Lesson 8 : Key-Policy Attribute-Based Encryption and Public Key Encryption with Keyword Search

November 3, 2014

teacher : Benoît Libert
scribe : Florent Bréhard

Key-Policy Attribute-Based Encryption (KP-ABE)

Motivation

In a standard public-key encryption scheme, each user has his own public key and secret key, so that if one wants to encrypt a message intended for several receivers (for example, according to their jobs in a company), it will be necessary to compute the ciphertext for each public key of each recipient, which implies a huge loss of time and space.

The idea of a Key-Policy Attribute-Based Encryption scheme (KP-ABE) is to have the sender encrypt the message only once. It is the policy assigned to users' keys that will determine if these users will be allowed to decrypt:

- Ciphertexts are labeled with a set ω of descriptive attributes.
- Private key corresponds to an access policy P.
- Decryption works if and only if $P(\omega) = 1$

Here are some examples:

- Attributes can be {Researcher, Teacher, Student, ENS Lyon, CNRS}.
- M. Dupont is researcher at the CNRS. So, his policy will be : (Researcher AND CNRS).
- Mme Dupré is researcher at the ENS Lyon and also teacher at the ENS Lyon. So her policy will be : ((Researcher OR Teacher) AND ENS Lyon).

- M. Dupuis is researcher at the CNRS and also gives courses at the ENS Lyon. His policy is : ((Researcher AND CNRS) OR (Teacher AND ENS Lyon)).
- Chloé is student at the ENS Lyon, her policy is : (Student AND ENS Lyon)
- If ω = {Researcher, CNRS}, which means that only researchers at the CNRS should be able to decrypt the message, then only M. Dupont and M. Dupuis have the good policy to read the message.
- If ω = {Researcher, ENS Lyon, CNRS}, meaning that recipients must be all the research staff in both ENS Lyon and CNRS, then M. Dupont, Mme Dupré and M. Dupuis can all read the message.
- If ω = {Teacher, Student, ENS Lyon}, only teachers or students at the ENS Lyon will be able to decrypt the message. Here, Mme Dupré, M. Dupuis and Chloé have the corresponding policy.

Definition

A *KP-ABE scheme* is a tuple of algorithms with the following specification:

$\mathsf{Setup}(\lambda)$	:	Given $\lambda \in \mathbb{N}$, outputs a master key pair (MPK, MSK).
$\mathsf{Keygen}(MSK, P)$	:	Given MSK and a policy P, outputs SK_P.
$\mathsf{Encrypt}(MPK, M, \omega)$	:	Given an attribute set ω, outputs ciphertext CT.
$\mathsf{Decrypt}(MPK, SK_P, CT)$	:	Given CT and SK_P, outputs M or $\perp$.

Correctness condition

The correctness condition expresses the fact tha,t if a message is encrypted with an attribute set ω and if user A holds a key SK_P for a policy P that accepts these attributes (i.e., $P(\omega) = 1$), then A can decrypt the ciphertext with his secret key SK_P.

For any policy P and any attribute set ω such that $P(\omega) = 1$,

$$M = \mathsf{Decrypt}(MPK, SK_P, \mathsf{Encrypt}(MPK, M, \omega))$$

Selective security

The intuitive notion of security for such an encryption scheme is that an adversary $\mathcal{A}$ should not be able to decrypt a ciphertext labeled with some attribute set $\omega^\star$ if he does not have access to a secret key $SK_{P^\star}$ such that the policy $P^\star$ satisfies $P^\star(\omega^\star) = 1$, even if he can obtain SK_P for policies P such that $P(\omega^\star) = 0$. Precisely, no PPT adversary $\mathcal{A}$ should have a non-negligible advantage in this game :

The adversary $\mathcal{A}$ chooses $\omega^\star$ at the beginning (*before* seeing the master public key MPK) and can adaptively obtain a polynomial number of private keys SK_P for a arbitrary policies P sucht that $P(\omega^\star) = 0$. After a first series of queries, $\mathcal{A}$ chooses two messages M_0 and M_1 and send them to challenger. The challenger chooses a random bit $\gamma \xleftarrow{R} \{0,1\}$ and sends ciphertext $c^\star = \mathsf{Encrypt}(MPK, M_\gamma, \omega^\star)$ to $\mathcal{A}$. $\mathcal{A}$ can make further queries for private keys SK_P such that $P(\omega^\star) = 0$ and finally outputs $\gamma' \in \{0,1\}$. The adversary $\mathcal{A}$ wins if $\gamma' = \gamma$. Its advantage is defined in the usual way, as the distance $\mathbf{Adv}_{\mathcal{A}}(\lambda) := |\Pr[\gamma' = \gamma] - 1/2|$.

