On the Difficulty of Securing Web Applications using CryptDB

Ihsan H. Akın
Dept. of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: ihakin@wpi.edu

Berk Sunar
Dept. of Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: sunar@wpi.edu

***Abstract*—CryptDB has been proposed as a practical and secure middleware to protect databases deployed on semi-honest cloud servers. While `CryptDB` provides sufficient protection under Threat-1, here we demonstrate that when `CryptDB` is deployed to secure the cloud hosted database of a realistic web application, an attacker to database or a Malicious Database Administrator (mDBA) can easily steal information, and even escalate his privilege to become the administrator of the web application. Our attacks, fall under a restricted form of Threat-2 where we only assume that the attackers or the mDBA tampers with the `CryptDB` protected database and is opens an ordinary user account through the web application. Our attacks, are carried out assuming perfectly secure proxy and application servers. Therefore, the attacks work without recovering the master key residing on the proxy server. At the root of the attack lies the lack of any integrity checks for the data in the `CryptDB` database. We propose a number of practical countermeasures to mitigate attacks targeting the integrity of the `CryptDB` database. We also demonstrate that the data integrity is not sufficient to protect the databases, when query integrity and frequency attacks are considered.**

***Keywords*-CryptDB; database integrity; query integrity; frequency attacks**

I. Introduction

Motivated by drastic savings in hardware, software, and IT costs cloud computing gained mainstream popularity. However, due to the unprecedented level of data sharing cloud computing also gave rise to new security concerns. For instance, despite intense efforts by the research community, secure cloud based data storage has remained elusive. While standard encryption techniques provide a baseline solution, they are too rigid, i.e. further useful operations on encrypted data such as text search, or standard aggregation operations on encrypted databases are difficult to support in a practical manner. Especially given the diversity of queries supported by databases such as point, range, and aggregate queries the task at hand becomes even more difficult.

Numerous techniques were proposed to execute private queries on encrypted databases. Deterministic encryption is a well known and efficient – yet insecure – method to execute point queries. Song et al. proposed an elegant technique based on the coupling of stream ciphers and pseudorandom function to perform search directly on encrypted data [17]. Agrawal et al. proposed an order preserving encryption (OPE) scheme that preserves the order relation of plaintexts in ciphertexts, which permits range queries in encrypted database [2]. Boldyreva et al. proposed to use OPE on symmetric encryption and gave a formal security analysis [4]. Popa et al. proposed a protocol to achieve order preserving encryption [16] by using a binary tree structure to store the order relation on a semi-trusted cloud server. This is the first scheme to achieve ideal security, i.e. only the order relation is leaked on the semi-trusted server. Along with homomorphic encryption these techniques, e.g. Paillier's scheme [13], form a toolset for building encrypted database.

In 2011, Popa et al. proposed a software only, server-side solution named CryptDB aiming practical deployment by relaxing security requirements, i.e. confidentiality without integrity, and by introducing a trusted proxy server. A significant goal of `CryptDB` is to providey confidentiality of cloud hosted databases, without disturbing the business processes of the cloud providers. The scheme is tightly integrated with MySQL, and heavily tested. `CryptDB` elegantly combines DET, OPE, additive homomorphic encryption [13], and a modification of Song's idea to permit encrypted search on text fields. Popa et al. claim that except for some well-known weaknesses such as frequency analysis, order relation and queries hits, the database is secure as long as the proxy server is trusted [14]. More specifically, the machines hosting the database management software and their administrators are not trusted (semi-honest, non-malicious passive adversaries), but the application and the proxy server are trusted.

Our Contribution. In this work we focus on the security of web applications that are running with a `CryptDB` protected database hosted on the cloud. It is well known that `CryptDB` was designed without data integrity and authenticity in mind [15]. We have divided databases secured by `CryptDB` into two types; as **single user databases**, e.g. personal calendar and contacts database, and **multi-user databases**, e.g. the phpBB, HotCrp, and OpenEMR web applications would use `CryptDB` in this mode. Under

The work of this author was supported by TUBITAK-2219
The work of this author was in part supported by NSF Awards #1117590 and #1319130.

realistic scenarios played on CryptDB protected multi-user sites, we show that on-line attackers or a Malicious Database Administrator (mDBA) can exploit the lack of data and query integrity protection measures in CryptDB and steal other user's private data and even mount privilege escalations attacks on the web applications

