
Branch	Branchname	1000	2182
Branch	Branchname	10000	22741

The Table. 5.8 summarizes the time taken for executing the operation to verify the corresponding fields of database tables in azure web site.

Table 5.8: Time taken to verify in azure site

TableName	ColumnName	No of Records	Time taken to verify in azure site (MillSec)
AccountTable	CustomerID	100	12
AccountTable	CustomerID	500	57
AccountTable	CustomerID	1000	104
AccountTable	CustomerID	10000	1079
Loan	Loanamount	100	4
Loan	Loanamount	500	40
Loan	Loanamount	1000	80
Loan	Loanamount	10000	1079
Branch	Branchname	100	12
Branch	Branchname	500	44
Branch	Branchname	1000	107
Branch	Branchname	10000	1079

The Table. 5.9 summarizes the time taken for executing the operation to encrypt the corresponding fields of database tables in azure web site.

Table 5.9: Time taken to encrypt in azure site

TableName	ColumnName	No of Records	Time taken to encrypt in azure site (MillSec)
AccountTable	CustomerID	100	10
AccountTable	CustomerID	500	24
AccountTable	CustomerID	1000	51
AccountTable	CustomerID	10000	483
Loan	Loanamount	100	7
Loan	Loanamount	500	30
Loan	Loanamount	1000	54
Loan	Loanamount	10000	479
Branch	Branchname	100	10
Branch	Branchname	500	18
Branch	Branchname	1000	68
Branch	Branchname	10000	545

The Table. 5.10 summarizes the time taken for executing the operation to decrypt the corresponding fields of database tables in azure web site.

Table 5.10: Time taken to decrypt in azure site

TableName	ColumnName	No of Records	Time taken To decrypt in azure site (MillSec)
AccountTable	CustomerID	100	883
AccountTable	CustomerID	500	4395
AccountTable	CustomerID	1000	8774
AccountTable	CustomerID	10000	87920
Loan	Loanamount	100	878
Loan	Loanamount	500	4444
Loan	Loanamount	1000	8839
Loan	Loanamount	10000	88165
Branch	Branchname	100	860
Branch	Branchname	500	4348
Branch	Branchname	1000	8723
Branch	Branchname	10000	88166

The (Fig. 5.1) compares the execution time of signing and verification in azure web site for CustomerID.

