Database Protection Security: ASCII Primary Key Record

Mohammad S. Ghatasheh
Information Technology Faculty
Isra University
Amman, Jordan
MohadGh@ipu.edu.jo

Zahraa Muhsen
Information Technology Faculty
Isra University
Amman, Jordan
z_muhsin@ipu.edu.jo

In today's electronic world, information security has become one of the most critical issues facing companies. External and internal security threats continue to grow in intensity, threaten the daily safety of sensitive database information. Many database security methods focus on data in transit, without taking measures to protect the data where it is. Organizations that need to protect sensitive information, database encryption should be a key part of their security policy. This research produces a novel method for database encryption. It proposed ASCII Primary Key Record (APKR) method, which suggests to encrypting all tables except the primary column. Many encryption methods suggest using one public key for all records in the table, which produce a problem in producing encrypted data. APKR is depending on encrypting the database with the RSA method, with using a different public key for each record in the table that means no duplicate producing in the encrypted data.

**Keywords:** Database Security, Database Encryption, Public Key, Private Key, RSA, Primary Field, ASCII.

1. Introduction

The Internet has created an ever-growing number of server sites that deliver information to individuals around the world at the speed of light. With Internet, organizations develop effective business practices, which build around highly available information resources. Applications like e-banking or e-commerce are prime examples of these real-time online resources that deliver value-added services through high availability databases. With the development of these server-based systems, the requirement for secure storage of data has become more important than ever before. The downside to this growth of online information systems is the new threat which has developed for these systems such as the threat of illegal or uncontrolled access to these valuable resources. On the top of these threats is the growing concern of the core assets of organizations which could be modified, compromised, or even deleted information, and this could put the organizations out of business within seconds. To dismiss these threats, organizations need to spend all control of access to information resources. They also need to protect data against disclosure, modification or deletion, finally they need to refine the access control and data protection down to individual pieces of information to a granular level (column, record or field) in order to guarantee an optimal use of database resources (protect only what is necessary to protect and leave non), sensitive data unencrypted.

While database systems offer some security services (content management, access control or database encryption), availability of the services generally come at a price. Database systems generally offer an “either-or” approach to encryption: either the database is totally encrypted or it is not. So, security cannot be provided without a radical hit on the overall level of performance and an increase in the investment in resources.

The eternal struggle between performance and security hits database systems in Internet world. Organizations always determine the degree of security at their databases according to the balance between the cost and database confident. Encryption is one of the most security approaches applied on the data. It is a very expensive process since encrypted database fields are larger than unencrypted fields and required a time to any process at the encrypted data [1].

In most database encryption methods, they encrypt the entire columns, and if all data are encrypted (encrypt and decrypt each piece of data) that will result a huge amount of overhead (especially in huge database). This means addition, insert and update operations will require time for encrypting all fields in every single row. When the database is encrypted, techniques such as search, query or report cannot be established [1, 2, 3, 4].

Many database encryption methods use one public and one private key to encrypt all records in the table (either to encrypt certain fields or all the fields for every record). This encryption will cause duplicate encrypted data in similar filed contents. Our proposed methods use a different public key for each record in the database which will not cause any redundant encrypted data in similar fields’ contents, and it will produce more efficient encrypting process.

In encryption process each record will be encrypt using its own public key, but in decryption process one private key is used to decrypt all the records. Encryption process can be applied when data in motion or data at rest (this proposed encryption process applied when data at rest (stored)). Most attacks do not occur on data-in-motion, but they occur against the end points of data, where data sit for long periods of time (data at rest).
Encryption of “data-at-rest” can be on all the columns in every table or on all but one column in a table, and this is the one that will be used in this method.

RSA algorithm will be used in encrypting all the fields, RSA uses two keys: public (in encrypting) and private (in decrypting). Fig.1. illustrated the proposed methods.

2. Public-Key Encryption

Public-key encryption (asymmetric encryption) involves a pair of keys, public key and private key, associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with a public key can be decrypted by a private key. Fig. 2 shows the way for working public-key encryption [2, 3, 4].