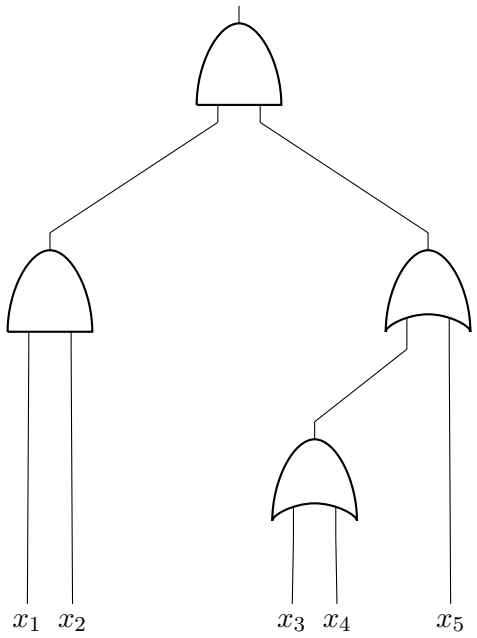
Formulae

For now, we restrict ourselves to the case where a policy P is defined by a monotone Boolean *formula*.

A monotone Boolean formula is a directed tree where :

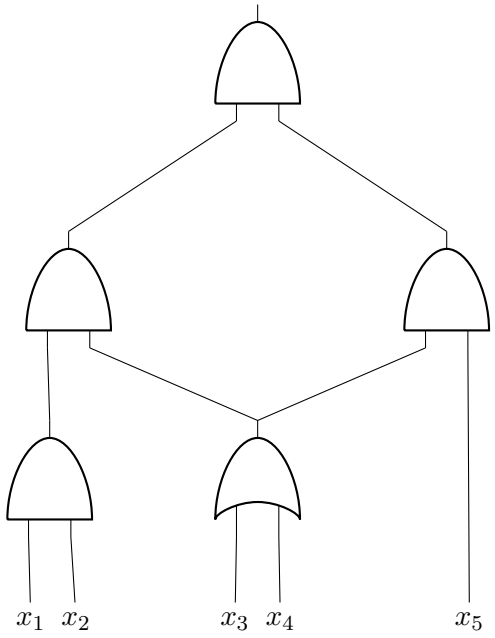
- leaves correspond to inputs.
- non-leaf nodes are gates (AND,OR).

Here is an example of a formula that calculates $(x_1 \wedge x_2) \wedge ((x_3 \vee x_4) \vee x_5)$:



The inputs (leaves) are the attributes. An attribute is set to TRUE if it belongs to the ω, and FALSE otherwise. A formula corresponds in that way to an access policy : its accepts or reject an attribute set ω.

It must be stressed out that this definition of formula is not as general as the notion of Boolean circuit (a circuit is a directed acyclic graph, which may not be a tree since the output of a gate may be the input of several other gates). However, Boolean formulas are sufficient for many applications. Here is an example of Boolean circuit that is not a formula:



Access structure

Let $\mathcal{P} = \{1, \ldots, n\}$ be a set of positive integers, representing the set of all possible attributes. An *access structure* $\mathbb{A} \subseteq 2^{\mathcal{P}}$ is a collection of non-empty subsets of $\mathcal{P}$. It corresponds exactly to the notion of access policy : $P(\omega) = 1$ if and only if $\omega \in \mathbb{A}$.

It is called *monotone* if:

$$\forall B, C \in 2^{\mathcal{P}} \cdot B \in \mathbb{A} \ \wedge \ B \subseteq C \Rightarrow C \in \mathbb{A}$$

It is easy to see that access structures obtained by formulae as above are monotone. This is due to the fact that we do not allow NOT gates in the definition of formulae.

Monotone Span Program

Let $\mathbb{K}$ be a field and let $\{x_1, \ldots, x_n\}$ be a set of variables.