	Login access to site is required	Can be achieved by
Attack III-A	Yes	mDBA or Attackers
Attack III-A	Yes	mDBA or Attackers
Attack VI-A	Yes	mDBA or Attackers
Attack VI-B	No	sDBA or Attackers

Table I
OUR ATTACKS AND THEIR REQUIREMENTS

These attacks, Table I fall under a limited form of Threat-2 in the CryptDB security model with the additional assumption of login access to the CryptDB enabled database application. Even further, it is shown with additional scenarios that existing and new countermeasures to ensure database integrity and authenticity are not sufficient to secure CryptDB. The lack of query integrity protection and susceptibility to frequency analysis lie at the root of these attacks. Our attacks do **not target** the proxy or application servers, or the **Master Key** (MK) stored on the proxy server as assumed in the Threat 2 security model.

This remainder of the paper is organized as follows; In SectionII, we give a brief background on CryptDB. Tampering attacks targeting the database are presented in Section III. In Section IV, we describe the experimental validation of the attacks. In Section V, we present countermeasures against database tampering attacks. In Section VI, we present two new attacks based on lack of query integrity and frequency analysis. We draw the conclusions in Section VII.

II. A BRIEF BACKGROUND ON THE CRYPTDB

In this section, we briefly review CryptDB. CryptDB is designed to secure MySQL databases with the goal of supporting practical deployments in the cloud. With an average performance overhead on MySQL of only %21-26 CryptDB provides an attractive choice to developers who want to encrypt their databases [14]. To deploy CryptDB in applications**, in general,** adding only few lines of code is sufficient.

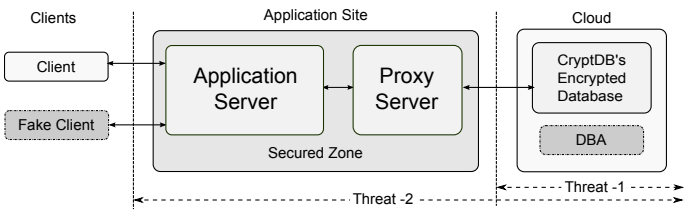

Figure 1. The attack methodology. The application and the proxy server are not under our attack

The CryptDB protocol assumes four parties as depicted in Figure 1:

1) **End Users:** Users are owners of the data or parties permitted to access the data through an application server. Users wish to carry out common database operations securely. Data must be revealed only to the intended users. **In Single user mode,** user maintains his own key to protect their private data. In addition, **in multi-user mode**, **MK is used to encrypt the common data** and CryptDB provides a private messaging service **and data accessible only for some entities** for the users of the database.
2) **Application server:** This is the server through which the users interact with the application that connects to the CryptDB database over the CryptDB proxy server. The application server manages the access control mechanisms. There may be a need to adjustment of a few lines of code in the application to correctly communicate with CryptDB. The application server takes the user's password to transmit to proxy server.
3) **Proxy server:** At the core of CryptDB lies the proxy server. It stores a Master Key (MK), and obscures the column names using the MK. In the single-user mode, the user's password is used to perform all the encryptions on behalf of the user. In multi-user mode; MK, random keys and users' keys are used to encrypt the sensitive data. In both modes, the user's key must be securely transferred to the proxy server. Thus, the proxy server is the ultimate trusted entity in the CryptDB protocol. The proxy server maintains an annotated schema on the server that defines the encryption level on the data whenever an encryption is needed. The proxy server is responsible for transforming the application's queries to match with the annotated schema, before forwarding them to the MySQL database.
4) **MySQL Server:** The MySQL server is placed on the cloud. It contains a number of User Defined Functions (UDFs), which are executed by the MySQL server with the instruction of the proxy server. The design of CryptDB introduces a minor change to MySQL. If encryption is enabled on a database column, then MySQL administrators only see the data in encrypted form. Moreover, MySQL administrators have access to the queries issued by the proxy server as well as to their responses. The MySQL server is assumed to be semi-honest, i.e. the server is assumed to be trusted to carry out the operations precisely as instructed. However, the server administrator is also curious, i.e. he may examine all data as well as the queries and their responses during the transactions.

