

Private-Key Hidden Vector Encryption with Key Confidentiality

C. Blundo, V. Iovino, G. Persiano

Dipartimento di Informatica ed Appl.
"Renato M. Capocelli"
Università di Salerno

SEDAN workshop on Searchable Encryption
Enschede, The Netherlands
October 7th, 2009

Outline

1 Hidden Vector Encryption

Outline

- 1 Hidden Vector Encryption
- 2 Key Security

Outline

- 1 Hidden Vector Encryption
- 2 Key Security
- 3 Partial Public Key Model

Outline

- 1 Hidden Vector Encryption
- 2 Key Security
- 3 Partial Public Key Model
- 4 Construction

Outline

- 1 Hidden Vector Encryption
- 2 Key Security
- 3 Partial Public Key Model
- 4 Construction
- 5 Private-Key Searchable Encryption

Predicate Encryption

Predicate Encryption

- Ciphertexts have attributes.
- Keys have attributes.
- There is a predicate \mathcal{P} and key K with pattern \vec{y} can decipher ciphertext Ct with attribute \vec{x} iff and only if $\mathcal{P}(\vec{x}, \vec{y}) = 1$.

Hidden Vector Encryption

Hidden Vector Encryption

- Ciphertext Ct is associated with *attribute* vector \vec{x} of length ℓ over alphabet Σ .
- Key K is associated with *pattern* vector \vec{y} of length ℓ over alphabet $\Sigma_{\star} = \Sigma \cup \{\star\}$.
- Predicate $\text{Match}(\vec{x}, \vec{y})$ which is true if and only if $\vec{x} = \langle x_1, \dots, x_{\ell} \rangle$ and $\vec{y} = \langle y_1, \dots, y_{\ell} \rangle$ agree in all positions i for which $y_i \neq \star$.

Hidden Vector Encryption

Hidden Vector Encryption

- Ciphertext Ct is associated with *attribute* vector \vec{x} of length ℓ over alphabet Σ .
- Key K is associated with *pattern* vector \vec{y} of length ℓ over alphabet $\Sigma_\star = \Sigma \cup \{\star\}$.
- Predicate $\text{Match}(\vec{x}, \vec{y})$ which is true if and only if $\vec{x} = \langle x_1, \dots, x_\ell \rangle$ and $\vec{y} = \langle y_1, \dots, y_\ell \rangle$ agree in all positions i for which $y_i \neq \star$.

If pattern vectors $\vec{y} \in \Sigma^\ell$ we have the original notion of **searchable encryption**.

Hidden Vector Encryption – The syntax

Hidden Vector Encryption

- 1 **Setup**($1^n, 1^\ell$) outputs the *public key* PK and the *secret key* SK.
- 2 **Encryption**(PK, \vec{x} , M) encrypts message M producing ciphertext Ct with attribute vector \vec{x} .
- 3 **GenToken**(SK, \vec{y}) outputs *key* $K_{\vec{y}}$.
- 4 if $\text{Match}(\vec{x}, \vec{y}) = 1$ then **Decrypt**(Ct, $K_{\vec{y}}$) returns M ;
otherwise returns a random element.

Hidden Vector Encryption – The syntax

Hidden Vector Encryption (Attribute Only)

- 1 **Setup** $(1^n, 1^\ell)$ outputs the *public key* PK and the *secret key* SK.
- 2 **Encryption** (PK, \vec{x}) outputs an *encrypted attribute vector* \tilde{X} .
- 3 **GenToken** (SK, \vec{y}) outputs *key* $K_{\vec{y}}$.
- 4 **Test** $(\tilde{X}, T_{\vec{y}})$ returns $\text{Match}(\vec{x}, \vec{y})$ with overwhelming probability.