A *Monotone Span Program* (MSP) is a pair $(\boldsymbol{M}, \rho)$, where $\boldsymbol{M} \in \mathbb{K}^{l \times k}$ is a matrix and $\rho : \{1, \ldots, l\} \to \{x_1, \ldots, x_n\}$ is a labelling function.

For any $\gamma \subseteq \{x_1, \ldots, x_n\}$, define the submatrix $\boldsymbol{M}_\gamma$ of $\boldsymbol{M}$ obtained by keeping only the rows of index i such that $\rho(i) \in \gamma$:

$$\boldsymbol{M}_\gamma = (M_{ij})_{\substack{i \in \rho^{-1}(\gamma) \\ 1 \leq j \leq k}}$$

$(\boldsymbol{M}, \rho)$ *accepts* γ if and only if $\overrightarrow{1} = (1, 0, \ldots, 0) \in \mathsf{rowspan}(\boldsymbol{M}_\gamma)$.

A MSP $(\boldsymbol{M}, \rho)$ computes a Boolean function $f_{\boldsymbol{M}}$ over $\{x_1, \ldots, x_n\}$ if it accepts exactly those γ such that $f_{\boldsymbol{M}}(\gamma) = 1$. Note that the access structure $\mathbb{A}$ defined here by its characteristic function is clearly monotone (because the notion of linear spanning is monotone).

Linear Secret Sharing

Let $\mathcal{P} = \{1, \ldots, n\}$ be a set of parties, let $\boldsymbol{M} \in \mathbb{F}_p^{l \times k}$ be a matrix over a finite field $\mathbb{F}_p$ and let $\rho : \{1, \ldots, l\} \to \mathcal{P}$ be a function that maps the rows of $\boldsymbol{M}$ to $\mathcal{P}$ for labelling. A *Linear Secret Sharing Scheme* (LSSS) for a (monotone) access structure $\mathbb{A} \subseteq 2^{\mathcal{P}}$ represented by $(\boldsymbol{M}, \rho)$ consists of algorithms :

$\mathsf{Share}(\boldsymbol{M}, \rho, s)$: Given $(\boldsymbol{M}, \rho)$ and a secret $s \in \mathbb{F}_p$, picks $\boldsymbol{\beta} = (s, \beta_2, \ldots \beta_k)$ with $\beta_2, \ldots, \beta_k \xleftarrow{R} \mathbb{F}_p$ and defines $\lambda_i = \boldsymbol{M}_i^T \boldsymbol{\beta}$ which is the share assigned to party $\rho(i)$.

$\mathsf{Reconstruct}(\boldsymbol{M}, \rho, S)$: Given an access set $S \in \mathbb{A}$, lets $I = \{i \mid \rho(i) \in S\}$. It outputs constants $\{\mu_i\}_{i \in I}$ such that $s = \sum_{i \in I} \mu_i \lambda_i$.

The idea behind such a scheme is the following. A secret s is split into several shares λ_i. If ones wants to reconstruct the secret s, one can do that by taking a linear combination of the $\{\lambda_i\}_{i \in I}$ such that I corresponds (via ρ) to a valid $S \in \mathbb{A}$. The following proposition makes the link with MSP defining an access structure $\mathbb{A}$:

Proposition. *There exists an efficient LSSS for a monotone access structure* $\mathbb{A}$ *if and only if there exists a small MSP for the characteristic function of* $\mathbb{A}$.

KP-ABE for monotone Boolean formulae (Goyal, Pandey, Sahai, Waters, ACM-CCS '06 [2])

Here, we present a concrete KP-ABE scheme that can represent all monotone access structures $\mathbb{A}$ defined using a MSP $(\boldsymbol{M}, \rho)$:

- $\mathsf{Setup}(\lambda)$:
  1. Chooses groups (G, G_T) of prime order $p > 2^\lambda$ with a pairing $e : G \times G \to G_T$ and generators $g, g_2 \xleftarrow{R} G$.