CryptDB uses various encryption schemes to process SELECT, RANGE, JOIN, and simple aggregation queries

over various forms of encrypted data:

- **Randomized Symmetric Encryption (RND):** AES is used with an IV in CBC mode. Due to randomization, the scheme is semantically secure. However, this forbids any computation on the ciphertext.
- **Deterministic Symmetric Encryption (DET):** AES in ECB mode is used to support conditional selection queries via equality checks on ciphertexts. While this type of encryption allows comparisons, it also opens the door to frequency analysis attacks.
- **Order Preserving Encryption (OPE):** Boldyreva's OPE scheme is used to support range queries. This scheme reveals the minimum, maximum, order and the most significant digit. With the knowledge of the ciphertext of plaintext x and its successor $x+1$, an attacker may recover the other plaintexts without using the encryption key [3], [9]. Recently a more secure OPE protocol was introduced in [16].
- **Additive Homomorphic Encryption (HOM):** Paillier's encryption scheme [13] is used to support simple aggregation queries and arithmetic operations such as ADD, INCREMENT, SUM, and AVERAGE.

A. Schema of a CryptDB Protected Database

To support various queries over encrypted data, CryptDB's team designed four types of onions, i.e. layers of encryption, as shown in Figure 2. The onion layers encrypt data stored in a column denoted by v. The onions are stored in the separate columns of the encrypted database to support various queries. For each onion level, a different key derived from the IDs of the table, column, onion, onion layer with the aid of a pseudo-random permutation, e.g. AES is used:

$$K_{t;c;o;l} = \mathsf{PRP}_{MK}(\text{table } t; \text{column } c; \text{onion } o; \text{layer } l).$$

The DET and OPE layers can be protected by RND encryption. The onion layers supported in CryptDB are as follows:

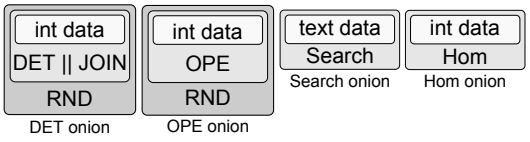

Figure 2. The onions are in their most secure level. The RND layers of DET and OPE must be removed to perform range and point queries.

1) **DET Onion:** This onion is formed as $\mathsf{RND}(\mathsf{IV}, \mathsf{DET}(v) \| \mathsf{JOIN\_DET}(v))$. To be able to execute a point query, if an RND layer exist, it must be removed falling back to a $\mathsf{DET}(v)$ onion which allows point queries. The $\mathsf{JOIN\_DET}(v)$ part is used to perform equality JOIN operations among the tables. Dynamic adjustment and predetermined version of the equality JOIN are suggested and implemented.
2) **OPE Onion:** This onion is formed by $\mathsf{RND}(\mathsf{IV}, \mathsf{OPE}(v) \| \mathsf{JOIN\_OPE}(v))$. As in DET, The RND layer must be removed to execute a query. After removal, the $\mathsf{OPE}(v)$ onion can be used to perform range queries. Dynamic adjustment for $\mathsf{JOIN\_OPE}(v)$ to perform dynamic JOIN operations with range queries was proposed but not implemented in CryptDB. Therefore, $\mathsf{JOIN\_OPE}(v)$ functionality must be declared in advance in the schema, for queries that require a range JOIN.
3) **Homomorphic Onion:** $\mathsf{Hom}(v)$. To support additive based operation on the data, Paillier's homomorphic encryption scheme is used. Paillier's scheme is CPA secure, and therefore an RND layer is not needed.
4) **Search Onion:** This onion uses a modified version Song et al.'s searchable encryption scheme [17]. The words are, first, split by predetermined delimiters, edited to eliminate duplicates, padded to the same size, randomly permuted, and finally encrypted with a simplified version of Song et al.'s idea.

CryptDB Threat Model. CryptDB was designed **and considered** to survive two types of threats:

- **Threat 1** covers database management system server compromises. In this threat CryptDB guards against a curious database administrator or external attacker. CryptDB assumes a passive attacker model and does not consider attacks compromising the integrity of the server.
- **Threat 2** includes attacks that target the entire system including the application, the proxy as well as the database server. The assumption is that the attack would persist for a short time period and only records of users accessing their data during this time period would be compromised since the remaining entries in the database are kept in encrypted form.

B. Multiple Principals in CryptDB

For multi-user web sites, like phpBB, CryptDB supplies private messaging and forum protection according to groups' permissions. Private messaging achieved by encrypting the message with a random key key_{msg_i} and then encrypting this message key for the parties; one with the sender's key $E_{Sender\_key}(key_{msg_i})$ and other with the receiver's key $E_{Receiver\_key}(key_{msg_i})$. These encryptions are performed by the proxy-server. In thread-2 model only the online user's password and their allowed groups are compromised. Similarly, the permission based access, like forums in phpBB are protected. The posts in the forum are encrypted with a random key, and the key is encrypted for each user who can access the forum.

