Setup

The setup for Always Encrypted (with secure enclaves) requires at least 2 servers – one running the SQL Server, and another running the Host Guardian Service (HGS) needed for enclave attestation. The SQL Server Management Studio (SSMS) can be run on the server hosting the SQL Server.

Encryption

Encryption is done on the AE-enabled window – we encrypted both the Ohio and HCUP/NIS-2019 databases using both types of encryptions (deterministic and randomized). For randomized encryption, we additionally indexed the targeted columns with their corresponding plaintext columns (for ease of checking). A sample command is shown below:

Randomized:

ALTER TABLE [dbo].[NIS\_2019\_1000000\_tagged]

ALTER COLUMN [Age\_in\_years\_at\_admission] [int]

ENCRYPTED WITH (COLUMN\_ENCRYPTION\_KEY = [CEK1], ENCRYPTION\_TYPE = Randomized, ALGORITHM = 'AEAD\_AES\_256\_CBC\_HMAC\_SHA\_256') NOT NULL

CREATE INDEX IX\_0 ON [NIS\_2019\_1000000\_tagged] ([Age\_in\_years\_at\_admission]) INCLUDE ([Age\_in\_years\_at\_admission\_plain]);

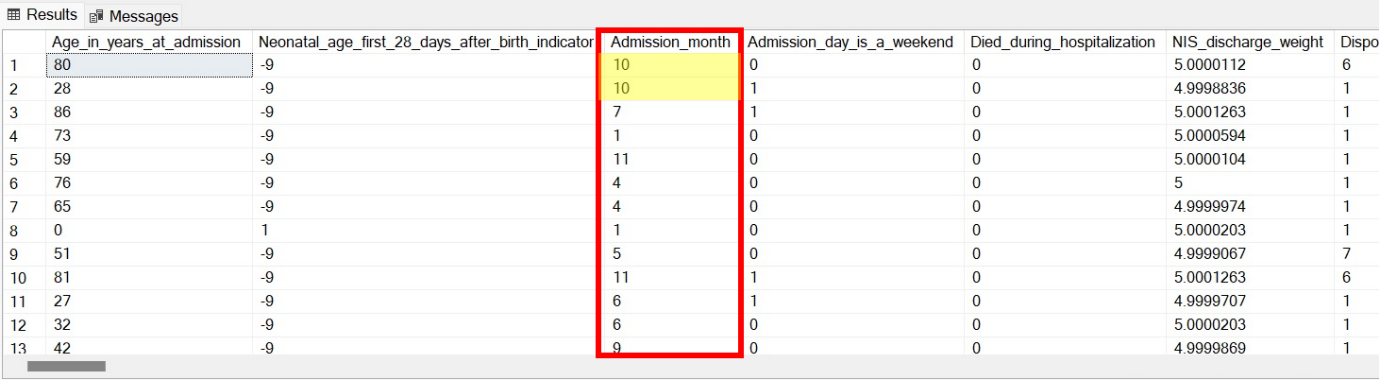
Deterministic:

ALTER TABLE [dbo].[NIS\_2019\_1000000\_tagged]

ALTER COLUMN [Age\_in\_years\_at\_admission] [int]

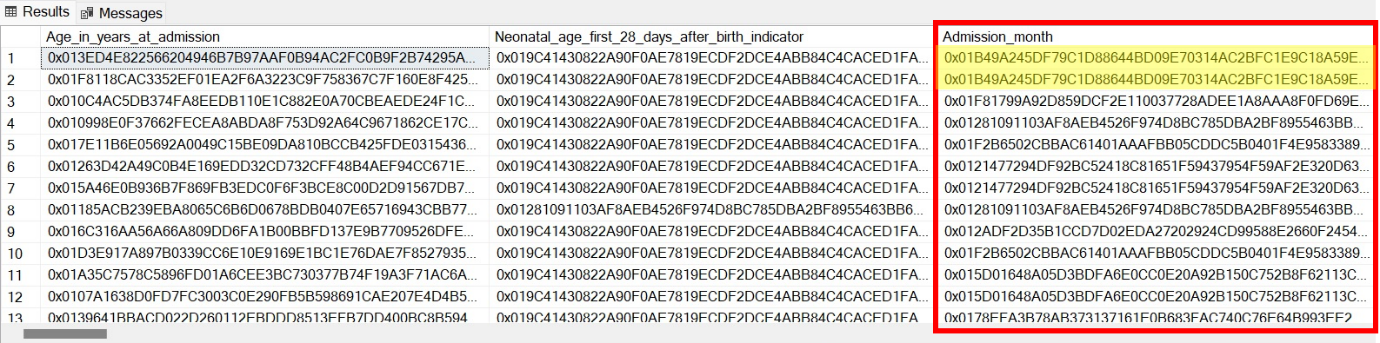
ENCRYPTED WITH (COLUMN\_ENCRYPTION\_KEY = [CEK1], ENCRYPTION\_TYPE = Deterministic, ALGORITHM = 'AEAD\_AES\_256\_CBC\_HMAC\_SHA\_256') NOT NULL

From the AE-enabled window, the columns are automatically decrypted.



From the AE-disabled window, we can confirm that the columns are encrypted.

(Deterministic)



(Randomized)



From the first screenshot, we can see that the first 2 rows in the “Admissions\_month” column are the same values – this translates to the same ciphertext in the deterministic case, but different ciphertexts in the randomized case.

Leakage Retrieval

For deterministic encryption, we can simply retrieve the full database, and perform frequency analysis on it (essentially the SF case).