Application scenario

Search on Encrypted Data

- 1 user has a huge amount of data;
 - ▶ tables in which each row has ℓ attributes and one value;
- 2 storage is done by a third party;
- 3 data is encrypted before storage;
- 4 to select all rows that satisfy pattern \vec{y} , give $K_{\vec{y}}$ to storage manager;

Semantic security

$\text{SemanticExp}_{\mathcal{A}}(1^n, 1^\ell)$

1. **Initialization Phase.** \mathcal{A} announces two challenge attribute vectors $\vec{z}_0, \vec{z}_1 \in \Sigma^\ell$.
2. **Key-Generation Phase.** \mathcal{C} computes $(\text{PK}, \text{SK}) \leftarrow \text{Setup}(1^n, 1^\ell)$.
PK is given to \mathcal{A} .
3. **Query Phase I.** \mathcal{A} can make any number of key queries.
 \mathcal{C} answers key queries only for patterns \vec{y} such that $\text{Match}(\vec{z}_0, \vec{y}) = \text{Match}(\vec{z}_1, \vec{y}) = 0$.
4. **Challenge construction.** \mathcal{C} chooses random $\eta \in \{0, 1\}$ and returns $\text{Encryption}(\text{PK}, \vec{z}_\eta)$ to \mathcal{A} .
5. **Query Phase II.** Identical to Query Phase I.
6. **Output phase.** \mathcal{A} returns η' .
If $\eta = \eta'$ then the experiments returns 1 else 0.

Known Constructions

Pairing

(symmetric version)

- multiplicative groups \mathbb{G} and \mathbb{G}_T ;
- non-degenerate pairing function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$;
 - ▶ for all $x \in \mathbb{G}$, $x \neq 1$, and $a, b \in \mathbb{Z}_p$,

$$e(x, x) \neq 1 \text{ and } e(x^a, x^b) = e(x, x)^{ab}.$$

Known Constructions

Pairing

(symmetric version)

- multiplicative groups \mathbb{G} and \mathbb{G}_T ;
- non-degenerate pairing function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$;
 - ▶ for all $x \in \mathbb{G}$, $x \neq 1$, and $a, b \in \mathbb{Z}_p$,

$$e(x, x) \neq 1 \text{ and } e(x^a, x^b) = e(x, x)^{ab}.$$

Constructions

- Boneh and Waters [TCC 07] gave a construction based on complexity assumption for pairing with composite order group;
- Iovino and P. [Pairing 08] gave a construction for **prime** order groups;

Key Security

Security threat

Storage manager knows all searches I have done.

Key Security

Security threat

Storage manager knows all searches I have done.

Token security is impossible for public key

- storage manager receives $K_{\vec{y}}$ and wants to check if $\text{Match}(\langle 1, \dots, 1 \rangle, \vec{y}) = 1$;
 - ▶ encrypt $\langle 1, \dots, 1 \rangle$ using \bar{K} ;
 - ▶ run **Test** to obtain the answer;

We should go private key!

Key Security

Security threat

Storage manager knows all searches I have done.

Token security is impossible for public key

- storage manager receives $K_{\vec{y}}$ and wants to check if $\text{Match}(\langle 1, \dots, 1 \rangle, \vec{y}) = 1$;
 - ▶ encrypt $\langle 1, \dots, 1 \rangle$ using \bar{K} ;
 - ▶ run **Test** to obtain the answer;

We should go private key!

Or maybe not....

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle$

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;
 - ▶ $\text{Pol} = \Sigma^\ell$

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;
 - ▶ $\text{Pol} = \Sigma^\ell \Rightarrow$ all vectors (public-key setting);

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;
 - ▶ $\text{Pol} = \Sigma^\ell \Rightarrow$ all vectors (public-key setting);
 - ▶ if any entry is \emptyset

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;
 - ▶ $\text{Pol} = \Sigma^\ell \Rightarrow$ all vectors (public-key setting);
 - ▶ if any entry is $\emptyset \Rightarrow$ no vector (private-key setting);

Partial Public Key Model

Key Policy

- public keys associated with policies;
- a policy Pol is a vector of subsets of Σ ;
- it encodes the set \mathbb{X}_{Pol} of attribute vectors that can be encrypted;
 - ▶ $\text{Pol} = \langle \{0, 1\}, \{0\}, \{0, 1\} \rangle \Rightarrow$ vectors with a 0 entry in position 2;
 - ▶ $\text{Pol} = \Sigma^\ell \Rightarrow$ all vectors (public-key setting);
 - ▶ if any entry is $\emptyset \Rightarrow$ no vector (private-key setting);