C. Query execution in CryptDB

CryptDB is designed for securing cloud hosted databases under two use scenarios: the **single-user mode** and the

Employees Table	
ID	Name
12	Alice Wu
17	Evil Eve
25	Bob Marley

Table 0x65BD659A (Employees table in CryptDB)						
Obscured ID columns			Obscured Name columns			
C1-IV	C1-DET	C1-OPE	C2-IV	C2-DET	C2-OPE	C2-Search
0x651E	0x3A3	0x5CD	0x567	0x93B	0x122	0x453
0x211	0xE33	0x555	0x5B7	0x666	0x11A	0x3B8
0x2A	0x4C4	0x66A	0x713	0x78A	0x282	0x453

Table II
A SAMPLE TABLE (LEFT) AND ITS CRYPTDB VERSION (RIGHT). THE COLUMN NAMES ARE KEPT IN OBSCURED FORM. HOWEVER, THE **same** KEY IS USED TO ENCRYPT ALL DATA IN THE SAME COLUMNS.

multi-user mode. In both approaches, the user's password must be transferred securely to proxy server by traditional ways. In the single-user mode, the user has a **private database** secured by his password. We will focus for the remainder of this paper on the multi-user mode. In the multi-user mode, the application server manages the access control mechanism. Encryption and the decrption of all data are performed by the proxy-server. The user passwords are used by the proxy server to ensure the confidentiality of the private messages exchanged between users. The restricted data are encrypted with a random key, and this key encrypted for every user who can access with keys and stored in the database. The users personal information that are required in the signup and login processes are encrypted with the MK.

When a client performs an action that is turned into a query by the application server, if it is a private message, the query and the user's password is transferred to the proxy server. The proxy server rewrites this query; encrypts the necessary parts according to annotated schema with **MK**, and performs the necessary modifications to the query so that it matches with the current database schema. Before executing the query on the database, the proxy server may issue a pre-query so that the onions of the table which are under *selection*, can correctly execute the query. This pre-query removes the RND layer of the necessary columns of the table or adjusting the keys for a JOIN query.

An Example. Table II shows a simplified CryptDB schema (right) of the table (left) with only two columns. Assume a legitimate user wants to retrieve all information about on Evil Eve. Per request of the user, the application server prepares a query to be executed on Employees Table, where the query is

```
SELECT * from FROM users WHERE Name =
    'Evil Eve'
```

The proxy server receives this query from the application server. First, it checks the Name column's current layer corresponding to the DET onion. If needed, it calls an UDF function to remove the RND layer of the DET onion, i.e. the C2-DET column, so that the equality can be tested. With the layer dropped down to DET, it executes the modified query, in which the table and column names are obscured and the strings are encrypted with the correct key to match the encryption of the table.

```
SELECT * from FROM 0x65BD659A WHERE
    C2-DET = '0x666'
```

MySQL executes this query in encrypted form and returns the result to proxy server which then decrypts the result and sends it back to the application server.

III. TAMPERING ATTACKS ON CRYPTDB

In this section, we consider the security of **multi-user** online sites run via the open source phpBB [1], a bulletin board software, and HotCRP [8], a conference management software where the backend of the sites is secured using CryptDB.

The attacks outlined in this section, do fall under a restricted version of type Threat-2. For instance, we **do not**:

- target the application and proxy servers, or
- steal the MK from the proxy server or
- target any user's personal computer to steal the user's key.

Rather, the attacks presented here are enabled by online attackers. They gain access to the database server and also act as a registered users of the web application accessing the system only through the application server. This attack could also be enabled by the Malicious DBA (mDBA) taking on two roles, i.e. mDBA managing the cloud database, but also acting as a registered user of the web application. And another combination for this attack; it could also be enabled by a naïve user who leaks information to the mDBA. This scenario is quite plausible since many online applications such as phpBB and hotCRP allow easy creation of user accounts which is meant to facilitate user acquisition. For the remainder of this section we will simply refer to the attackers as mDBA.

For simplicity we assume that the proxy and application servers are running in the same physical server, although split versions will not affect the viability of the attacks. We also assume that not all of the onions are under an RND layer so that database can execute the queries. In what follows, with the lack of any measures to protect the integrity of the database rows, we demonstrate the viability of the proposed attacks under several realistic scenarios.

