Information Flows in Encrypted Databases : Supplement

1. Motivating example

Figure 1 shows the partitioning that contains no explicit or implicit flows. This partitioning requires the column C_DATA to be encrypted, and consequently incurs an additional round trip to the client because the string concatenation operation must evaluated on the client.

```
--Server
CREATE PROCEDURE [dbo].[__Closure]
@c_w_id INT, @h_amount VARBINARY(2048), @c_last VARBINARY(256)
AS BEGIN
 SELECT @pubkey = ...
 UPDATE dbo.CUSTOMER
 SET @c_id = C_ID,
  @c_first =  C_FIRST,
  @c_credit = C_CREDIT,
  @c_balance = C_BALANCE =
   dbo.PaillierAdd(C_BALANCE, @h_amount, @pubkey)
 WHERE CUSTOMER.C_W_ID = @c_w_id
  AND CUSTOMER.C_LAST = @c_last;

 INSERT dbo.HISTORY (H_C_ID, H_C_BALANCE)
 VALUES (@c_id, @c_balance)

 SELECT @c_id AS N'@c_id',
  @c_first AS N'@c_first',
  @c_last AS N'@c_last',
  @c_credit AS N'@c_credit',
  @c_balance AS N'@c_balance'
END

-- Shell
CREATE PROCEDURE [dbo].[PAYMENT]
@c_w_id INT, @h_amount NUMERIC(6,2), @c_last CHAR(16)
AS BEGIN
 SELECT @key = ... // private key
 SELECT @pubkey = ... // public key for paillier
 SELECT
  @enc_h_amount =
   dbo.AEncrypt(@h_amount, @key, @pubkey),
  @enc_c_last = dbo.DEncrypt(@c_last, @key),

 BEGIN TRANSACTION
  EXEC [SERVER].[tpcc].dbo.__Closure1
   @enc_c_w_id, @c_d_id,
   @enc_c_amount, @enc_c_last,
   out @c_id, out @c_first, out @c_last,
   out @c_balance, out @c_credit

  if (@c_credit = 0x002057E9A8865AAA7D59DA69AD...)
   UPDATE [SERVER].dbo.CUSTOMER
   SET C_DATA = dbo.REncrypt(@h_amount + C_DATA)
   WHERE CUSTOMER.C_W_ID = @c_w_id
    AND CUSTOMER.C_LAST = @enc_c_last;

  SELECT @c_id AS @c_id,
   dbo.RDecrypt(@c_first, @key) AS @c_first,
   dbo.DDecrypt(@c_last, @key) AS @c_last,
   dbo.DDecrypt(@c_credit, @key) AS @c_credit,
   dbo.ADecrypt(@c_balance, @key, @pubkey) AS @c_balance,
 COMMIT TRANSACTION
END
```

Figure 1: A T-SQL procedure derived from TPC-C

2. Type inference

The type inference algorithm is described as a set of type rules.

$$\frac{\beta = \mathsf{FRESH}()}{\{\mathsf{CT} <: \beta\}, \{x : \mathsf{CT}\} \vdash x : \beta}\ \text{[CONST]}$$

$$\frac{\alpha, \beta = \mathsf{FRESH}()}{\{\alpha <: \beta\}, \{v : \alpha\} \vdash v : \beta}\ \text{[VAR]}$$

$$\frac{\alpha = E(t,c) \qquad \beta = \mathsf{FRESH}()}{\{\alpha <: \beta\}, \{v : \alpha\} \vdash t.c\ : \beta}\ \text{[COLUMN]}$$