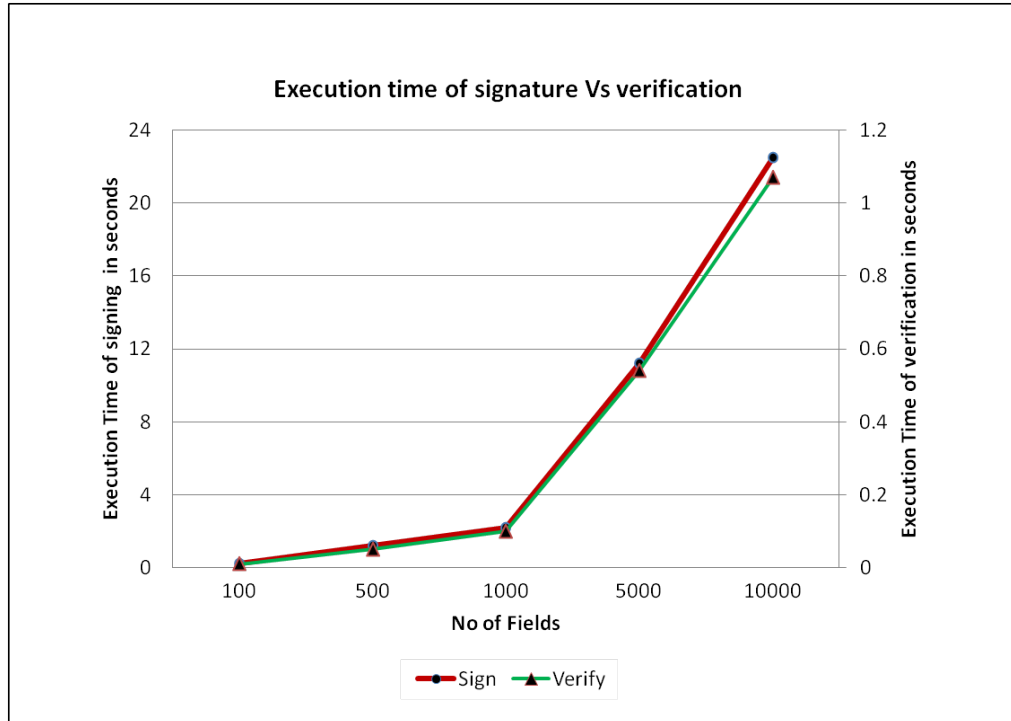


Figure 5.1: Exeuction time of sign Vs verification

The reason behind the increase in digital sign time is due to the use of RSA algorithm because it uses key length of 1024 bits for signature generation which provides strong signing and good level of security. Not only that RSA computational time is also high. Jansma & Arrendondo (2004) mention that in RSA algorithm the time taken for signature generation is low compared with other algorithms one of algorithm is Elliptic Curve Digital Signature algorithm (ECDS) this is proved on the above evaluation tables in all three environments because of the greater key length (1024 bit). Signature verification is faster than generation. Finally, from this analysis it is proven that signature verification is faster when compared with signature generation because of RSA key is 1024 bits. If the key length was about only 128 to 256 bits then signature generation performance would have been better, but the level of security will be definitely reduced. (As AzureDBApplication is a banking management system, security is the primary concern than time. Padmavathi & Kumari (2013) also mention that the most commonly used algorithm for encryption is RSA algorithm its the most secure way of authentication on cloud provider site at the same time it is too slow in terms of encryption for large data volumes.)

The (Fig. 5.2) compares the execution time of encryption and decryption in azure web site for CustomerID.

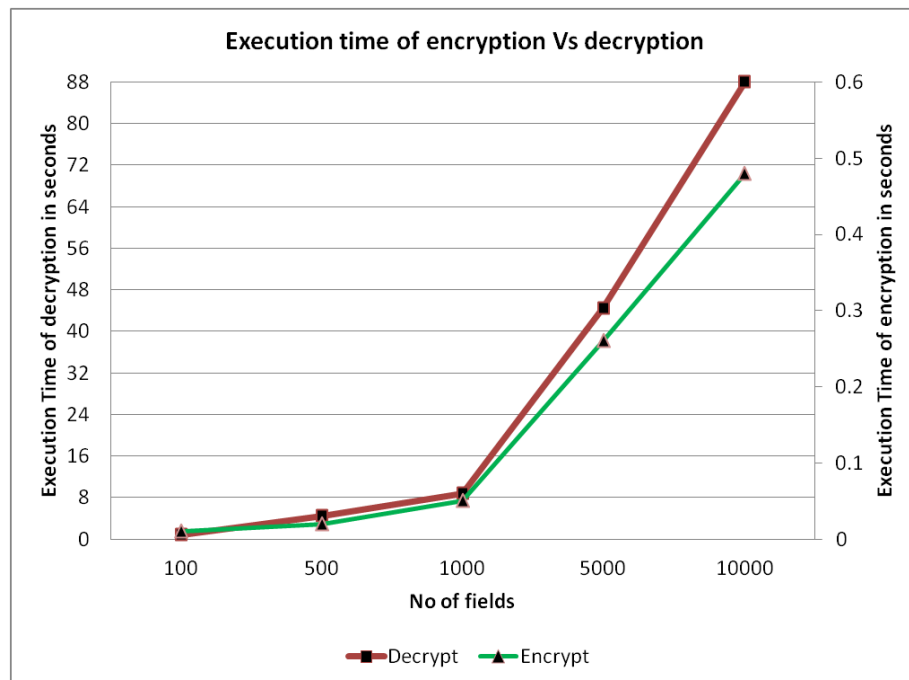


Figure 5.2: Execution time of encryption Vs decryption

The reason behind the increase in decryption time then that of encryption is due to

the RSA algorithm design, like for n bit key length the encryption will be twice ($2n$), and the decryption will be thrice ($3n$). So because of this reason time for decryption of the fields are more when compared with encryption as per the graph.

The (Fig. 5.3) compares the execution time of signing and verification in azure web site for Branchname.