A. Attacking phpBB

Consider an online forum that runs on phpBB. The site administrators reduce operating costs by moving their

CryptDB enabled database to a cheap cloud server. CryptDB's proxy server and the application server are both run on the forum administrator's closely guarded physical server.

user_id	username	user_password	...	usr_lvl	salt
0x85B	target i	MD5(Yc!N6PaS$)	...	1	0x8347
0x7A3	attacker	MD5(EasyAttack)	...	0	0xEC62
0x74A	target j	MD5(IT@cKd.)	...	2	0x7014

Table III
PHPBB USERS TABLE

user_id	username	user_password	...	usr_lvl	salt
0x85B	target i	**MD5(EasyAttack)**	...	1	**0xEC62**
0x7A3	attacker	MD5(EasyAttack)	...	**1**	0xEC62
0x74A	target j	**MD5(EasyAttack)**	...	2	**0xEC62**

Table IV
PHPBB USERS TABLE IN THE TIME OF ATTACK

Fake User Account Identification. To gain access to the application server, the mDBA, first registers to the website as a regular user. The first goal of the mDBA is to identify the username_clean field in the phpbb_users table for the user account he has established. The table names are obscured as shown in Table II. To get the right table, the mDBA enables the MySQL's query log. After these steps, the mDBA logs into the website. To verify the user login attempt, phpBB executes an SQL query such as

```
SELECT user_id, username,
    user_password, user_passchg,
    user_pass_convert, user_email,
    user_type, user_login_attempts
FROM phpbb_users
WHERE username_clean = 'mDBAs_FAKE_USER'
```

It is clear that the username_clean's onion's RND layer must be revealed in advance by the proxy server to execute this point query on CryptDB. In the mean time on the cloud server, the mDBA checks the log the read queries. By checking the login time in the CryptDB enabled MySQL server log, the mDBA can identify the query and recover the DET encrypted version of his username of his fake phpBB account. At this point the mDBA can identify all traces for his fake account in the CryptDB database. Even further, in the current public version of CryptDB the table creation operation preserves the orders of columns. Since the phpBB source code is public, and since the order is preserved during CryptDB table creation, one can match them against their encrypted versions. At this point, the mDBA can tamper with the CryptDB tables stored on the cloud server, and with the help of his his fake phpBB account can compromise the security of CryptDB in a number of ways:

Locating a Target. In phpBB, the user names can be seen on the forums. For a specific target the attackers try to login with the target name a few times. Since they do not have the password they will fail. However on the database the login queries over the same row will be noticed by the attackers. With this notification, they can locate the position of the target in the phpbb_users table.

Gleaning User Information. With this simple attack the mDBA can view all sorts of sensitive information, e.g. e-mail, last visit, registration date, website, time zone etc. The mDBA copies encrypted fields from other users records into the fake account's corresponding record fields. In this process, the username, the user_password, and IV of user_password fields are not altered. Through the fake account, the mDBA can then retrieve other user's information using the account info pages of the phpBB site. The proxy server just decrypts the query result as usual. In a sense we are using the proxy server as a decryption oracle.

Account Hijacking. The mDBA can overwrite the password and IV fields, i.e. user_password and IV, of other user's records, with his own fields taken from the records of his fake account. After this, the mDBA can login into phpBB as any other user where the username will be visible on the posts in the forums. The attack can also be modified to hijack a specific user's account. The mDBA can copy into all, and then login as target account, determine the specific row on which the login took place (in the log) and restore the remaining rows.

Privilege Escalation. In a typical installation of phpBB, the second row in the users table always belongs to the Administrator. The mDBA can use this information to raise the privilege level of his fake account to that of an phpBB administrator. For this he copies the user_type column from the administrator record into his fake accounts respective column. The mDBA also needs to set the user_permissions field to blank. It can be extracted from a previous user type change in the binary log or any other well known empty fields. After this simple task, the mDBA logs back into his fake phpBB account but in administrator mode. The changes CryptDB tables go through, before and after this attack are shown in Tables III and IV, respectively. Table IV shows the state of the system, after the attacker copied his password hash and salt into other users fields and managed to become an administrator.

The attacker, once become an administrator, can access all the restricted forums by adjusting the previalges. Since the phpBB is a dynamic web application, after the adjustments, the system must supply them the password to access the forums.

