CHAPTER A

Some algorithms

The Algorithm 22 takes a one dimension chain hierarchy (named ODH) as input and produce an output hierarchy enabling 3-hop shortcut scheme (3-HS). In a 3-HS, distance between any two nodes in the hierarchy will be at most 3 edges (or hops).

Algorithm 22 3HS_Gen(ODH)

Input: A one-dimension hierarchy (ODH).

Output: Hierarchy with 3-HS.

1: Create a set of special nodes $S$ consisting of every $\sqrt{n}$th node in the graph.
   That is, initialize $S$ with $v_1$ and then add nodes $v_{j\sqrt{n}+1}$ for all $j$ such that $j\sqrt{n} \leq n$. If $v_n \notin S$, set $S = S \cup \{v_n\}$.

2: Insert new edges between the nodes in $S$ to form the transitive closure of the set.

3: For each node $v_i \notin S$, find $v_j \in S$ such that $j < i$ and $i < j + \sqrt{n}$. Insert an edge $(v_j, v_i)$ if it is not already present.

4: For each node $v_i \notin S$, find $v_j \in S$ such that $i < j$ and $j - \sqrt{n} < i$. Insert an edge $(v_i, v_j)$ if it is not already present.

5: return

Algorithm 23 creates a tree data structure in Step 1 corresponding to the input set $T$ of time intervals and user hierarchy $UH$. Then it assigns secret keys, public labels and public edge values to the tree data structure (represents subscription hierarchy) corresponding to each node in the given user hierarchy.
Algorithm 23 Gen($1^k, T, \mathcal{U}H$)

Input: Security parameter $1^k$, set $T$ of time intervals and a user hierarchy $\mathcal{U}H(V_U, E_U)$.

Output: It create a tree data structure and for each node in the user hierarchy it assigns different set of secret keys, public labels and public edge values to the tree data structure.

1: Create a root node $root$ for the data structure and run $DataStuctBuild(root, T)$. Let $G = (V, E)$ denote the tree data structure returned.

2: For each $v \in V$, randomly choose a secret key $k_w \in \{0, 1\}^k$ and an unique public label $l_w \in \{0, 1\}^k$ associated with each node $w$ in $D(v)$, $R(v)$, and $L(v)$.

3: For each $t \in T$, randomly choose a secret key $k_t \in \{0, 1\}^k$ and an unique public label $l_t \in \{0, 1\}^k$.

4: For each $v \in V_U$, randomly choose a secret key $k_v \in \{0, 1\}^k$.

5: For each node $u \in V_U$, perform the following:
   (a) For each $v \in V$, randomly choose a secret key $k_{u,w} \in \{0, 1\}^k$ associated with each node $w$ in $D(v)$, $R(v)$, and $L(v)$.
   (b) For each $v \in V$, construct public information about each edge in $D(v)$, $R(v)$, and $L(v)$ using the key derivation method.
   (c) For each $t \in T$, randomly choose a secret key $k_{u,t} \in \{0, 1\}^k$.

6: For each $t \in T$, compute public information to permit key derivation between nodes: for each edge $(u_1, u_2) \in E$ compute public information by setting $S_{u_1} = k_{u_1,t}$ and $S_{u_2} = k_{u_2,t}$ and using the key derivation method and public labels $l_{u_1}$ and $l_{u_2}$.

7: For each $t \in T$, let $V_t \subset V$ denote the set of nodes in $G$ access to which implies access to $t$. Then for each $V_t$, for each $v \in V_t$:
   (a) Find in $D(v)$ the node corresponding to the time interval $t$; call it $w$.
   (b) Create an edge from $w$ to $t$ by computing public information using enabling key $k_{w,t}$’s secret key $k_t$, public label $l_t$, and the key derivation method. Mark such an edge with the level of $v$ and type $D$.
   (c) Repeat (a) and (b) for $R(v)$ and $L(v)$, using types $R$ and $L$, respectively.

8: Let $K$ consist of the secret keys $k_t$ for each $t \in T$ and $Sec$ consist of the remaining secret keys $k_w$. Also let $Pub$ consist of $G$, all public labels (of the form $l_w$ and $l_t$), and public information about all edges generated above.

9: return