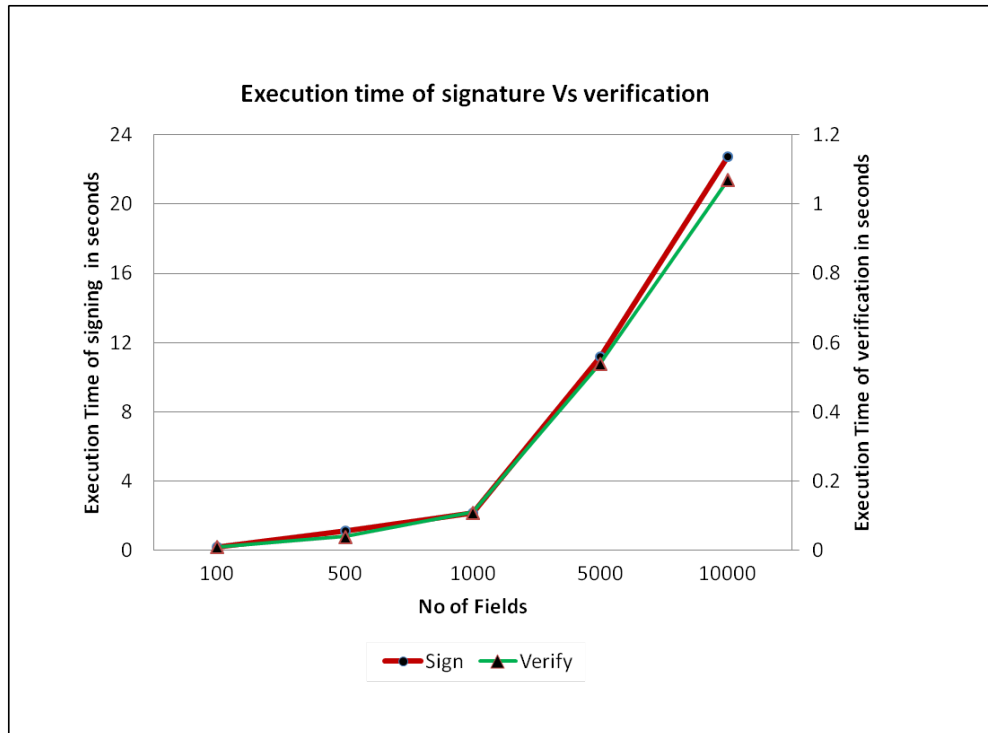


Figure 5.3: Execution time of sign Vs verification

The (Fig. 5.4) compares the execution time of signing and verification in azure web site for Branchname.

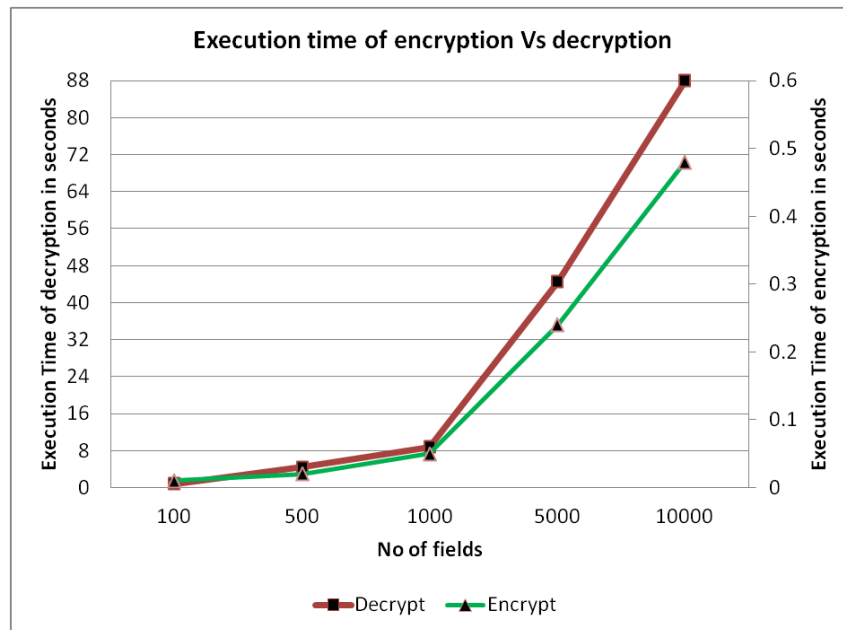


Figure 5.4: Execution time of encrypt Vs decrypt

Chapter 6

Conclusions

6.1 Conclusion

In this research certain challenges has considered such as data security, data privacy, data integrity and anti deniability that is a part of database management. The database security is very vital factor that has to be considered in any software development corporate or industries. Almost all in every software application, such as distributed applications, client server applications, Web applications database involvement is mandatory database security is one of factors that has to be handled seriously. In conclusion although there are many security technologies like Fullhomomorphic encryption, Layered based encryption, PKI encryption. Every technology involves a layer of computation, complexity. In any encryption technique if the Key is lost or stolen then security is broken. Digital signature is technique which can be applied to any kind of data. As outlined in the introduction and main body, efficiency of encryption based on security components did not address few aspects like data integrity, ant deniability and data privacy completely. This is purely addressed by using digital signature. Although digital signatures cannot prevent fraud from being attempted, they prevent attempted fraud from succeeding by giving application the ability to detect fraudulent transactions. By providing digital signature as security to DBAAS the data is completely protected because both digital signature and DBAAS have very unique attributes. Using digital signature the complex computations are not required. Digital signature system will not have any additional servers that could become cause for application to stop. This implies scalability of database increases. This is resolved using digital signature. Therefore, digital signature is more efficient when compared with other encryption techniques.

The paper describes an approach and contribution towards to implement digital signed database as a service. The most significant achievement is including the digital signature object in DBAAS database table. Unlike other systems this digital signature solution can be applied to any database not only banking system. Service provider azure can compute and return the results set for the requested query in very secure manner since the application operates on encrypted field data. In this research the digital signature is applied to only selected fields so that results set are compact, computational time is less and memory overhead to reduced, there is no need to sign entire table.

When the intruders break the security at database level (authentication), we can prevent the table data being modified by adding methodology of digital signature. In this we dont need to apply the digital signature to all the tables, but only selected sensitive database tables. In this research digital signature generation, verification and storing the digital signature as an object into a database will be implemented. There is third party .Net API's (System.Security.Cryptography) are available to generate and verify the digital signature (RSA algorithm with 1024 bits and SHA1). Using the third party API digital signature can be implemented and stored into database efficient way.