B. Attacking HotCRP

Essentially the same attack can be used to target HotCRP. An unsecured HotCRP executes a read query for a login which is

```
SELECT ContactInfo.* FROM ContactInfo
WHERE email='test@test.com'
```

As before the mDBA recovers the encrypted ID on the cloud server upon which he can repeat the same attacks, i.e. stealing other users information, account hijacking, privilege escalation, etc.

IV. Experimental Validation of Proposed Attacks

Unfortunately CryptDB with multiple principles is not yet made available to the public. We want to perform these attacks on a phpBB secured by CryptDB. Consequently, the code released for phpBB is not functional. Therefore, we used MySQL's sample database to validate our attacks. In our implementation, we set every column in the tables to the highest level of onions and left no column in plaintext form. Clearly, to execute a login query, the name column must be in the DET layer. To see that even under a JOIN operation that our attack works, we have added a password table into this database, and written a simple login system on a webpage to display the personal information and salaries. We developed some MySQL scripts to swap the columns, e.g. see Listing 1 which is used to swap personal information. The double run of this procedure leaves no traces in the table and attackers can clearly delete the necessary logs. After the mDBA logs into the site and the the row for the mDBA is identified on the cloud server. The mDBA runs this script on any other user's row. Note that, the names are under DET encryption. After the execution of this procedure, the mDBA refreshes his personal information page to access the leaked information.

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost`
    PROCEDURE `NAMESWAP`
     (IN emp1 VARCHAR(100) , IN emp2
        VARCHAR(100) )
UPDATE table_MJBMWEKXOD
 SET table_MJBMWEKXOD.EMKBCPSXBHoDET =
     (CASE WHEN
     table_MJBMWEKXOD.EMKBCPSXBHoDET =
     emp1 THEN emp2
 WHEN table_MJBMWEKXOD.EMKBCPSXBHoDET =
     emp2 THEN emp1 END)
 WHERE table_MJBMWEKXOD.EMKBCPSXBHoDET
     IN (emp2, emp1)
```

Listing 1. MySQL script that swaps a user's personal information to DBAs account

When we turn our attention to phpBB, the queries tell us that the necessary column that RND layer must be removed from users table is username_clean and this is for executing the login point query with the following clause.

```
... WHERE username_clean = '0x784EB1A'
```

V. Countermeasures to Tampering Attacks

In the previous section, we have seen that a DBA can cause great harm with unusually simple attacks as soon as we allow go beyond the honest-but-curious model allow the DBA tamper with the database even slightly.

CryptDB is designed to ensure the confidentiality of databases. Ensuring confidentiality is not sufficient to establish a secure system. Malicious administrators can alter the results of queries by deleting, inserting, copying or swapping data in the database as discussed above. To prevent these kinds of attacks, a database system regardless of whether the rows are encrypted or not, needs to ensure data and query integrity as well as the integrity of the query response. Even further, the authenticity and freshness of these also need to be ensured. We would like to mention that the countermeasures will significantly decrease the performance of the CryptDB.

There are a number of techniques in the literature that can provide this kind of facility. Merkle, in his pioneering work, proposed to arrange hashes values, which may be used to construct integrity checks, into a binary tree structure [10]. The verifier stores only the signed root. To see the membership problem, the claimer must give its sibling's hash and all other second winners. Devanbu uses balanced and I/O efficient B-trees, [6]. His approach is an extension of Merkle trees in a single dimension. Hacıgümüş et al [7] proposes a scheme to ensure row integrity using hash functions. The scheme uses buckets to show the query result integrity. Narasimha and Tsudik proposed Digital Signature Aggregation and Chaining to provide integrity, completeness and authentication [12]. Their idea is based on hash chaining and signing. Their hash chaining includes immediate predecessor row among any dimension (column) that requires the order relation on dimensions. Mykletun et al. uses condensed RSA signature scheme, which allows aggregation of signature into one signature for single signer [11]. Boneh et al. proposed multiple signers for aggregate signatures [5].

VI. More Powerful Attacks

In Section III, we demonstrated that the lack of an integrity exposes CryptDB attacks on multi-user databases. Single user databases do still have security vulnerabilities such as data deletion, replay, fork etc. We did not give any scenarios for these attacks. These problems may be solved by ensuring data integrity, authenticity, and freshness [6], [12]. However, here we show that CryptDB will remain to be vulnerable to more powerful attacks **on multi-user databases**.

A. Query Compilation Attack

In CryptDB the queries are visible to database administrators. If the integrity of a query is not protected, it can bring about serious privilege violations. We demonstrate how

the lack of *query integrity* can result in severe information leakages.

In this attack, The database by it's schema has security levels to access the information. A user with security level 3, cannot access all the information of his target that requires level 1 privilege. To access this information he contacts some attackers or mDBA to help him. Here we assume that online attackers have full access to a cloud server hosting a CryptDB protected database (or simply by the mDBA). Furthermore, we assume that the integrity protection and freshness shortcomings of CryptDB have been resolved.

Victim Identification. The first step in the attack is to recover the victim's encrypted CryptDB ID. User issues queries that retrieves non-confidential information of his target through the application server. Simultaneously, the mDBA checks the log on the cloud server, identifies the his query, and extracts the row and the ID of the target in the CryptDB secured database.

Collecting Words of Interest. In the next step the user makes requests, e.g. through a search query, for words of interest using his fake account. The application server issues selection queries based on his security level, as in the following code fragment.