2. Chooses $y \xleftarrow{R} \mathbb{F}_p$ and computes $g_1 = g^y$.
3. Chooses a function $T : \mathbb{F}_p \to G$ (to be specified later) and sets:

$$MPK = ((G, G_T), g, g_1 = g^y, g_2, T) \quad \text{and} \quad MSK = y$$

- $\mathsf{Keygen}(MSK, (\boldsymbol{M}, \rho))$:

  To generate a key for $\mathbb{A} = (\boldsymbol{M}, \rho)$ where $\boldsymbol{M} \in \mathbb{F}_p^{l \times k}$ and $\rho : \{1, \ldots, l\} \to \mathcal{P}$, choose a random vector $\boldsymbol{\beta} \xleftarrow{R} \mathbb{F}_p^k$ such that $\boldsymbol{\beta} \cdot (1, 0, \ldots, 0) = y$. For each row $\boldsymbol{M}_i$ of $\boldsymbol{M} \in \mathbb{F}_p^{l \times k}$, choose $r_i \xleftarrow{R} \mathbb{F}_p$ and compute a pair

$$(D_i, d_i) = \left(g_2^{\boldsymbol{M}_i \cdot \boldsymbol{\beta}} \cdot T(\rho(i))^{r_i}, g^{r_i} \right).$$

  Notet that (D_i, d_i) satisfy the following relations :

$$\begin{aligned} e(D_i, g) &= e(g_2, g)^{\boldsymbol{M}_i \cdot \boldsymbol{\beta}} \cdot e(T(\rho(i)), d_i) \\ e(D_i, g^s) &= e(g_2, g^s)^{\boldsymbol{M}_i \cdot \boldsymbol{\beta}} \cdot e(T(\rho(i))^s, d_i), \end{aligned}$$

  for any $s \in \mathbb{F}_p$.

  Return $SK_{\mathbb{A}} = \{(D_i, d_i)\}_{1 \le 1 \le l}$.

- $\mathsf{Encrypt}(MPK, \mathsf{Msg}, \omega)$: To encrypt $\mathsf{Msg} \in G_T$ under ω, chooses $s \xleftarrow{R} \mathbb{F}_p$ and computes :

$$CT = \left(\omega, E' = \mathsf{Msg} \cdot e(g_1, g_2)^s, E = g^s, \{E_i = T(i)^s\}_{i \in \omega} \right)$$

- $\mathsf{Decrypt}(MPK, SK_{\mathbb{A}}, CT)$:

  Given $SK_{\mathbb{A}}$ for $A = (\boldsymbol{M}, \rho)$ and $CT = (\omega, E', E, \{E_i\}_{i \in \omega})$, define the index set

$$I = \{ i \in \{1, \ldots, l\} \mid \rho(i) \in \omega \}.$$

  Since $\mathbb{A}(\omega) = 1$, there exist coefficients $\{\mu_i\}_{i \in I}$ such that $(1, 0, \ldots, 0) = \sum_{i \in I} \mu_i \boldsymbol{M}_i$.

  For each $i \in I$, compute

$$\Theta_i = \frac{e(d_i, E_i)}{e(D_i, E)} = e(g_2, g)^{-s \cdot \boldsymbol{M}_i \cdot \boldsymbol{\beta}}.$$

  Then, return

$$\mathsf{Msg} = E' \cdot \prod_{i \in I} \Theta_i^{\mu_i} = E' \cdot e(g, g_2)^{-s \cdot y}$$

Theorem ([2]). *The scheme provides selective security under the DBDH assumption.*

Proof. We will first need the following lemma from linear algebra :

Lemma. *A vector $\boldsymbol{\pi} \in \mathbb{F}_p^k$ is linearly independent of the rows of a matrix $\boldsymbol{N} \in \mathbb{F}_p^{l' \times k}$ if and only if there exists a vector $\boldsymbol{w} \in \mathbb{F}_p^k$ such that $\boldsymbol{N}\boldsymbol{w} = 0$ and $\boldsymbol{\pi} \cdot \boldsymbol{w} \neq 0$.*

Let $\mathcal{A}$ be a selective adversary with advantage ϵ. We build a DBDH distinguisher $\mathcal{B}$ that takes as input (g, g^a, g^b, g^c, Z) and uses $\mathcal{A}$ to decide whether $Z = e(g,g)^{abc}$ or $Z \in_R G_T$.