6.2 Future Work

The most exciting part of the paper is implementing TPL in digital signed database in DBAAS. We believe that adding concurrency and parallelism really brings significant reduction in processing time. To handle the multiple requests from multiple users/-clients , TPL library is used because nowadays, all computers (workstations, laptops, servers) come with multiple cores and most of the applications fail to harness the full potential of this computational ability of the systems . The task parallel library allows writing code which is adjustable to itself with the number of Cores available in the computer. So it is sure that the software would auto-upgrade itself with the upgrading environment. The concept of "Task" is introduced by TPL and Task parallelism is the process of running these tasks in parallel. A Task is an independent unit of work, which runs within a program. The TPL is more efficient, scalable and more programmatic control is possible than a thread or work item. The thesis paper contributes by providing security to database by using digital signature and efficient handling of multiple requests from multiple users.

Bibliography

- Aki, S. G. (1983), ‘Digital signatures: A tutorial survey’, *Computer* **16**(2), 15–24.
URL: <http://dx.doi.org/10.1109/MC.1983.1654294>
- Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J. & Brandic, I. (2009), ‘Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility’, *Future Generation Computer Systems* **25**(6), 599 – 616.
URL: <http://www.sciencedirect.com/science/article/pii/S0167739X08001957>
- Curino, C., Jones, E. P., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H. & Zeldovich, N. (2011), ‘Relational cloud: A database-as-a-service for the cloud’.
- Dubey, A. & Wagle, D. (2007), ‘Delivering software as a service’, *The McKinsey Quarterly* **6**(2007), 2007.
- Framework.net, E. (2014), ‘Create DbContext in Entity Framework 5.0’, <http://www.entityframeworktutorial.net/EntityFramework5/create-dbcontext-in-entity-framework5.aspx>. [Accessed 1-August-2014].
- Gahi, Y., Guennoun, M. & El-Khatib, K. (2011), A secure database system using homomorphic encryption schemes, in ‘DBKDA 2011, The Third International Conference on Advances in Databases, Knowledge, and Data Applications’, Think Mind, St. Maarten, The Netherlands Antilles, pp. 54–58.
- Hacigumus, H., Iyer, B. & Mehrotra, S. (2002), Providing database as a service, in ‘Data Engineering, 2002. Proceedings. 18th International Conference on’, IEEE, San Jose, California, pp. 29–38.
URL: <http://dl.acm.org/citation.cfm?id=876875.879015>
- huang Wu, C. (2010), Self-generated-certificate digital signature, in ‘Genetic and Evolutionary Computing (ICGEC), 2010 Fourth International Conference on’, IEEE, Shenzhen, China, pp. 379–382.
URL: <http://dx.doi.org/10.1109/ICGEC.2010.100>
- Jansma, N. & Arrendondo, B. (2004), ‘Performance comparison of elliptic curve and rsa digital signatures’, *nicj. net/files* .
- Kadhem, H., Amagasa, T. & Kitagawa, H. (2009), A novel framework for database security based on mixed cryptography, in ‘Internet and Web Applications and Services, 2009. ICIW ’09. Fourth International Conference on’, IEEE Computer Society, Venice Mestre, Italy, pp. 163–170.
URL: <http://dx.doi.org/10.1109/ICIW.2009.31>
- Lehner, W. & Sattler, K.-U. (2010), Database as a service (dbaas), in ‘Data Engineering (ICDE), 2010 IEEE 26th International Conference on’, IEEE Computer Society, Los Alamitos, CA, USA, pp. 1216–1217.
- Mani, M., Shah, K. & Gunda, M. (2013), ‘Enabling secure database as a service using fully homomorphic encryption: Challenges and opportunities’, *CoRR* **abs/1302.2654**.