$$\frac{\forall i \in (1..n)\ \Sigma_i, \Gamma_i \vdash e_i : \beta_i \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_j \wedge v : \beta \in \Gamma_k\}}{\cup_{i=1}^{n} \mathrm{S}(\Sigma_i), \cup_{i=1}^{n} \mathrm{S}(\Gamma_i) \vdash [e_1, ..., e_m] : [\mathrm{S}(\beta_1), ..., \mathrm{S}(\beta_n)]}\ \text{[REC]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \beta = \mathsf{FRESH}() \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2\})}{\begin{array}{c}\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\beta \in [\mathsf{CT}, \mathsf{CT}_{\mathsf{OPE}}, \mathsf{CT}_{\mathsf{DE}}, \mathsf{CT}_{\mathsf{RE}}]\} \cup \{\beta_1 \in [\mathsf{OPE}, \mathsf{OPE}_{\mathsf{lval}}, \mathsf{DE}, \mathsf{DE}_{\mathsf{lval}}, \mathsf{CT}, \mathsf{CT}_{\mathsf{OPE}}, \mathsf{CT}_{\mathsf{DE}}, \mathsf{CT}_{\mathsf{RE}}], \\ \mathrm{S}(\beta_1) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e_1 = e_2 : \beta\end{array}}\ \text{[EQUALS]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \beta = \mathsf{FRESH}() \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2\})}{\begin{array}{c}\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\beta \in [\mathsf{CT}, \mathsf{CT}_{\mathsf{OPE}}, \mathsf{CT}_{\mathsf{DE}}, \mathsf{CT}_{\mathsf{RE}}]\} \cup \{\beta_1 \in [\mathsf{OPE}, \mathsf{OPE}_{\mathsf{lval}}, \mathsf{CT}, \mathsf{CT}_{\mathsf{OPE}}, \mathsf{CT}_{\mathsf{DE}}, \mathsf{CT}_{\mathsf{RE}}], \\ \mathrm{S}(\beta_1) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e_1 < e_2 : \beta\end{array}}\ \text{[COMP]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \beta = \mathsf{FRESH}() \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2\})}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\beta_1 \in [\mathsf{AE}, \mathsf{AE}_{\mathsf{lval}}, \mathsf{CT}, \mathsf{CT}_{\mathsf{OPE}}, \mathsf{CT}_{\mathsf{DE}}, \mathsf{CT}_{\mathsf{RE}}], \mathrm{S}(\beta_1) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e_1 + e_2 : \beta}\ \text{[ADD]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \tau, \mu = \mathsf{FRESH}() \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2 \to \tau\})}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\tau <: \mu\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash fe : \mu}\ \text{[APPLY]}$$

$$\frac{\Sigma, \Gamma \cup \{v : \alpha\} \vdash e : \beta \qquad \tau = \mathsf{FRESH}()}{\Sigma \cup \{\alpha \to \beta <: \tau\}, \Gamma \vdash f(v).e : \tau}\ \text{[ABS]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \beta = \mathsf{FRESH}() \Sigma_{\mathrm{S}}, \Gamma_{\mathrm{S}} \vdash e_{\mathrm{S}} : \beta_{\mathrm{S}} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \mathsf{Cleartext}\})}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\mathrm{S}(\beta_2) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash \sigma_{e_1}(e_2) : \beta}\ \text{[SELECT]}$$

$$\frac{\forall i,\ \beta_i = E(T, c_i) \qquad \beta = \mathsf{FRESH}()}{\{[\beta_1, ..., \beta_n] <: \beta\}, \{\} \vdash: \pi_{n_1, ..., n_m}(e) : \beta}\ \text{[PROJECT]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2\}) \qquad \beta = \mathsf{FRESH}()}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{, \mathrm{S}(\beta_1) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e_1 \cup e_2 : \beta}\ \text{[UNION]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\beta_1, \beta_2\}) \qquad \beta = \mathsf{FRESH}()}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{, \mathrm{S}(\beta_1) <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e_1 \backslash e_2 : \beta}\ \text{[DIFF]}$$

$$\frac{\forall i \in (1,n),\ \beta_i = E(T_1, c_i) \qquad \forall j \in (1,m),\ \tau_j = E(T_2, c_j) \qquad \beta = \mathsf{FRESH}()}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{[\beta_1, ..., \beta_n, \tau_1, ..., \tau_m] <: \beta\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \vdash e1 \times e2 : \beta}\ \text{[PRODUCT]}$$