```
... WHERE security_level >= 3
     AND
   incidents LIKE '%WORD%';
```

Note that application server prevents the search below security level 3. Hence, the queries do not result in any useful responses about the victim user which is assumed to be at a privilege level. However, in the cloud server log the mDBA is now able to pick up the encrypted versions of the words used in the queries. In a sense, the mDBA managed to use the proxy server as an encryption oracle.

Modified Search Query. At this point the mDBA has encrypted versions of the words of interest and the victim's CryptDB ID. He can now forge valid queries and execute them on the cloud server. For example, if one of the words was 'DRUG', he may execute a search query by calling the UDF to learn whether there is a match in the target victim's row.

```
...  WHERE ID = 'target'
   incidents LIKE '%DRUG%';
```

If there is match this means that they identified one word. This attack can be performed until a significant portion of the words appearing in the victim's rows are determined. A basic understanding of the organization that owns the database, e.g. law enforcement, medical, or financial records, will significantly simplify the process.

Enrolling integrity checks or performing authentication checks will not prevent this attack, since the attacker does not need any decrypted values. The mDBA and users always have access to the query interface which is sufficient to carry out the attack. To counter this attack, one solution is to keep every column under RND which does not allow any comparisons on the data. Another solution is to embed access control information into the fields, e.g. append all data in the database with a prefix according to the security level like 1_drug, 2_drug etc. This approach will restrict searches to result in a match only at the same security level.

B. Frequency Analysis Attacks

Here we give a simple example of a frequency attack applied on a CryptDB protected database. The attack falls under Threat-1 in the CryptDB security model, since no malicious modification is needed and the adversary remains passive. In general, frequency attacks are dismissed as to require too many samples to build a useful statistical profile. However, here we show that when the search space is small it is rather easy to narrow down the options and gain useful information as long as what we are searching for is deterministically encrypted.

visit_id	patient_id	visit_hour	visit_date
0x593	**0x345**	0x6134BD	0x6434
...	...	...	...
0x355	0x53B	0cEE542	0x9457
0c522	**0x345**	0x447E3	0xEED4

treatment_id	patient_id	treatment
...	...	...
0x6341	0x53B	0x645342
0x55BA	**0x345**	0xBBAE21
...	...	...

Table V
VISIT (UP) AND TREATMENT (DOWN) TABLES

To illustrate this attack, we consider a more concrete example. Assume that a hospital database residing on a cloud server is secured using CryptDB. Further, the integrity and authenticity problems of CryptDB resolved. Assume a notable person, e.g. the Governor of the state, visits the hospital on day x to have an examination. The real purpose for this visit is not publicized. This is followed by another visit by the Governor a few days later, say y. With this information, the sDBA (semi-honest DBA) examines the logs of the CryptDB database. He compares all the insert logs of day x and day y and computes their intersection on the ID column, see Visit table in Table V. The intersection will give the encrypted CryptDB IDs of patients who came in for a visit to the hospital on both days x and y. Only very few patients will fall into the intersection besides the Governor. With a few visits the sDBA will be able to identify the ID of the Governor in the CryptDB protected visitors table.

To support the execution of queries with a JOIN, all of the ID's have a deterministically encrypted value in the

all of the CryptDB protected tables. With the help of this ID, the sDBA is able to locate the Governor's rows in the table keeping treatment information, see Treatment table in Table V. If the treatment table is secured by deterministic encryption, further frequency analysis focusing on the specific treatment is possible.

