Computing over encrypted data: homomorphic encryption and CryptDB
Homomorphic encryption

- Encryption $Enc$ is called homomorphic with respect to an operation $*$ if

- $Enc(x*y) = Enc(x)*Enc(y)$.

- That is given encrypted forms of $x$ and $y$, in order to compute encrypted form of $x*y$ one does not need to decrypt $Enc(x)$ and $Enc(y)$.

- Computations over encrypted values!
Partial vs Fully homomorphic schemes

- Partially homomorphic encryption: with respect just to one operation;
- RSA (unpadded) is homomorphic with respect to multiplication. Why?
- Fully homomorphic schemes:
  - With respect to multiplication and addition
  - Allow to perform arbitrary computations
  - Existence is by no means obvious
Breakthrough: FHE is possible!

- **Craig Gentry**: first fully homomorphic encryption scheme is announced by IBM on June 25, 2009.

- The scheme is impractical for many applications: ciphertext size and computation time increase sharply as one increases the security level. Key’s size is also an issue.
Recent developments

• New more efficient schemes and implementations since 2010, key size is reduced at least to 600Kb (~2016)
• HELib is an open source implementation (2013, new version 2018) (C++)
• More implementations available, including in R and Python;
• New library SEAL made available by Microsoft in 2018
• Still more work is needed to make it practical;
Potential applications

• Computations on not entirely trusted services (e.g. in the cloud):
  • Encrypt your computational task and send it to a remote server;
  • The server computes over encrypted data and returns an encrypted result;
  • Decrypt result;
• Pipeline processing without revealing intermediate data;
• …
CryptDB

- Similar idea in data processing:
  - To query encrypted SQL database without decrypting;
  - Selected fields can be encrypted;
  - Practical working prototype system: CryptDB,
- Low overhead: reducing throughput 15-25%
Onion-layered SQL-aware encryption

- All data in CrypDB can be encrypted using several layers of encryption;
  - Each layer may "release" some information about encrypted value
Querying in CryptDB

Before querying, depending on a query:
- some values in the query are encrypted;
- encryption layers in the database are adjusted (both steps are done by a proxy)

After the query execution: encrypted results are returned
- The proxy decrypts them and returns to the client the final result

Examples to consider:
SELECT * FROM Customers
SELECT * FROM Customers WHERE Country = ‘Mexico’
(SQL tutorial at w3schools.com/sql)
Developments here in the Department

• In two PhD projects:
  
  • CryptDB-like approach to graph DBs (Neo4j);
  • CryptDB-like approach do document-based DBs (MongoDB).