

DATA SECURITY AND PRIVACY

PROJECT

TEAM 17

Vennela Chadalavada – 811232432

Sreenath Reddy Kurukunda - 811211580

Rohitha Reddy Muppidi - 811232522

Introduction:

Order preserving encryption means it is one of the encryption scheme types which helps to store the order of values whenever the encryption is done. For suppose 2 plain text values are arranged in order then their related ciphertexts will also be arranged in the same order. Mainly these will be utilized scenarios wherever the order of data maintenance is more necessary like operations like sorting or searching etc.

The main theme for order-preserving encryption is to convert the plaintext values into ciphertexts so that the order of the values will be stored. This make sure that on the data which is encrypted, it can perform comparisons and operations without any necessary of decrypting it first.

Here I am going to give an explanation how order-preserving encryption will work:

1.Mapping Function: Order preserving encryption (OPE) will utilizes a mapping function to convert the values in plaintext to ciphertexts when storing the order. This mapping function will take the input as plaintext and gives the output as ciphertext that is mathematically relates the value which is given input. The mapping function is determined in a way that for suppose if $a < b$, then $f(a) < f(b)$

Here $a, b \rightarrow$ plaintext values

$f \rightarrow$ mapping function

2.Encryption process: we will apply the mapping function whenever we want the plaintext to be encrypted makes to generate ciphertext accordingly. The generated ciphertext will be arranged in an order based on the other ciphertexts, which will be arranged based on the original plaintext values.

3.Decryption: Whenever there is any retrieval or computation at that time by utilizing the same mapping function the ciphertext will be compared or processed to store the plaintext values order. However, Decryption process will not share the original values of plaintext but gives only their order.

Choosing an OPE Scheme:

For this implementation, we chose the OPE library (e.g., **pyope**), which provides a straightforward API for encrypting and decrypting data while preserving order. The library is assumed to handle integer values, necessitating the conversion of floating-point numbers to integers.

Design and Implementation:

The integration of OPE into the database system involved the following steps:

a. Database Setup (db_setup.py):

- **Weight Encryption:** The 'Weight' values, originally in floating-point format, are scaled to integers by multiplying with a factor of 100. This scaling ensures that the precision is maintained when converting to integers.
- **Data Insertion:** The scaled 'Weight' values are then encrypted using the OPE cipher before being inserted into the database.

b. Main Application (main.py):

- **Data Addition UI (add_data_item_ui):** A UI form allows users to input new data, including the 'Weight' attribute. The weight is scaled, encrypted, and then stored in the database.
- **Data Retrieval and Display (fetch_and_display_data):** When retrieving and displaying data, the encrypted 'Weight' is decrypted and rescaled back to its original floating-point format. This ensures that users see the actual weight values, not their encrypted forms.

c. Encryption and Decryption:

- The OPE cipher is used to encrypt and decrypt the 'Weight' values, preserving the order of these values for efficient range querying.

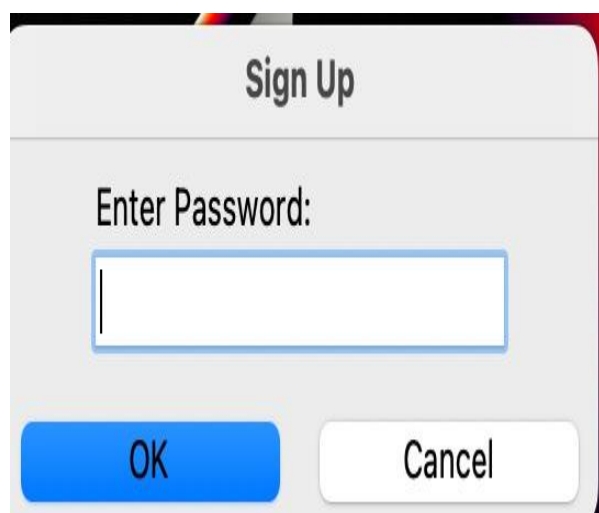
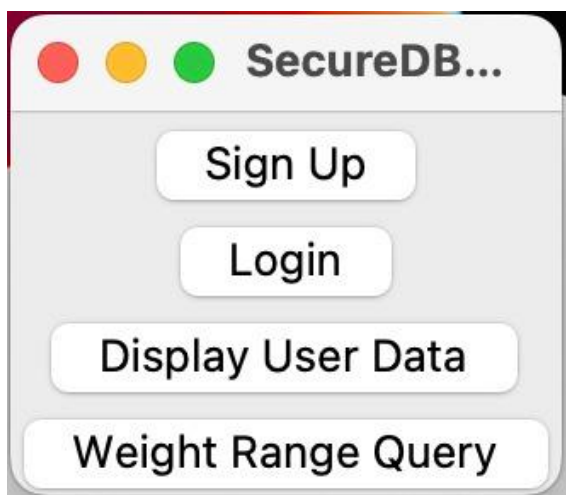
d. Error Handling:

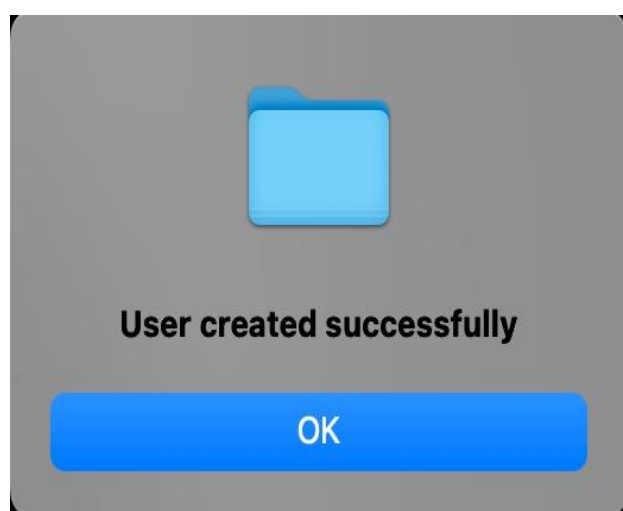
- The system includes error handling to manage non-numeric inputs and database errors.

Outputs:

Sign Up:

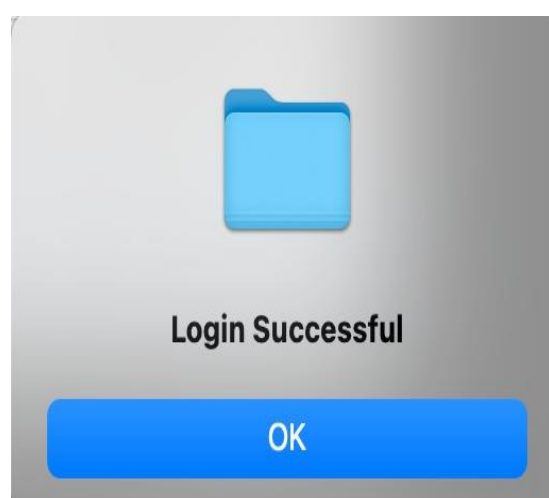
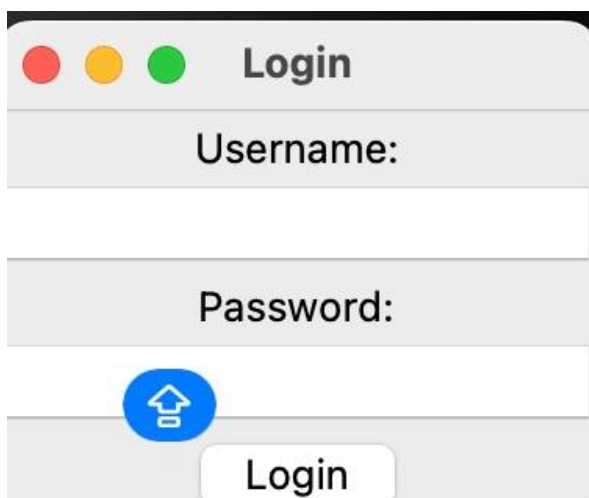
Firstly, the user needs to enter username, password, and group either H or R then successfully the user will be created.





Login:

Enter the username and password if the credentials are correct then user will login successfully.



Query Data:


Users should enter user ID and username if there is any data found it will display as shown below otherwise it will display no data found for this ID.

Query Data

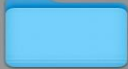
Enter User ID:

Query Data

Enter Username:




No data found for this ID



**first_name: ZXNLlrvOSz
last_name: nqMVgaUmVo
gender: 1
age: 71
weight: 57.32
height: 1.05
health_history:
jELPGqmhxRkFmtCEExSZ**


Weight Range Query:

If we enter minimum weight with highest age and maximum weight with lowest age, then it automatically prompts an alert message.

 **Weight Range Query**

Minimum Weight

Maximum Weight



Alert

If we have entered minimum weight and maximum weight correctly then it displays the data within that range.