Compared with symmetric-key encryption, public-key encryption requires more computation (not appropriate for large data). However, it's possible to use public-key encryption to send a symmetric key (to encrypt additional data). Data encrypted with the public key can be decrypted only with the private key. This would not be a desirable way to encrypt sensitive data because anyone with your public key could decrypt the data. Nevertheless, private-key encryption is useful since you can use your private key to sign data with your digital signature (an important requirement for electronic commerce and other commercial applications of cryptography). Client software such as Communicator can be used your public key to confirm the message that was signed with your private key and hasn't been tampered [5].

The idea of public key cryptography is closely related with the idea of one-way function. Given an argument value x, it is easy to compute the function value f(x), whereas it is intractable to compute x from f(x). Here “intractable” is understood in the sense of complexity theory. A receiver can receive secured information from the sender by publicizing the function f(x), and then the sender computes f(x) from x, and sends f(x) to the receiver. Only the receiver can calculate x from f(x), assuming it is intractable to compute x from f(x) [6, 7].

The most commonly used implementations of public-key encryption are based on algorithms patented by RSA Data Security.

3. Key Length and Encryption Strength

In general, the strength of encryption is related to the difficulty of discovering the key, which in turn depends on both the cipher used and the length of the key. For example, the difficulty of discovering the key for the RSA cipher most commonly used for public-key encryption depends on the difficulty of factoring large numbers, a well-known mathematical problem [5].

Encryption strength is often described in term of the size of the keys used to perform the encryption (in general, longer keys provide stronger encryption). Key length is measured in bits. For example, 128-bit keys use the RC4 symmetric-key cipher which supported by Secured Socket Layer (SSL) that provide significantly better cryptographic protection than 40-bit keys that use with the same cipher [5].

Different ciphers may require different key lengths to achieve the same level of encryption strength. The RSA cipher used for public-key encryption, for example, can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based. Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values. Thus a 128-bit key been used in a symmetric-key encryption cipher would provide stronger encryption than a 128-bit key been used in RSA public-key encryption cipher [8]. This explains why RSA public-key encryption cipher must use a 512-bit key (or longer) to be considered strong cryptographically, whereas symmetric key ciphers can achieve approximately the same level of strength with a 64-bit key. Even this level of strength may be vulnerable to attacks in the near future [9, 10].

4. Public Key Cryptosystems Requirements

IEEE The public key encryption algorithm must fulfill the following conditions as explain in [2] (where A is sender, B recipient):

1- It is easy for recipient B to generate a pair of keys.
2- It is easy for sender A to encrypt using B’s public key.
3- It is easy for B to decrypt using his own private key.
4- It is impossible for an attacker knowing B’s public key to know B’s private key.
5- It is impossible for an attacker knowing B’s public key and the cipher to decrypt.

In the following example, suppose that person A wants to make a public key, and that person B wants to use that key to send a message to A. Also suppose that A will send numbers as a message to B. Assume that A and B have agreed on a method to encode text as numbers:

1- Person A selects two prime numbers (p=23 and q=41) but A should use larger numbers.
2- Person A multiplies p and q together to get pq=(23)(41)=943. 943 is the “Public Key”, which tells to person B.
3- Person A also chooses another number e which must be relatively prime to \((p-1)(q-1)\).
   In this case, \((p-1)(q-1)=(22)(40)=880\)
   So e=7 is fine. e is also part of the Public Key, so B is also know about the value of e.
4- Now B knows enough to encode a message to A. Suppose the message is the number M=35.
5- B calculates the value of C:
   \[ C = M^e \pmod{N} = 35^7 \pmod{943} \]
   \[ 35^7 = 64339296875 \]
   \[ 64339296875 \pmod{943} = 545 \]
   The number 545 is the encoding that B sends to A.
6- Then,
   \[ 357 = 64339296875 \Rightarrow 4339296875 \pmod{943} = 545 \]
7- Now A wants to decode 545. To do so, A needs to find number d such that
   \[ C^d \equiv 1 \pmod{(p-1)(q-1)} \]
   Or in this case, such that:
   \[ 7d \equiv 1 \pmod{880} \]
   A solution is d=503, since
   \[ 7*503=3521=4(880)+1 \equiv 1 \pmod{880} \]
8- To find the decoding, A must calculate
   \[ C^d \pmod{N} = 545503 \pmod{943} \]
   Notice that:
   \[ 503=256+128+64+32+16+4+2+1 \]
   This means:
   \[ 545503 \pmod{943} = 545 \]
   \[ 324.18.215.795.857.400.923.545 \]
   \[ \equiv 35 \pmod{943} \]
   Then we obtain table 1:

<table>
<thead>
<tr>
<th>Table 1. Decoding Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculation</td>
</tr>
<tr>
<td>5451 (mod 943)</td>
</tr>
<tr>
<td>5452 (mod 943)</td>
</tr>
<tr>
<td>5454 (mod 943)</td>
</tr>
<tr>
<td>5458 (mod 943)</td>
</tr>
<tr>
<td>54516 (mod 943)</td>
</tr>
<tr>
<td>54532 (mod 943)</td>
</tr>
<tr>
<td>54564 (mod 943)</td>
</tr>
<tr>
<td>545128 (mod 943)</td>
</tr>
<tr>
<td>545256 (mod 943)</td>
</tr>
</tbody>
</table>

Using this calculation, A can decrypt B’s message and obtain the original message N=35.

No one is going to criticize the user for using a key that is too long provide his software still performs adequately. However, the biggest danger in using a key that is too large is the false sense of security it provides to the implementers and users. "Oh, we have 4096-bit security in our system" may sound impressive in a marketing blurb, but the fact that the private key is not adequately protected or the random number generator is not random may mean that the total security is next to useless.

If we are encrypting the plaintext with a conventional symmetrical algorithm like DES, our session key is going to be 64 bits long. Triple DES will need 192 bits, and AES will need up to 256 bits. That gives us great security. Unlike the simple example above where we had to deal with a series of integers, to encrypt a 256-bit key with a 1024-bit RSA modulus, that means we need a single representative message integer. In fact, the user need to pad the 256 bits to ensure that he has a large enough integer before he encrypts it with RSA [2].

5 Proposed Method ASCII Primary Key Record (APKR)
The ASCII Primary Key Record (APKR) methods started with determine the public keys and the private key for each record in the table. Primary field will represent
the public key (primary field not a duplicate container),
that means each table has numbers of public keys as
much as the records number. In decryption process, one
private key will be suggested by the user to be used to
decrypt all records in the table as shown in Fig.3.

The following steps show the APKR process:

Step one (Choosing Data): User must determine a
table to be encrypted (example Fig.1).

Step two (ASCII Converting): In order to provide
single private key for a record, each corresponding
primary filed value will be converted to ASCII value
(example Fig.1).

Step three (Public Keys): As mentioned before, every
record has its own public key which has been set from its
ASCII primary value (insure no redundant in public
keys). These public keys used to encrypt the information
in all the fields excluding the primary field (example
Fig.1).

Step four (Private Key): After choosing a public key
for each record, a single private key will be produced
from them. The single private key will be used to decrypt
the encrypted data for selecting a record or records in the
table. (Fig.4 explains this step).

Step five (RSA): In this phase, fields except the
primary one will be encrypted using the RSA algorithm.
The basic RSA encryption is based on public key and
private key. The public key is published, and it is used to
encrypt data. The private key is kept secret by the user to
retrieve the encrypted data. The brilliant part about RSA
encryption is that the encrypted data cannot be decrypted
without discovering the private keys. Every record will be
encrypted according to its public key (ASCII primary
field value), so there is no duplication encrypted data
even at similar data stored in the rest of the table records.
Looking for a relation on the encrypting Boolean fields
will be so hard in APKR because similar contents will be
different in each record (each one has a different public
key) (example Fig.3).