- Meijer, H. & Aki, S. (1982), 'Digital signature schema', *Cryptologia* **6**(4), 329–338.
URL: <http://www.tandfonline.com/doi/abs/10.1080/0161-118291857154>
- Narasimha, M. & Tsudik, G. (2005), Dsac: Integrity for outsourced databases with signature aggregation and chaining, in 'Proceedings of the 14th ACM International Conference on Information and Knowledge Management', CIKM '05, ACM, New York, NY, USA, pp. 235–236.
URL: <http://doi.acm.org/10.1145/1099554.1099604>
- Ngai, E. & Wat, F. (2002), 'A literature review and classification of electronic commerce research', *Information Management* **39**(5), 415 – 429.
URL: <http://www.sciencedirect.com/science/article/pii/S0378720601001070>
- Padmavathi, B. & Kumari, S. R. (2013), 'A survey on performance analysis of des; aes and rsa algorithm along with lsb substitution technique', *International Journal of Science and Research (IJSR)* **2**(4), 170–174.
- Popa, R. A., Redfield, C., Zeldovich, N. & Balakrishnan, H. (2012), 'Cryptdb: Processing queries on an encrypted database', *Commun. ACM* **55**(9), 103–111.
URL: <http://doi.acm.org/10.1145/2330667.2330691>
- Rewagad, P. & Pawar, Y. (2013), Use of digital signature with diffie hellman key exchange and aes encryption algorithm to enhance data security in cloud computing, in 'Proceedings of the 2013 International Conference on Communication Systems and Network Technologies', CSNT '13, IEEE Computer Society, Gwalior India, pp. 437–439.
URL: <http://dx.doi.org/10.1109/CSNT.2013.97>
- Sklavos, N., Kitsos, P., Papadomanolakis, K. & Koufopavlou, O. (2002), Random number generator architecture and vlsi implementation, in 'Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on', Vol. 4, IEEE, Phoenix-Scottsdale, AZ, USA, pp. IV–854–IV–857 vol.4.
- Somani, U., Lakhani, K. & Mundra, M. (2010), Implementing digital signature with rsa encryption algorithm to enhance the data security of cloud in cloud computing, in 'Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on', IEEE, Solan, India, pp. 211–216.
- Walton, J. (2009), 'Cryptographic Interoperability: Digital Signatures - CodeProject', <http://www.codeproject.com/Articles/25590/Cryptographic-Interoperability-Digital-Signatures>.
[Online; accessed 19-July-2014].
- Zhang, J. (2010), A study on application of digital signature technology, in 'Networking and Digital Society (ICNDS), 2010 2nd International Conference on', Vol. 1, IEEE, Wenzhou, China, pp. 498–501.

Appendix A

How to create an Entity Data Model

Steps to create Entity Data Model using VS 2013.

- Open Visual Studio 2013 and ASP.Net Web Application project.
- Now, add EDM by right clicking on the project in the solution explorer select Add click on New Item and select ADO.NET Entity Data Model from popup, give Name and click Add button.(Fig. A.1)

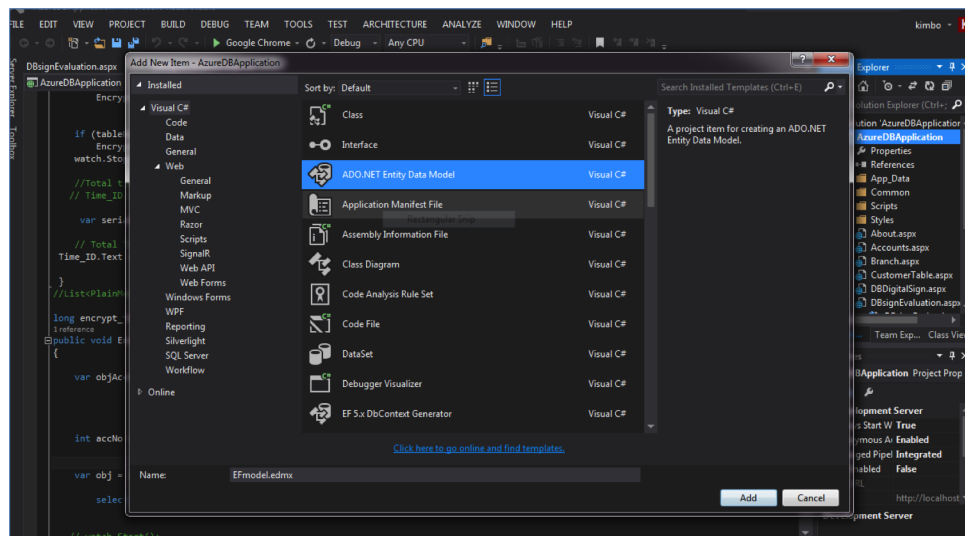


Figure A.1: ADO.NET Entity Model

- VS 2013 opens with four options in EDM wizard select option EF designer from database option and click Next.(Fig. A.2)