Figure 2: Type inference rules for expressions in λ_{SQL}

$$\frac{\Sigma_1, \Gamma_1 \cup \{e_1 : \alpha_1\} \vdash e_1 : \beta_1 \qquad \Sigma_2, \Gamma_2 \vdash e_2 : \beta_2 \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_1 \wedge v : \beta \in \Gamma_2\} \cup \{\alpha_1, \beta_1\})}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \{\mathrm{S}(\beta_2) <: \mathrm{S}(\beta_1)\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2), \mathrm{S}(\beta_1) \vdash e_1 := e_2}\,\text{[ASSIGN]}$$

$$\frac{\forall i,\ \beta_i = \mathsf{LVALTYPE}(E(t, c_i)) \qquad \Sigma, \Gamma \vdash e : \beta \qquad \gamma = \mathsf{FRESH}()}{\Sigma \cup \{\beta <: [\beta_1, ..., \beta_n] \cup \{\forall i \in [1, ..., n], \mathrm{S}(\beta_i) <: \gamma\}, \Gamma, \gamma \vdash: \mathsf{insert}_{n_1, ..., n_m}(t)\ e}\,\text{[INSERT]}$$

$$\frac{\forall i,\ \beta_i = \mathsf{LVALTYPE}(E(t, c_i)) \qquad \Sigma, \Gamma \vdash e : \beta \qquad \gamma = \mathsf{FRESH}()}{\Sigma \cup \{\beta <: [\beta_1, ..., \beta_n] \cup \{\forall i \in [1, ..., n], \mathrm{S}(\beta_i) <: \gamma\}, \Gamma, \gamma \vdash: \mathsf{delete}(t)\ e}\,\text{[DELETE]}$$

$$\frac{\forall i,\ \beta_i = \mathsf{LVALTYPE}(E(t, c_i)) \qquad \Sigma, \Gamma \vdash e : \beta \qquad \gamma = \mathsf{FRESH}()}{\Sigma \cup \{\beta <: [\beta_1, ..., \beta_n] \cup \{\forall i \in [1, ..., n], \mathrm{S}(\beta_i) <: \gamma\}, \Gamma, \gamma \vdash: \mathsf{update}_{n_1, ..., n_m}(t)\ e}\,\text{[UPDATE]}$$

$$\frac{\Sigma, \Gamma \vdash e : \beta \qquad \forall i \in [1, ..., m],\ \beta_i = \mathsf{PROJECT}(\beta, n_i) \qquad \gamma = \mathsf{FRESH}()}{\{\Sigma \cup \{\forall i \in [1, ..., n], \mathrm{S}(\beta_i) <: \gamma\}, \Gamma, \gamma \vdash: \mathsf{select}_{n_1, ..., n_m}(e)}\,\text{[SELECT]}$$

$$\frac{\Sigma_1, \Gamma_1 \vdash e_1 : \beta_1 \qquad \Sigma_1, \Gamma_1, \gamma_1 \vdash s_1 \qquad \Sigma_2, \Gamma_2, \gamma_2 \vdash s_2 \qquad \mathrm{S} = \mathsf{UNIFY}(\{\alpha, \beta \mid v : \alpha \in \Gamma_i \wedge v : \beta \in \Gamma_j\} \cup \{\gamma_1, \gamma_2\})}{\mathrm{S}(\Sigma_1) \cup \mathrm{S}(\Sigma_2) \cup \mathrm{S}(\Sigma_3) \cup \{\mathrm{S}(\beta_1) <: \mathrm{S}(\gamma_1)\}, \mathrm{S}(\Gamma_1) \cup \mathrm{S}(\Gamma_2) \cup \mathrm{S}(\Gamma_3), \mathrm{S}(\gamma_1) \vdash \mathsf{if}\ e_1\ s_1\ s_2}\,\text{[IF]}$$

Figure 3: Type inference rules for statements in λ_{SQL}