$\mathcal{A}$ begins the game by choosing some attribute set $\omega^\star$. To generate MPK, $\mathcal{B}$ defines $g_1 = g^a$ and $g_2 = g^b$. Then, $\mathcal{B}$ chooses a function $T : \mathbb{F}_p \to G$ such that:

$$T(x) = g_2^{F(x)} \cdot g^{J(x)} \qquad \forall x \in \mathbb{F}_p$$

for certain functions $F, J : \mathbb{F}_p \to \mathbb{F}_p$, which are kept internal to $\mathcal{B}$, that satisfy the conditions

$$\begin{array}{ll} F(x) = 0 & \forall x \in \omega^\star \\ F(x) \neq 0 & \forall x \notin \omega^\star \end{array}$$

For example, if we fix an upper bound $n \geq |\omega^\star|$ on the cardinality of any attribute set, $F(X)$ and $J(X)$ can be chosen as polynomials

$$\begin{array}{rclcl} F(X) & = & \prod_{i \in \omega^\star}(X - i) & = & \sum_{j=0}^n F_j X^j \\ J(X) & = & \sum_{j=0}^n J_j X^j & \xleftarrow{R} & \mathbb{F}_p[X] \end{array}$$

and we can define $T(X) = \prod_{j=0}^n u_j^{X^j}$, where $u_j = g_2^{F_j} \cdot g^{J_j}$ for each $j \in \{0, \ldots, n\}$ and $\{u_j\}_{j=0}^n$ are included in MPK.

$\mathcal{A}$ is given $MPK = \big((G, G_T), g, g_1 = g^a, g_2 = g^b, T\big)$, meaning that $\mathcal{B}$ implicitely defines $MSK = a$ (which is unknown).

Queries: Suppose $\mathcal{A}$ queries $SK_{\mathbb{A}}$ such that $\mathbb{A}(\omega^\star) = 0$, where $\mathbb{A} = (\boldsymbol{M}, \rho)$ with $\boldsymbol{M} \in \mathbb{F}_p^{l \times k}$. B defines $I = \{i \in \{1, \ldots, l\} \mid \rho(i) \in \omega^\star\}$. Since $\mathbb{A}(\omega^\star) = 0$, $(1, 0, \ldots, 0)$ is not in the row space of $\boldsymbol{M}_I$, the submatrix of $\boldsymbol{M}$ obtained by keeping only the rows $\boldsymbol{M}_i$ of $\boldsymbol{M}$ such that $i \in I$. Hence $\exists \boldsymbol{w} \in \mathbb{F}_p^k \cdot \boldsymbol{M}_I \cdot \boldsymbol{w} = 0$ and $(1, 0, \ldots, 0) \cdot \boldsymbol{w} \neq 0$ (so $w_1 \neq 0$).

$\mathcal{B}$ chooses $\boldsymbol{v} = (v_1, \ldots, v_k) \xleftarrow{R} \mathbb{F}_p^k$ and implicitely defines $\boldsymbol{u} = \boldsymbol{v} + \psi \boldsymbol{w}$ where $\psi = \frac{a - v_1}{w_1}$ (not computable by $\mathcal{B}$). Note that $\boldsymbol{u} \cdot (1, 0, \ldots, 0) = v_1 + \frac{a - v_1}{w_1} \boldsymbol{w} \cdot (1, 0, \ldots 0) = a$.

For each $i \in I$ (so that $\rho(i) \in \omega^\star$), we have $\lambda_i = \boldsymbol{M}_i \cdot \boldsymbol{u} = \boldsymbol{M}_i \cdot \boldsymbol{v}$ and $\mathcal{B}$ can compute :

$$(D_i, d_i) = \Big(g_2^{\boldsymbol{M}_i \cdot \boldsymbol{v}} \cdot T(\rho(i))^{r_i}, g^{r_i}\Big) \quad \text{where} \quad r_i \xleftarrow{R} \mathbb{F}_p$$

