### **IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**Modules:**

**1) Notations**

**2) Symmetric-key Encryption (SKE)**

**3) Homomorphic Encryption (HE)**

**4) Supporting Index Dynamics**

**1) Notations**

In cloud-based database systems, the data owner outsources a large-scale collection of data files d = (d1.......... d#d) to the remote database server in encrypted form c = (c1........ c#c). The data files d can represent text files or records in a relational database. The keyword universe (which contains all distinct keywords) extracted from d is denoted by w = (w1.........w#w). We use id(dj) to denote the identifier of data file dj and id(d) to denote the identifiers of all data files in d. Given a vector v, we refer its jth element as v[j] or vj . In our construction, we use div 2 f0; 1g#d to denote a data identifier vector (DIV), where the jth entry of div is 1 if dj is included; 0 otherwise. By this definition, given wi, we can use divi or divwi to denote all data files that contain wi (i.e., the corresponding entries in divi are 1’s). Different from most previous SSE constructions, we consider multikeyword query q = fw1.......w#qg. Hereafter, we use dq to represent the data files for both conjunction and disjunction logic queries.

**2) Symmetric-key Encryption (SKE)**

An SKE scheme is a set of three polynomial-time algorithms SKE = (Gen; Enc; Dec). Gen takes a security parameter k as input and outputs a secret key sk. Enc takes a key sk and message m as input and outputs a ciphertext c. Dec takes sk and c as input and outputs m iff c was encrypted under the key sk. Intuitively, an SKE scheme is secure against chosen-plaintext attacks (CPA) if the ciphertexts do not leak any useful information about the plaintext even to an adversary that can query an encryption oracle. Many common SKE schemes such as AES in counter mode satisfy this definition.

**3) Homomorphic Encryption (HE)**

HE allows certain computations to be carried out on ciphertexts to generate an encrypted result which matches the result of operations performed on the plaintext after being decrypted. Although fully homomorphic encryption (FHE) for arbitrary operations is prohibitively slow, homomorphic encryption for solely addition operations is practically efficient. In particular, we adopt the Paillier cryptosystem which is additively homomorphic in our construction.

**4) Supporting Index Dynamics**

In practice, supporting efficient data dynamics is a natural requirement for full-fledged database systems. Both addition and removal of data files should be supported without either re-indexing the whole database from scratch or making use of generic and relatively expensive dynamization techniques. UpdateToken(K; dj): To add a new file dj , the client locally encrypts it as cj and constructs the sub-index for dj which has the same structure as . To this end, the client simply runs BuildIndex(K; dj) and sets a = (cj ; ). To delete an existing file dj , the client first determines its location, then generates d = (j : dj 2 d). Update(a=d; ; c): For addition, merge into the original index and cj into ciphertexts c, respectively. For deletion, set j-th entry of each bucket in to 0, and delete cj from c.