Hidden Vector Encryption

- 1 **Setup**($1^n, 1^\ell$) outputs the *secret key* SK.
- 2 **PPKeyGen**(SK, Pol) outputs the *partial public key* PPK_{Pol} .
- 3 **Encryption**($\text{PPK}_{\text{Pol}}, \vec{x}$) outputs *encrypted attribute vector* \tilde{X} for attribute vector $\vec{x} \in \mathbb{X}_{\text{Pol}}$.
- 4 **GenToken**(SK, \vec{y}) outputs *key* $K_{\vec{y}}$.
- 5 **Test**($\tilde{X}, T_{\vec{y}}$) returns $\text{Match}(\vec{x}, \vec{y})$ with overwhelming probability.

Semantic Security with Partial Public Keys

1. **Initialization Phase.** \mathcal{A} announces two challenge attribute vectors $\vec{z}_0, \vec{z}_1 \in \Sigma^\ell$ and policy $\text{Pol} \in (2^\Sigma)^\ell$.
2. **Key-Generation Phase.** \mathcal{C} computes $\text{SK} \leftarrow \text{Setup}(1^n, 1^\ell)$ and $\text{PPK}_{\text{Pol}} \leftarrow \text{PPKeyGen}(\text{SK}, \text{Pol})$.
 PPK_{Pol} is given to \mathcal{A} .
3. **Query Phase I.** \mathcal{A} can make any number of key queries.
 \mathcal{C} answers key queries only for patterns \vec{y} such that $\text{Match}(\vec{z}_0, \vec{y}) = \text{Match}(\vec{z}_1, \vec{y}) = 0$.
4. **Challenge construction.** \mathcal{C} chooses random $\eta \in \{0, 1\}$ and returns $\text{Encryption}(\text{SK}, \vec{z}_\eta)$.
5. **Query Phase II.** Identical to Query Phase I.
6. **Output phase.** \mathcal{A} returns η' .
If $\eta = \eta'$ then the experiments returns 1 else 0.

Token Security with Partial Public Keys

- Initialization Phase.** \mathcal{A} announces $\vec{y}_0, \vec{y}_1 \in \Sigma_\star^\ell$ **with \star in the same positions** and a policy Pol such that
$$\vec{x} \in \mathbb{X}_{\text{Pol}} \Rightarrow \text{Match}(\vec{x}, \vec{y}_0) = \text{Match}(\vec{x}, \vec{y}_1) = 0.$$
- Key-Generation Phase.** \mathcal{C} computes $\text{SK} \leftarrow \text{Setup}(1^n, 1^\ell)$ and $\text{PPK}_{\text{Pol}} \leftarrow \text{PPKeyGen}(\text{SK}, \text{Pol})$.
 PPK_{Pol} is given to \mathcal{A} .
- Query Phase I.** \mathcal{A} can make any number of key queries.
 \mathcal{A} gets $\text{GenToken}(\text{SK}, \vec{y})$.
- Challenge construction.** η is chosen at random from $\{0, 1\}$ and receives $\text{GenToken}(\text{SK}, \vec{y}_\eta)$.
- Query Phase II.** Identical to Query Phase I.
- Output phase.** \mathcal{A} returns η' .
If $\eta = \eta'$ then the experiments returns 1 else 0.

Construction for Partial Public Key HVE

Setup $(1^n, 1^\ell)$

1. Select a symmetric bilinear instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, e]$.
2. For $i \in [2\ell - 1]$, choose random $t_{1,i,0}, t_{2,i,0}, t_{1,i,1}, t_{2,i,1} \in \mathbb{Z}_p$ and set

$$\begin{aligned} K_i &= \begin{pmatrix} T_{1,i,0} = g^{t_{1,i,0}}, & T_{2,i,0} = g^{t_{2,i,0}} \\ T_{1,i,1} = g^{t_{1,i,1}}, & T_{2,i,1} = g^{t_{2,i,1}} \end{pmatrix} \\ \bar{K}_i &= \begin{pmatrix} \bar{T}_{1,i,0} = g^{1/t_{1,i,0}}, & \bar{T}_{2,i,0} = g^{1/t_{2,i,0}} \\ \bar{T}_{1,i,1} = g^{1/t_{1,i,1}}, & \bar{T}_{2,i,1} = g^{1/t_{2,i,1}} \end{pmatrix}. \end{aligned}$$