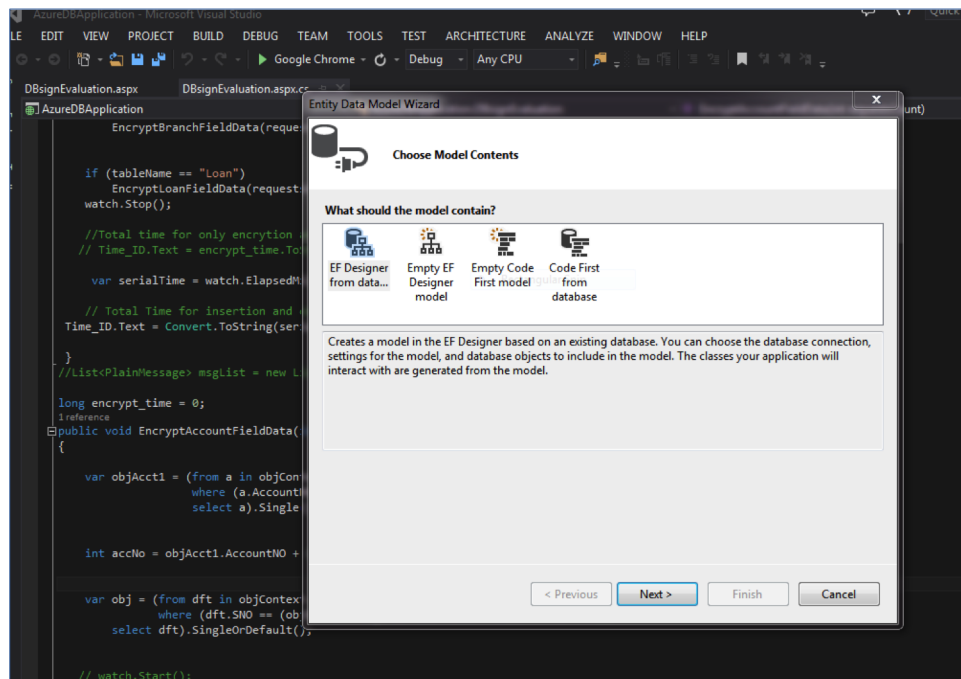


Figure A.2: Entity Data Model

- Select existing SQL Server 2012 DB connections or create new connection by clicking New Connection button. Use existing db connection to KIMBO22 azure SQL server database. This will also add connection string to your web.config file with default suffix with db name. That can be changed as per the requirement and click Next after setting the db connection.(Fig. [A.3](#))

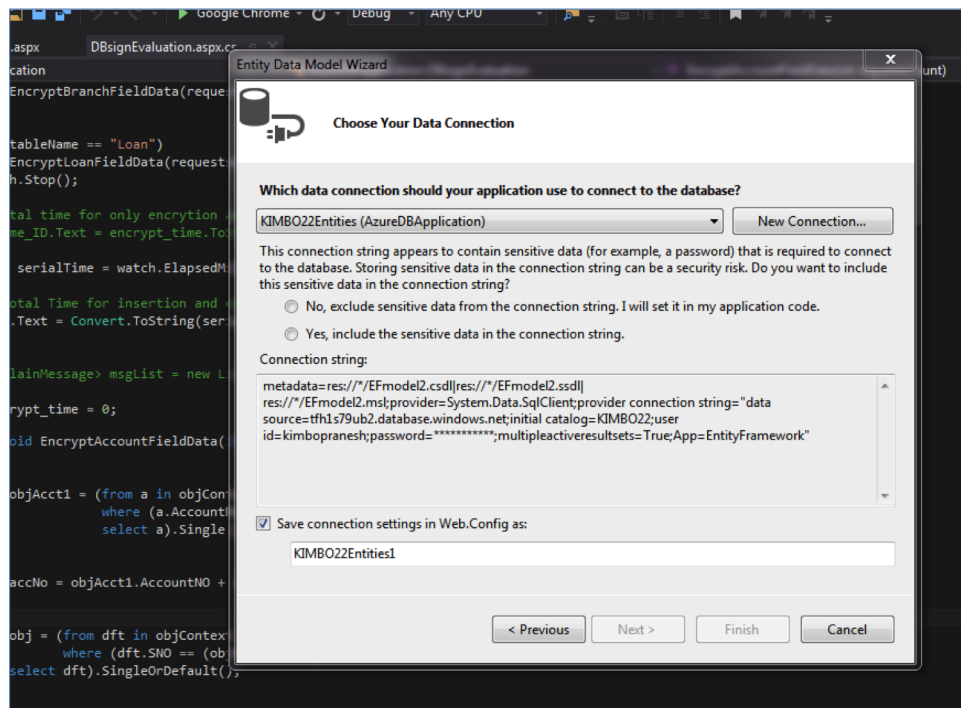


Figure A.3: Database Connection

- Select the version of Entity Framework and click Next.(Fig. A.4)