Step Six (Encrypted Data): Two options available in
this step to store the encrypted data. First option:
replacing the current unencrypted data with encrypted
one, so the user can later decrypt it when entering the
private key using applications that support RSA
algorithm. This option is more secure since one encrypted
copy of the database was stored (reduces store database
capacity). It is good to remember that the search operation
in APKR will be efficient (depend on the primary key),
but if the search operation was dependent on any
encrypted field, then decrypting fields record are
required. Also database utilities and report generators
which considered as powerful case tools in dealing with
databases will not be used for the encrypted databases.
This is a big challenge to entering the encrypting world
which can simply cancel the encryption idea from some
companies' strategies which mainly based on reports and
queries. Encryption does not protect data from being
deleted from database, so this will considered as a
limitation for this option because there is no another copy
of this encrypted database. Also it's hard to specify the
deleted records because they are all encrypted.

In APKR user treats data with some kind of
applications that support encrypted databases. The user
can save, update and delete data from the database, but
problem released in data manipulation because they
required data decryption before updating. Also if
encrypted database is sniffed or hacked, hacker will not
be able to understand anything from those tables (except
the primary field) (Fig.5 illustrates this step).

Second option: is to take a mirror of database (a copy)
as illustrated in Fig.6. That means there will be two
databases a normal form (unencrypted), and secure form
(encrypted) in different places to ensure good protection.
This option produce less risk level, because there is
another copy of the database, which the user system can
use in any damage on the normal form (hacked or
crashed) that will cause a lower risk level. On the other
hand, this option is costly because it required more
capacity since there are two copies of the database.
Referential integrity between the first copies of the
database (encrypted one) and the other one (unencrypted)
is also considered as a challenge to use this option. Backup time needs much more time than the first option, either in backing up the entire database or in dealing with specific record. Search operation in this option will be efficient here also if it is depend on the primary key, that because if the search operation was depends on any other encrypted field, then there is a need to decrypt each record (with all its fields) to check if is it match the required condition/s in the query. Database user here not treats only with some kind of applications that support treating with encrypted databases so the user can save, update and delete data from the database, but also the system must have other applications to take a back up of the database and ensure the referential integrity between the two copies.

![Fig. 6. Mirror of Database](image)

In both options, data type will be changed after encrypting, that cause many problems from that mentioned before. In addition both options also need extra capacity need to save the encrypted data (depends on the used algorithm) so it is very important to note every field size in the database.

Step seven (Retrieving Data): When retrieving data from the database in this proposed method, one private key is used to decrypt the required record (Fig. 7). RSA algorithm is used to decrypt the required record after entering the private key, that process can be done using special applications.

![Fig.7. Retrieving Data](image)

Without entering that private key, data will not be understandable by anyone even if it is retrieve. Fig.8 shows the result example of using APKR.

![Fig.8. Decrypting Data Using Private Key](image)

6. Conclusion and Recommendations

There are many cost result from applying databases security requirements in any large organizations, but organization or company cannot dispense with these security requirements. Encrypting data is a step in better securing any enterprise, it does not protect data from being deleted or modified, and user cannot join or search on encrypted data. Also data sharing process cannot be allowed with application packages, database utilities and report generators.

APKR methods is depend on encrypting the database with the RSA method, but the main idea foxed on using different record public key in the table that means there is no duplicate in produced encrypted data.

The recommendations of choosing RSA method required to change fields’ type because the encrypted data contain different types of data. It is necessary to increase the field size to the maximum size because always the encrypted data size is larger than the plain data size. The best practice in APKR is choosing the primary field data as a prime integer (to produce the public and private keys in easy way). After encrypting the data, user must replace the original database with the encrypted database to increase the security. Save the private key in a secure and strong protection. Inquiring or searching the database can be efficient if it was dependent on the primary key field in the table. Updating or modifying process in the table must depend on its primary key (to decrypt only the needed record not to decrypt all the records in the database).

REFERENCES