3. Return $SK = [\mathcal{I}, (K_i, \bar{K}_i)_{i \in [2\ell-1]}]$.

Construction for Partial Public Key HVE

Setup $(1^n, 1^\ell)$

1. Select a symmetric bilinear instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, e]$.
2. For $i \in [2\ell - 1]$, choose random $t_{1,i,0}, t_{2,i,0}, t_{1,i,1}, t_{2,i,1} \in \mathbb{Z}_p$ and set

$$\begin{aligned} K_i &= \begin{pmatrix} T_{1,i,0} = g^{t_{1,i,0}}, & T_{2,i,0} = g^{t_{2,i,0}} \\ T_{1,i,1} = g^{t_{1,i,1}}, & T_{2,i,1} = g^{t_{2,i,1}} \end{pmatrix} \\ \bar{K}_i &= \begin{pmatrix} \bar{T}_{1,i,0} = g^{1/t_{1,i,0}}, & \bar{T}_{2,i,0} = g^{1/t_{2,i,0}} \\ \bar{T}_{1,i,1} = g^{1/t_{1,i,1}}, & \bar{T}_{2,i,1} = g^{1/t_{2,i,1}} \end{pmatrix}. \end{aligned}$$

3. Return $\text{SK} = [\mathcal{I}, (K_i, \bar{K}_i)_{i \in [2\ell-1]}]$.

Notice: if $x = y$ then

$$e(T_{b,i,x}, \bar{T}_{b,i,y}) = e(g, g)$$

for all $i \in [2\ell - 1]$ and $b = 1, 2$.

Construction for Partial Public Key HVE

PPKeyGen (SK, Pol)

1. For $i = 1, \dots, \ell$,
for every $b \in \text{Pol}_i$, add $T_{1,i,b}$ and $T_{2,i,b}$ to PPK_i .
2. For $i = \ell + 1, \dots, 2\ell - 1$,
add $T_{1,i,0}$ and $T_{2,i,0}$ to PPK_i .
3. Return $\text{PPK}_{\text{Pol}} = [(\text{PPK}_i)_{i \in [2\ell-1]}]$.

Construction for Partial Public Key HVE

Encryption($\text{PPK}_{\text{Pol}}, \vec{x} = \langle x_1, \dots, x_\ell \rangle$)

1. If $\vec{x} \notin \mathbb{X}_{\text{Pol}}$ return \perp .
2. Append $(\ell - 1)$ 0-entries to \vec{x} .
3. Pick s at random from \mathbb{Z}_p .
4. $(s_1, \dots, s_{2\ell-1}) \leftarrow \text{LSS}(\ell, 2\ell - 1, 0)$.
5. For $i = 1, \dots, 2\ell - 1$,
set $X_{1,i} = T_{1,i,x_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,x_i}^{-s_i}$.
6. Return $\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}]$.

Construction for Partial Public Key HVE

Encryption($\text{PPK}_{\text{Pol}}, \vec{x} = \langle x_1, \dots, x_\ell \rangle$)

1. If $\vec{x} \notin \mathbb{X}_{\text{Pol}}$ return \perp .
2. Append $(\ell - 1)$ 0-entries to \vec{x} .
3. Pick s at random from \mathbb{Z}_p .
4. $(s_1, \dots, s_{2\ell-1}) \leftarrow \text{LSS}(\ell, 2\ell - 1, 0)$.
5. For $i = 1, \dots, 2\ell - 1$,
set $X_{1,i} = T_{1,i,x_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,x_i}^{-s_i}$.
6. Return $\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}]$.