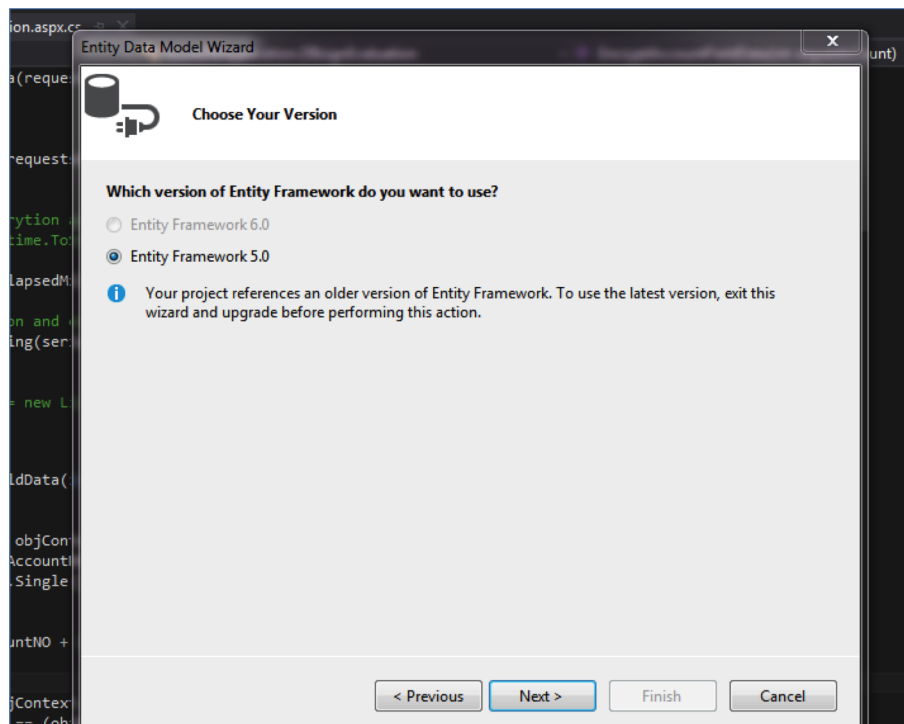


Figure A.4: Version selection

- It will show the tables, views and stored procedures in the database. Select tables, views and SPs that are required , keep the default checkboxes selected and click Finish. Change model Namespace as per requirement.

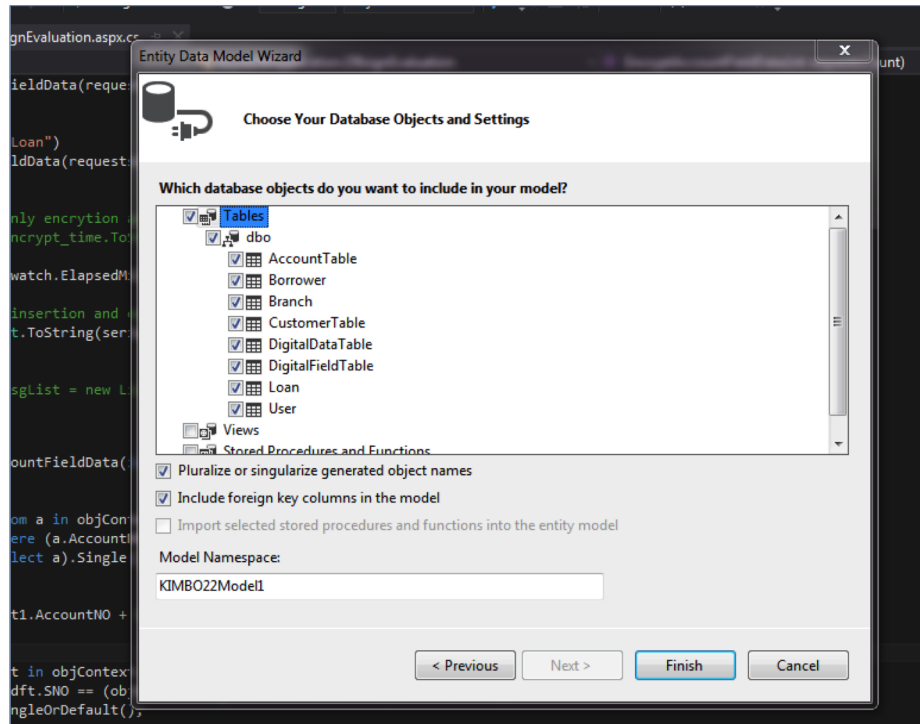


Figure A.5: Database Settings

- After clicking on Finish, EFModel.edmx file into the project. Double click on EFModel.edmx file , which displays all the entities and its relation with database.

Appendix B

How to Migrate the AzureDBApplication from local to Azure instance

- First login into azure account with the credentials and create a new virtual machine with instance name, specified domain name, configuration of VM with 2 cores and 3.75 RAM.
- When VM is created successfully it will be in running status with unique DNS name. In this case the instance name is Kimbopranesh22. Next once the Kimbopranesh22 VM status is in running state then the web application in local can be migrated to the VM in the azure with few steps. Initially in order to migrate the code from local machine to remote desktop we need a remote desktop connection.
- The remote desktop connection is established in the local machine by download a link file (kimbopranesh22.rdc) from azure.
- Then the webapplication can be migrated to Vm in azure for further evaluation that will be explained on the next chapter (Fig. [B.1](#))