Some notes on this attack. This attack does not require to breaking into the client's computer, or into the application or the proxy servers. It passive access to cloud server to recover the tables and the binary log. The binary log is the key in this attack. The binary log in general requires for recovery keeping track of write transactions, and database replications. Disabling the binary log is an option to prevent this attack. There is another way to realize this attack that will work even with the log disabled. For this, the mDBA takes snapshots of the database at reasonably chosen time intervals. By checking for updates he can recover newly updated IDs. This attack will work perfectly with slowly updated databases.

VII. Conclusions

In this work, we analyzed the prospects of securing web applications running CryptDB protected databases on cloud servers under realistic adversarial assumptions. We target web applications that run CryptDB in the multi-user setting. Therefore all queries and data are relayed through a trusted proxy server which also maintains the master key. As it turns out that online attacker or malicious database administrator by tampering with the integrity of the CryptDB protected database entries, which are communicated between the database server and the proxy server, can recover sensitive information of other users through any user account on the web application. Even worse, the administrator can easily escalate the privilege level of his web application account to that of an administrator. What is striking is that these attacks are possible without attacking the proxy and the web application server in any way. The attacks manage to extract information and escalate privilege without recovering the mastering key stored on the proxy server. Also, in our all attacks, OPE, the weakest encryption scheme of CryptDB, is not targeted.

To mitigate the proposed attacks, we mentioned a number of countermeasures from the literature to protect the integrity of the CryptDB database and thereby render the malicious DBA to the equivalent of a passive attacker. However, we show that even when the integrity of the database is protected, by manipulating the queries with the aid if a simple user account on the web application with no additional privileges, the online attackers or DBA can scan the CryptDB protected database for sensitive information on specific victims by tampering with the queries. Finally, we demonstrated with a simple example, that frequency analysis attacks cannot be taken lightly and deserve more serious attention.

References

[1] PhpBB : Free, open source bulletin board software. http://www.phpbb.com/.

[2] Agrawal, Kiernan, Srikant, and Xu. Order preserving encryption for numeric data. In *SIGMODIC*, 2004.

[3] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. *In Advances in Cryptology (CRYPTO)*, page 578595, Aug. 2011.

[4] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. *IACR Cryptology ePrint Archive*, 2012:624, 2012.

[5] Boneh, Gentry, Lynn, and Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2003.

[6] Premkumar T. Devanbu, Michael Gertz, Charles U. Martel, and Stuart G. Stubblebine. Authentic third-party data publication. In *DBSec*, volume 201, pages 101–112. Kluwer, 2000.

[7] H. Hacıgümüş, B.R. Iyer, and S. Mehrotra. Encrypted database integrity in database service provider model. In *Certification and Security in E-Services*, volume 255, pages 165–174. Kluwer, 2002.

[8] Eddie Kohler. Hot crap! In *WOWCS*. USENIX Association, 2008.

[9] Vladimir Kolesnikov and Abdullatif Shikfa. On the limits of privacy provided by order-preserving encryption. *Bell Labs Technical Journal*, 17(3):135–146, 2012.

[10] Merkle. Protocols for public key cryptosystems. In *SIMMONS: Secure Comm. and Asymmetric Cryptosystems*, 1982.

[11] Einar Mykletun, Maithili Narasimha, and Gene Tsudik. Authentication and integrity in outsourced databases. *ACM Transactions on Storage*, 2(2):107–138, May 2006.

[12] M. Narasimha and G. Gene. DSAC: integrity for outsourced databases with signature aggregation and chaining. In *Proceedings of the 2005 ACM CIKM*, pages 235–236, 2005.

[13] Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1999.

[14] R.A. Popa, C.M.S. Redfield, Z. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *23rd SOSP'11*, pages 85–100, 2011.

[15] R.A. Popa, C.M.S. Redfield, Z. Zeldovich, and H. Balakrishnan. CryptDB: processing queries on an encrypted database. *Commun. ACM*, 55(9):103–111, 2012.

[16] Raluca A. Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. *IACR Cryptology ePrint Archive*, 2013:129, 2013.

[17] Song, Wagner, and Perrig. Practical techniques for searches on encrypted data. In *RSP: 21th IEEE Computer Society Symposium on Research in Security and Privacy*, 2000.