Notice: if $\vec{x} \in \mathbb{X}_{\text{Pol}}$, then $\forall i T_{1,i,x_i}, T_{2,i,x_i} \in \text{PPK}_{\text{Pol}}$.

Linear Secret Sharing

(k, n) Linear Secret Sharing

- **Input:** a secret $s \in \mathbb{Z}_p$;
- **Output:** n shares (s_1, \dots, s_n) such that
 - ▶ any $k - 1$ (or fewer) shares are random and independent among themselves and are independent from the secret s ;
 - ▶ for any $F \subseteq [n]$ of size k there exist **reconstruction coefficients** α_i such that

$$s = \sum_{i \in F} \alpha_i s_i.$$

Linear Secret Sharing

(k, n) Linear Secret Sharing

- **Input:** a secret $s \in \mathbb{Z}_p$;
- **Output:** n shares (s_1, \dots, s_n) such that
 - ▶ any $k - 1$ (or fewer) shares are random and independent among themselves and are independent from the secret s ;
 - ▶ for any $F \subseteq [n]$ of size k there exist **reconstruction coefficients** α_i such that

$$s = \sum_{i \in F} \alpha_i s_i.$$

Notice: the reconstruction coefficients depend only on the set F and not on the shares.

Construction for Partial Public Key HVE

GenToken ($SK, \vec{y} = \langle y_1, \dots, y_\ell \rangle$)

1. Pick random $r \in \mathbb{Z}_p$.
2. $h = \#$ of non- \star entries of \vec{y} .
append $(\ell - h)$ 0-entries and $(h - 1)$ \star -entries
 $S_{\vec{y}}$ the non- \star entries of the extended vector.
Notice that $|S_{\vec{y}}| = \ell$.
3. $(r_1, \dots, r_{2\ell-1}) \leftarrow \text{LSS}(\ell, 2\ell - 1, 0)$.
4. For $i \in S_{\vec{y}}$,
set $Y_{1,i} = \bar{T}_{1,i,y_i}^{r_i}$ and $Y_{2,i} = \bar{T}_{2,i,y_i}^{r-r_i}$.
5. Return $T_{\vec{y}} = [S_{\vec{y}}, (Y_{1,i}, Y_{2,i})_{i \in S_{\vec{y}}}]$.

Construction for Partial Public Key HVE

Test ($\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}], T_{\tilde{y}} = [S, (Y_{1,i}, Y_{2,i})_{i \in S}]$)

1. Let $(v_i)_{i \in S}$ be the reconstruction coefficients for S .
2. Return 1 iff

$$\prod_{i \in S} [e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i})]^{v_i} = 1$$

Construction for Partial Public Key HVE

Test ($\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}], T_{\tilde{y}} = [S, (Y_{1,i}, Y_{2,i})_{i \in S}]$)

1. Let $(v_i)_{i \in S}$ be the reconstruction coefficients for S .
2. Return 1 iff

$$\prod_{i \in S} [e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i})]^{v_i} = 1$$

$$\begin{aligned} e(X_{1,i}, Y_{1,i}) &= e(T_{1,i,x_i}^{s-s_i}, \bar{T}_{1,i,y_i}^{r_i}) = e(g, g)^{r_i(s-s_i)} \\ e(X_{2,i}, Y_{2,i}) &= e(T_{2,i,x_i}^{-s_i}, \bar{T}_{2,i,y_i}^{r-r_i}) = e(g, g)^{-s_i(r-r_i)} \end{aligned}$$

Construction for Partial Public Key HVE

Test ($\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}], T_{\tilde{y}} = [S, (Y_{1,i}, Y_{2,i})_{i \in S}]$)

1. Let $(v_i)_{i \in S}$ be the reconstruction coefficients for S .
2. Return 1 iff

$$\prod_{i \in S} [e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i})]^{v_i} = 1$$

$$\begin{aligned} e(X_{1,i}, Y_{1,i}) &= e(T_{1,i,x_i}^{s-s_i}, \bar{T}_{1,i,y_i}^{r_i}) = e(g, g)^{r_i(s-s_i)} \\ e(X_{2,i}, Y_{2,i}) &= e(T_{2,i,x_i}^{-s_i}, \bar{T}_{2,i,y_i}^{r-r_i}) = e(g, g)^{-s_i(r-r_i)} \end{aligned}$$

$$e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i}) = e(g, g)^{sr_i} \cdot e(g, g)^{-rs_i}$$