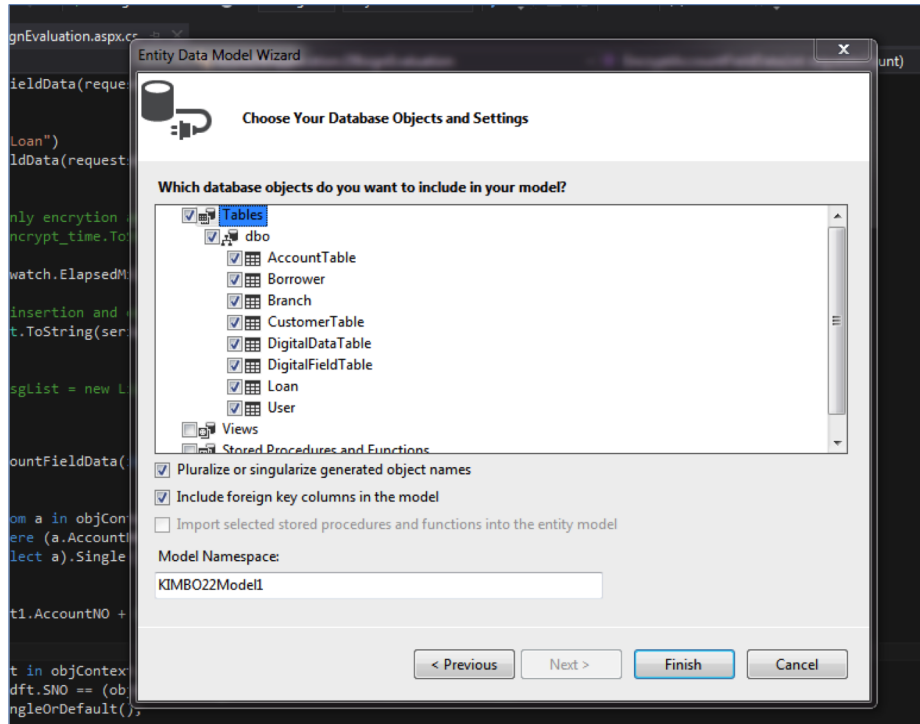


Figure B.1: Connecting to remote desktop of azure

We evaluate the efficiency of digital signature using AzureDbApplication as test application on test database banking database system. The AzureDBApplication deployed in azure cloud and evaluate efficiency, so to achieve this AzureDBApplication is deployed as VM. So that complete solution can be deployed in cloud.

Appendix C

How to Deploy an existing ASP.NET AzureDBApplication to run on Windows Azure website

Deployment of ASP.Net web application comprises of two main components.

1. Deploying an existing AzureDBApplication as web role in azure cloud service.
2. Publish the web application using VS 2013.
 - First select the AzureDBApplication in the solution explorer of the visual studio, right click and add New project in the list appeared then a window will be displayed with Windows Azure Cloud Services option.
 - This will add Windows azure cloud service to the project and simultaneously a new window will appear on the screen to choose the ASP.Net web role. Then AzureDBApplication is added as role in the windows azure cloud service to be deployed on azure website.
 - When the application is ready select the AzureDBApplication right click and select Publish option. Publish wizard will be displayed , in this option user can modify the settings for the deployment by signing with azure credentials are create a site to deploy the AzureDBApplication . There will be four configuration settings displayed in the wizard as shown (Fig. [C.1](#))

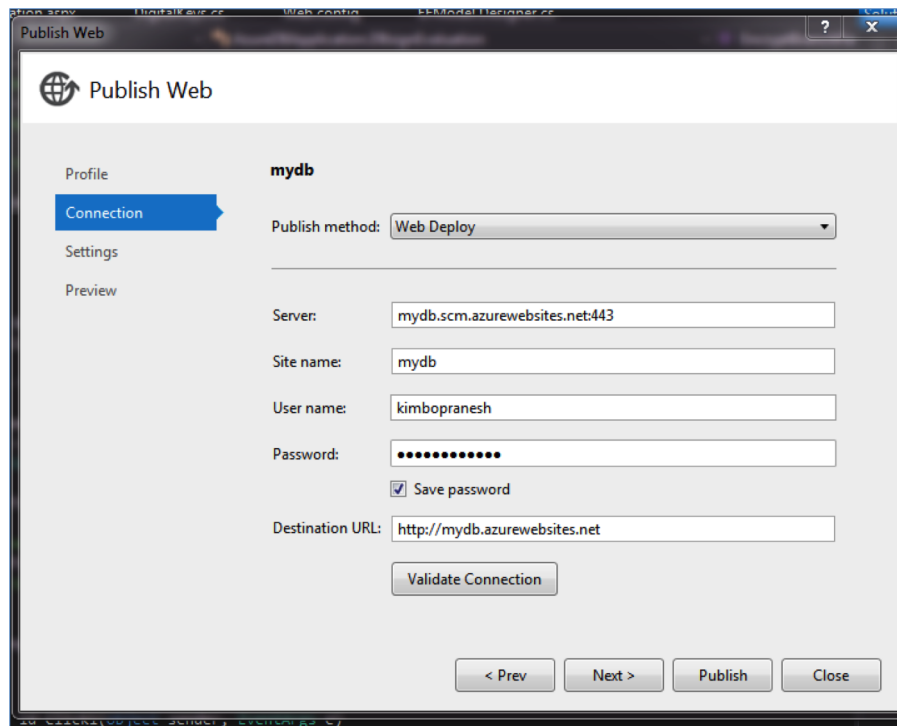


Figure C.1: Configuration setting to deploy the web application to azure site

- Finally choose next to publish application in the created site (<http://mydb.azurewebsites.net>).