Secure Compression and Pattern Matching Based on Burrows-Wheeler Transform

Raffaele Ceruso Giovanni Leo

November 24, 2018

Protocollo

Il protocollo prevede le seguenti fasi:
Inizializzazione. Il client genera le strutture dati *c* e *occ*. *occ* generata nel seguente modo.

- *superblock*:
  scegli un primo p_1 leggermente pi grande di n = L2, un numero diverso da zero β_1 e scegli un generatore g_1 di $Z^*_{p_1}$. Il client mantiene $g_1\beta_1$ segreti. Per *superblock* i, memorizziamo quei dati nel $(g_1^i\beta_1)^{th} mod\ p_1$ (omettiamo $mod\ p_1$ nella parte restante) della tabella. Per le entries vuote inseriamo alcuni numeri casuali.

Protocollo

Il protocollo prevede le seguenti fasi:
Inizializzazione. Il client genera le strutture dati c e occ. occ generata nel seguente modo.

- *block*:
  scegli un primo p_2 che poco pi grande di n/L^2, un numero diverso da zero β_2 e scegli un generatore g_2 di $Z^*_{p_2}$. Il client mantiene $g_2\beta_2$ segreti. Per *block* i, memorizziamo quei dati nel $(g_2^i\beta_2)^{th} mod\ p_2$ (omettiamo $mod\ p_2$ nella parte restante) della tabella. Per le entries vuote inseriamo alcuni numeri casuali.

Protocollo

Il protocollo prevede le seguenti fasi:
Inizializzazione. Il client genera le strutture dati c e occ. occ generata nel seguente modo.

- $Hash\_value$:
  Invece di memorizzare immagini di MTF tabelle, memorizziamo il valore hash di $(i-1)^{th}$ block per il blocco $(i)^{th}$. Il valore hash di $(i-1)^{th}$ block memorizzato nella $(g_2^i \beta_2)^{th}$ entry della tabella $Hash_value$. Per le entries vuote inseriamo alcuni numeri casuali.

Protocollo

Il protocollo prevede le seguenti fasi:
Inizializzazione. Il client genera le strutture dati c e occ. occ generata nel seguente modo.

- *Block_inner*:
  La riga $Block\_inner(mtf[i], BZ_i, e, h - L * i)$ viene spostato su $Block\_inner(Hash\_value[g_2^i \beta_2], BZ_{g^i \beta_2}, e, \tau(key, h - L * i))$ dove $i = \lfloor h/L \rfloor$ e $\tau : Z_L \rightarrow Z_L$ una funzione di permutazione casuale.

Protocollo

Il protocollo prevede le seguenti fasi:
Inizializzazione. Il client genera le strutture dati c e occ. occ generata nel seguente modo.
Pertanto, per calcolare $occ(e, h)$, il client deve inviare un vettore di posizione $pv(h) = (g_1^{h/L^2}\beta_1, g_2^{h/L}\beta_2, \tau(key, h - [h/L] * L))$ al server. Cos abbiamo $occ(e, h) = superblock(e, g_1^{h/L^2}\beta_1) + block(e, g_2^{h/L}\beta_2) +$ block_inner(Hash_value$[g_2^{h/L}\beta_2,]$, $BZ_{(g2)^{h/L}\beta_2}, e, \tau(key, h - \lfloor h/L \rfloor * L)$
Confrontando il blocco interno qui con quello introdotto nella sezione II-B, sappiamo che $BZ_{\lfloor h/L \rfloor}$ viene spostato in $BZ_{g_2^{\lfloor h/L \rfloor}\beta_2}$