Construction for Partial Public Key HVE

Test ($\tilde{X} = [(X_{1,i}, X_{2,i})_{i \in [2\ell-1]}], T_{\tilde{y}} = [S, (Y_{1,i}, Y_{2,i})_{i \in S}]$)

1. Let $(v_i)_{i \in S}$ be the reconstruction coefficients for S .
2. Return 1 iff

$$\prod_{i \in S} [e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i})]^{v_i} = 1$$

$$\begin{aligned} e(X_{1,i}, Y_{1,i}) &= e(T_{1,i,x_i}^{s-s_i}, \bar{T}_{1,i,y_i}^{r_i}) = e(g, g)^{r_i(s-s_i)} \\ e(X_{2,i}, Y_{2,i}) &= e(T_{2,i,x_i}^{-s_i}, \bar{T}_{2,i,y_i}^{r-r_i}) = e(g, g)^{-s_i(r-r_i)} \end{aligned}$$

$$e(X_{1,i}, Y_{1,i}) \cdot e(X_{2,i}, Y_{2,i}) = e(g, g)^{s r_i} \cdot e(g, g)^{-r s_i}$$

$$e(g, g)^{s \sum_i r_i v_i} \cdot e(g, g)^{-r \sum_i s_i v_i} = e(g, g)^{s \cdot 0} \cdot e(g, g)^{-r \cdot 0}$$

Security proofs

Security proofs

Can prove semantic and key security on complexity assumptions

Private-Key Searchable Encryption – The syntax

Private-Key Searchable Encryption

- 1 **Setup** $(1^n, 1^\ell)$ outputs the *secret key* SK.
- 2 **Encryption** (SK, \vec{x}) outputs ciphertext $\text{Ct}_{\vec{x}}$ with attribute $\vec{x} \in \Sigma^\ell$.
- 3 **GenToken** (SK, \vec{y}) outputs *key* $K_{\vec{y}}$ for pattern $\vec{y} \in \Sigma^\ell$.
- 4 **Test** $(\text{Ct}_{\vec{x}}, K_{\vec{y}})$ returns 1 iff $\vec{x} = \vec{y}$.

Semantic Security with Private Keys

1. **Initialization Phase.** \mathcal{A} announces two challenge attribute vectors $\vec{x}_0, \vec{x}_1 \in \Sigma^\ell$.
2. **Key-Generation Phase.** \mathcal{C} computes $\text{SK} \leftarrow \text{Setup}(1^n, 1^\ell)$.
3. **Query Phase I.** \mathcal{A} can make any number of **encryption** and key queries for patterns $\vec{y} \neq \vec{x}_0, \vec{x}_1$.
4. **Challenge construction.** \mathcal{C} chooses random $\eta \in \{0, 1\}$ and returns **Encryption**(SK, \vec{x}_η).
5. **Query Phase II.** Identical to Query Phase I.
6. **Output phase.** \mathcal{A} returns η' .
If $\eta = \eta'$ then the experiments returns 1 else 0.

Token Security with Private Keys

1. **Initialization Phase.** \mathcal{A} announces $\vec{y}_0, \vec{y}_1 \in \Sigma^\ell$.
2. **Key-Generation Phase.** \mathcal{C} computes $\text{SK} \leftarrow \text{Setup}(1^n, 1^\ell)$.
3. **Query Phase I.** \mathcal{A} can make any number of **key** queries and encryption queries for attributes $\vec{x} \neq \vec{y}_0, \vec{y}_1$.
4. **Challenge construction.** η is chosen at random from $\{0, 1\}$ and receives $\text{GenToken}(\text{SK}, \vec{y}_\eta)$.
5. **Query Phase II.** Identical to Query Phase I.
6. **Output phase.** \mathcal{A} returns η' .
If $\eta = \eta'$ then the experiments returns 1 else 0.