Weight Range Query

Minimum Weight 25

Maximum Weight 50

Submit

(7, 'wdaEcVFEWB', 'VmVxpNdzv', 0, 0.16, 3593.0, 1.31)
(14, 'xJqdJCBOin', 'XpwGULajJv', 0, 0.68, 3720.0, 1.4)
(16, 'CPTFIjURtV', 'elkVtPJvnu', 1, 0.98, 3132.0, 1.33)
(20, 'LZUstSudmF', 'BijielVQCg', 1, 0.43, 4822.0, 1.91)
(26, 'CiltdoGWDL', 'AlpaXiKvbS', 0, 0.58, 3490.0, 1.7)
(28, 'ZcufBXfvof', 'xLABdFTrtv', 1, 0.32, 4196.0, 1.22)
(31, 'TBmHOWxrtM', 'FDNCAYMBWK', 1, 0.17, 4592.0, 1.62)
(32, 'MdTMCTmzwl', 'czZdsqZgwK', 0, 0.27, 4148.0, 1.48)
(33, 'cJyTJNNEJF', 'WgYkQrknfz', 0, 0.63, 3146.0, 1.03)
(34, 'nxPPwtoEal', 'MIRibjOOwq', 1, 0.13, 3424.0, 1.68)
(38, 'QAnZuoFlvr', 'NwjdvqgdHP', 1, 0.92, 3726.0, 1.19)
(39, 'opxlunhKyD', 'iVEEJZciHc', 1, 0.4, 4992.0, 1.83)
(40, 'UKRznTGLiU', 'EqmwZFtFCP', 0, 0.21, 3136.0, 1.23)
(43, 'xcXGnsalBG', 'PgiZTOXQPH', 1, 0.31, 3104.0, 1.39)
(46, 'KSMfKmxHgm', 'LKXWcpCtTG', 0, 0.5, 3481.0, 1.74)
(48, 'jTSnrDCsMv', 'wkMVhfqAcY', 1, 0.07, 4193.0, 1.47)
(55, 'kiAoMhNUWS', 'xZxUAXCcTA', 0, 0.43, 4189.0, 1.87)
(56, 'OUIJWvzQThm', 'JkXzHDIAMe', 1, 0.6, 3249.0, 1.72)
(60, 'igsYRvvpjW', 'eiPdmrhymF', 1, 0.88, 4476.0, 1.77)
(64, 'BROoghPBUX', 'hjkJEuBgqI', 0, 0.59, 4577.0, 1.01)
(69, 'OGMdkPlzcx', 'RcEsePMblh', 1, 0.82, 4715.0, 1.19)
(72, 'FHhzoKzsDv', 'RPujzuQVqJ', 0, 0.43, 4804.0, 1.75)
(76, 'mopiifggMm', 'IMnBRUBuZA', 0, 0.21, 3136.0, 1.63)
(77, 'uSDGbgcqRN', 'VTvEnwFsRj', 0, 0.75, 4900.0, 1.66)
(78, 'oEdAJnyUxl', 'amMTswRnDZ', 0, 0.57, 4316.0, 1.87)
(80, 'JUtFiGUwhd', 'ddYoNOqeUX', 1, 0.06, 3821.0, 1.24)
(81, 'cKyJvRBuVj', 'iieUglKcSA', 1, 0.04, 3361.0, 1.03)
(85, 'wljNalfrn', 'hiEPBjeedg', 1, 0.15, 4845.0, 1.21)
(86, 'CoRrLbzCsb', 'chbrwHqYMQ', 1, 0.19, 3112.0, 1.39)
(87, 'hNeLaKwjKg', 'iVLaBMyWNO', 1, 0.9, 4209.0, 1.65)
(88, 'kDLBIQyNnN', 'AFHrmvvcfT', 0, 0.58, 3392.0, 1.4)
(90, 'DIRrHumJUG', 'zdPqqYbev', 1, 0.8, 4962.0, 1.43)
(93, 'xTMPohtXp', 'duvLWFinUt', 0, 0.04, 4507.0, 1.82)
(96, 'wrDzmYuMwz', 'qQVODHJOgU', 1, 0.37, 4029.0, 1.01)
(100, 'iVWYLaNuec', 'lgunbaVtNs', 0, 0.23, 3035.0, 1.4)
(101, 'OkinaJyebS', 'GpalEJbAFz', 0, 0.65, 3308.0, 1.5)
(102, 'HuKAqzoCCn', 'cvaZMLJjDS', 0, 0.56, 3707.0, 1.26)
(104, 'IDRANfrIRZ', 'uZVgVGcDWM', 1, 0.38, 3902.0, 1.77)
(106, 'UzZjBelPUR', 'oaBJrUkrTI', 0, 0.56, 3685.0, 1.17)
(110, 'sMFavEbTbQ', 'QbjuOSDXhx', 0, 0.36, 3497.0, 1.63)
(113, 'VePBMgZwOg', 'sOgyNhKvmY', 1, 0.4, 3919.0, 1.84)
(114, 'UcfvhezOfk', 'ZjyusZYWRZ', 0, 0.8, 4126.0, 1.74)
(119, 'MpKySLhBWc', 'wTMGQpPzVP', 1, 0.97, 4306.0, 1.98)
(124, 'rpAuaeaNFb', 'XdVPrZLxJW', 0, 0.17, 4070.0, 1.06)
(125, 'ZPOhurhPWT', 'hONTelHwud', 1, 0.62, 3457.0, 1.85)
(126, 'oTtiOXfHci', 'HquiUwGdki', 1, 0.19, 3474.0, 1.58)

Challenges and Limitations

- a. **Precision Handling:** Converting floating-point numbers to integers for OPE was a challenge. We mitigated this by scaling the numbers, which might not be ideal for all precision requirements.
- b. **Security Considerations:** While OPE preserves order, it can potentially leak information about the magnitude of the values.
- c. **Performance:** The encryption and decryption process may introduce latency, especially for large datasets.

Conclusion:

The integration of OPE into the database system successfully enables the encryption of the 'Weight' attribute while allowing range queries. This implementation demonstrates a practical application of OPE in preserving data confidentiality without compromising the functionality of the database system.