Example command:

USE NIS2019;

SELECT \* FROM [NIS\_2019\_1000000\_tagged]

For randomized encryption, the key values in the index data structure (B+ trees) are encrypted and sorted based on their plaintext values. We can inspect the index contents in SSMS using DBCC (Database Console Command).

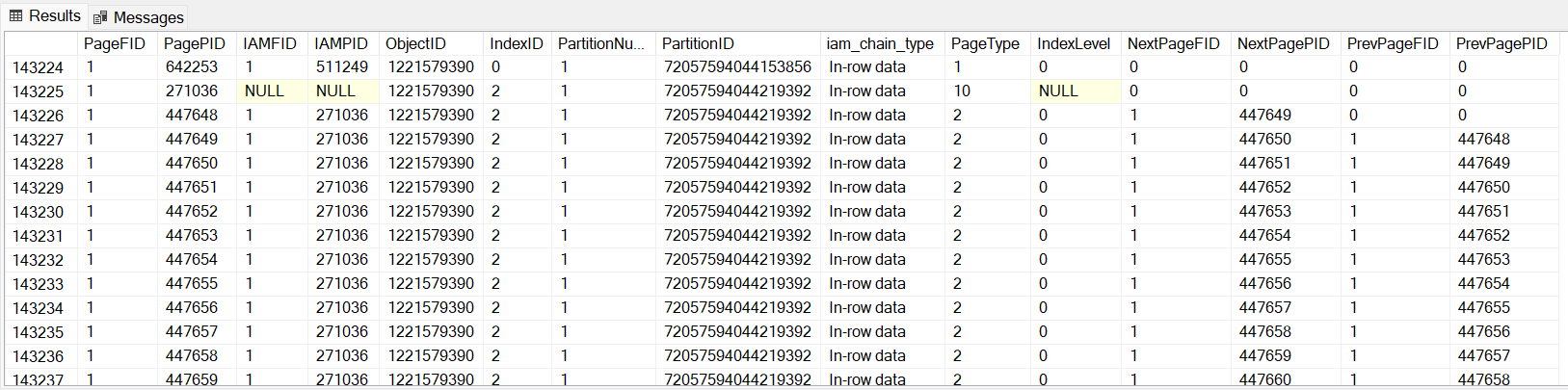
We perform the following steps.

Steps:

1. Retrieve indexes of all databases – use DBCC IND to list the File ID and Page ID of the pages of a table.

DBCC IND(<Database Name>, <Table Name>, -1)

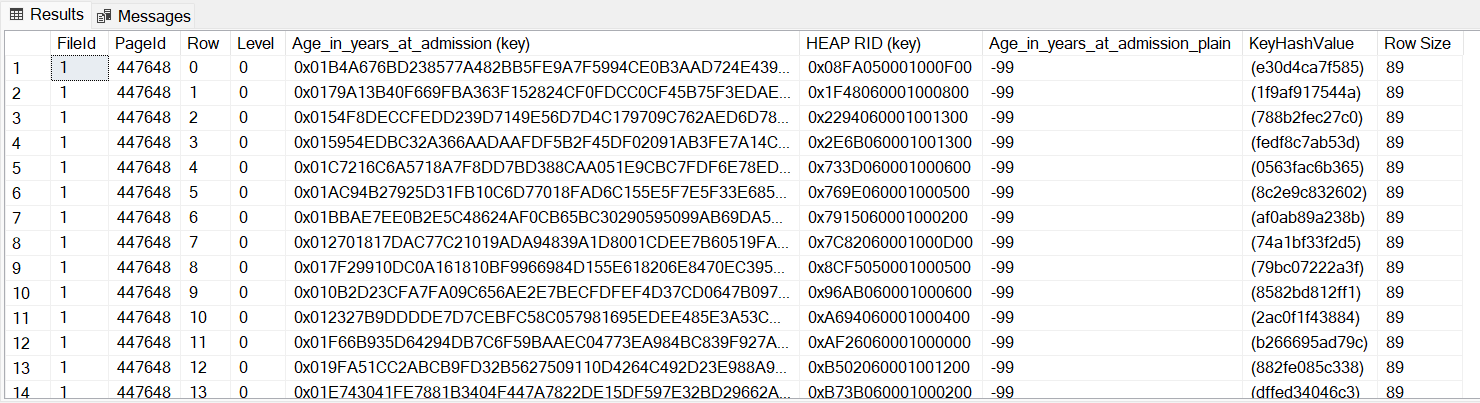
DBCC IND(NIS2019, NIS\_2019\_1000000\_tagged, -1)



1. The PageType column describes the content of the page. PageType==2 refers to an index page, which is what we’re interested in. Those with the same IndexID belong to the same index, and their order can be determined from PrevPagePID / NextPagePID. We can further inspect these index pages.

DBCC PAGE(<Database Name>, <PageFID>, <PagePID>, 3)

DBCC PAGE(NIS2019,1,447648,3)



From the first screenshot, we can see that the start of one of the indexes has PageID 447648, since its PrevPagePID == 0. Querying using DBCC PAGE gives us the corresponding ciphered column. We additionally included the plain column to show that ciphered column is indeed ordered by the plaintext values (here -99 is the minimum for this column). This plain column will not be present in actual attacks.

1. Query all index pages belonging to the same IndexID and order them using PrevPagePID / NextPagePID. This gives us the full ciphered column ordered by their plaintext values.
2. Perform the ORE-attack on this column. Use column distribution from an auxiliary database to create a sorted column with the same size as the ciphered database; this allows us to guess the corresponding ciphertext for each row.