Construction for Private-Key Searchable Encryption

Setup $(1^n, 1^\ell)$

1. Select a symmetric bilinear instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, e]$.
2. For $i \in [\ell]$, choose random $t_{1,i,0}, t_{2,i,0}, t_{1,i,1}, t_{2,i,1} \in \mathbb{Z}_p$ and set

$$\begin{aligned} K_i &= \begin{pmatrix} T_{1,i,0} = g^{t_{1,i,0}}, & T_{2,i,0} = g^{t_{2,i,0}} \\ T_{1,i,1} = g^{t_{1,i,1}}, & T_{2,i,1} = g^{t_{2,i,1}} \end{pmatrix} \\ \bar{K}_i &= \begin{pmatrix} \bar{T}_{1,i,0} = g^{1/t_{1,i,0}}, & \bar{T}_{2,i,0} = g^{1/t_{2,i,0}} \\ \bar{T}_{1,i,1} = g^{1/t_{1,i,1}}, & \bar{T}_{2,i,1} = g^{1/t_{2,i,1}} \end{pmatrix}. \end{aligned}$$

3. Return $\text{SK} = [\mathcal{I}, (K_i, \bar{K}_i)_{i \in [\ell]}]$.

Construction for Private-Key Searchable Encryption

Encryption (SK, \vec{x})

1. Pick random $s \in \mathbb{Z}_p$.
2. Pick random $s_1, \dots, s_\ell \in \mathbb{Z}_p$ that sum up to 0.
3. For $i = 1, \dots, \ell$,
set $X_{1,i} = T_{1,i,x_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,x_i}^{-s_i}$.
4. Return $Ct_{\vec{x}} = [(X_{1,i}, X_{2,i})_{i \in [\ell]}]$.

Construction for Private-Key Searchable Encryption

Encryption (SK, \vec{x})

1. Pick random $s \in \mathbb{Z}_p$.
2. Pick random $s_1, \dots, s_\ell \in \mathbb{Z}_p$ that sum up to 0.
3. For $i = 1, \dots, \ell$,
set $X_{1,i} = T_{1,i,x_i}^{s-s_i}$ and $X_{2,i} = T_{2,i,x_i}^{-s_i}$.
4. Return $Ct_{\vec{x}} = [(X_{1,i}, X_{2,i})_{i \in [\ell]}]$.

GenToken (SK, \vec{y})

1. Pick random $r \in \mathbb{Z}_p$.
2. Pick random $r_1, \dots, r_\ell \in \mathbb{Z}_p$ that sum up to 0.
3. For $i = 1, \dots, \ell$,
set $Y_{1,i} = \bar{T}_{1,i,y_i}^{r-r_i}$ and $Y_{2,i} = \bar{T}_{2,i,y_i}^{-r_i}$.
4. Return $K_{\vec{y}} = [(Y_{1,i}, Y_{2,i})_{i \in [\ell]}]$.

Security proof strategy

Semantic vs. Token security

- Encryption uses keys K_i , $i \in [\ell]$;

Security proof strategy

Semantic vs. Token security

- Encryption uses keys K_i , $i \in [\ell]$;
- Token generation is encryption w.r.t. to keys \bar{K}_i , $i \in [\ell]$;

Security proof strategy

Semantic vs. Token security

- Encryption uses keys K_i , $i \in [\ell]$;
- Token generation is encryption w.r.t. to keys \bar{K}_i , $i \in [\ell]$;
- In the game for semantic security, \mathcal{A} can ask
 - ▶ any encryption query for keys K_i ;
 - ▶ encryption queries for keys \bar{K}_i and pattern $\vec{y} \neq \vec{x}_0, \vec{x}_1$;

Security proof strategy

Semantic vs. Token security

- Encryption uses keys K_i , $i \in [\ell]$;
- Token generation is encryption w.r.t. to keys \bar{K}_i , $i \in [\ell]$;
- In the game for semantic security, \mathcal{A} can ask
 - ▶ any encryption query for keys K_i ;
 - ▶ encryption queries for keys \bar{K}_i and pattern $\vec{y} \neq \vec{x}_0, \vec{x}_1$;
- In the game for key security, \mathcal{A} can ask
 - ▶ any encryption query for keys \bar{K}_i ;
 - ▶ encryption queries for keys K_i and attributes $\vec{x} \neq \vec{y}_0, \vec{y}_1$;

Security proof strategy

Semantic vs. Token security

- Encryption uses keys K_i , $i \in [\ell]$;
- Token generation is encryption w.r.t. to keys \bar{K}_i , $i \in [\ell]$;
- In the game for semantic security, \mathcal{A} can ask
 - ▶ any encryption query for keys K_i ;
 - ▶ encryption queries for keys \bar{K}_i and pattern $\vec{y} \neq \vec{x}_0, \vec{x}_1$;
- In the game for key security, \mathcal{A} can ask
 - ▶ any encryption query for keys \bar{K}_i ;
 - ▶ encryption queries for keys K_i and attributes $\vec{x} \neq \vec{y}_0, \vec{y}_1$;

Semantic Security \iff Token Security

Zero Sum Assumption

Consider the following game between a challenger \mathcal{C} and an adversary \mathcal{A} .

ZeroSumExp $_{\mathcal{A}}(1^n, 1^\ell)$

01. \mathcal{C} randomly picks a_1, \dots, a_ℓ such that $\sum_i a_i = 0$;
02. \mathcal{C} chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, e]$ with security parameter 1^n ;
03. **for** $i \in [\ell]$
 \mathcal{C} chooses random $u_i \in \mathbb{Z}_p$ and sets $U_i = g^{u_i}$ and $V_i = U_i^{a_i}$;
04. \mathcal{C} chooses random $\eta \in \{0, 1\}$;
05. **if** $\eta = 0$ **then** \mathcal{C} sets V_1 to a random element of \mathbb{G} ;
06. \mathcal{C} runs \mathcal{A} on input $[\mathcal{I}, (U_i)_{i \in [\ell]}, (V_i)_{i \in [\ell]}]$;
07. Let η' be \mathcal{A} 's guess for η ;
08. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Split Zero Sum Assumption

Consider the following game between a challenger \mathcal{C} and an adversary \mathcal{A} .

SplitZeroSumExp $_{\mathcal{A}}(1^n, 1^\ell)$

01. \mathcal{C} randomly picks a_1, \dots, a_ℓ such that $\sum_i a_i = 0$;
02. \mathcal{C} chooses instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, e]$ with security parameter 1^n ;
03. \mathcal{C} chooses random $u, w \in \mathbb{Z}_p$ and sets $W = g^w$;
04. **for** $i \in [\ell]$
 - \mathcal{C} chooses random $u_i \in \mathbb{Z}_p$;
 - sets $U_i = g^{u_i}$, $V_i = U_i^{a_i}$, $A_i = g^{a_i}$, and $S_i = U_i^{u_i}$;
05. \mathcal{C} sets $\hat{U} = U_1^w$;
06. \mathcal{C} chooses random $\eta \in \{0, 1\}$;
07. **if** $\eta = 1$ **then** \mathcal{C} sets $Z = W^{u-a_1}$ **else** \mathcal{C} chooses random $Z \in \mathbb{G}$;
08. \mathcal{C} runs \mathcal{A} on input $[\mathcal{I}, (U_i)_{i \in [\ell]}, (V_i)_{i \in [\ell]}, (A_i)_{i \in [\ell]}, (S_i)_{i \in [\ell] \setminus \{1\}}, W, \hat{U}, Z]$;
09. Let η' be \mathcal{A} 's guess for η ;
10. **if** $\eta = \eta'$ **then** return 1 **else** return 0.

Theorem

Under the Zero Sum Assumption and the Split Zero Sum Assumption, there exists private-key searchable encryption with semantic and key security.

Notice: construction based on pairings on prime order groups.

Open problems

- Partial Public-Key HVE that hides the positions of the \star 's in the pattern, with prime order groups;
- Full security and not simply selective security;

Thank you