For each $i \in \{1, \ldots, l\} \setminus I$ (so that $\rho(i) \notin \omega^\star$), we have $T(\rho(i)) = g_2^{F(\rho(i))} \cdot g^{J(\rho(i))}$ such that $F(\rho(i)) \neq 0$. So, $\mathcal{B}$ can compute a pair

$$(D_i', d_i') = \big(g_2^a \cdot T(\rho(i))^{\tilde{r_i}}, g^{\tilde{r_i}}\big) \tag{1}$$

for some $\tilde{r}_i \in_R \mathbb{F}_p$ using the Boney-Boyen technique. To this end, $\mathcal{B}$ chooses a random $\tilde{\tilde{r}}_i \in_R \mathbb{F}_p$ and implicitly defines $\tilde{\tilde{r}}_i = \tilde{r}_i + \frac{a}{F(\rho(i))}$. So, we have:

$$\left(T(\rho(i)) = g_2^{F(\rho(i))} \cdot g^{J(\rho(i))}\right)^{\tilde{r}_i + \frac{a}{F(\rho(i))}} = g_2^a \cdot (T(\rho(i)))^{\tilde{r}_i} \cdot g^{a \cdot \frac{J(\rho(i))}{F(\rho(i))}}$$

So, the pair

$$\left(T(\rho(i))^{\tilde{\tilde{r}}_i} \cdot (g^a)^{-\frac{J(\rho(i))}{F(\rho(i))}}, g^{\tilde{\tilde{r}}_i} \cdot (g^a)^{-\frac{1}{F(\rho(i))}}\right)$$

is computable by $\mathcal{B}$ and forms a valid pair (D_i', d_i') of the form (1). From this point, $\mathcal{B}$ can obtain

$$(D_i, d_i) = \left(g_2^{\boldsymbol{M}_i \cdot \boldsymbol{u}} \cdot T(\rho(i))^{r_i}, g^{r_i}\right)$$

$$\text{as} \quad \begin{cases} D_i = g_2^{\boldsymbol{M}_i \cdot \boldsymbol{v}} \cdot \left(D_i' \cdot g_2^{-v_1}\right)^{\frac{\boldsymbol{M}_i \cdot \boldsymbol{w}}{w_1}} \\ d_i = d_i'^{\frac{\boldsymbol{M}_i \cdot \boldsymbol{w}}{w_1}} \end{cases}$$

Then, $\mathcal{B}$ returns $SK_{\mathbb{A}} = \{(D_i, d_i)\}_{1 \le i \le l}$ to $\mathcal{A}$.

Challenge : $\mathcal{A}$ chooses $\mathsf{Msg}_0, \mathsf{Msg}_1$. Then, $\mathcal{B}$ picks $\gamma \xleftarrow{R} \{0,1\}$ and computes

$$CT^\star = \left(\omega^\star, E' = \mathsf{Msg}_\gamma \cdot Z, E = g^c, \left\{E_i = (g^c)^{J(i)} = T(i)^c\right\}_{i \in \omega^\star}\right)$$

(For each $i \in \omega^\star$, we know that $T(i) = g_2^{F(i)} \cdot g^{J(i)} = g^{J(i)}$).

- If $Z = e(g,g)^{abc}$, $CT^\star = \left(\omega^\star, E' = \mathsf{Msg}_\gamma \cdot e(g_1, g_2)^c, E = g^c, \{E_i = T(i)^c\}_{i \in \omega^\star}\right)$ is a valid encryption of Msg_γ with the attribute set $\omega^\star$. In this case, $\mathcal{A}$ should output $\gamma' = \gamma$ with probability $\frac{1}{2} + \epsilon$.

- If $Z \xleftarrow{R} G_T$, the challenge ciphertext $CT^\star$ is distributed as an encryption of a random message $\mathsf{Msg}_{\text{rand}} \in_R G_T$, which is completely independent of $\mathsf{Msg}_0, \mathsf{Msg}_1$. So, $\mathcal{A}$ should output a bit $\gamma' \in \{0,1\}$ independent of the γ chosen by B, so $\gamma' = \gamma$ with probability $1/2$.

Output: $\mathcal{A}$ outputs $\gamma' \in \{0,1\}$. If $\gamma = \gamma'$, $\mathcal{B}$ outputs 1 (meaning $Z = e(g,g)^{abc}$). Otherwise, $\mathcal{B}$ outputs 0 (meaning that $Z \in_R G_T$). Clearly, this gives an advantage ϵ to $\mathcal{B}$ as a distinguisher for the DBDH problem. □

Public Key Encryption with Keyword Search (PEKS) (Boneh, Di Crescenzo, Ostrovsky, Persiano, EUROCRYPT '04 [1])

Idea: A trapdoor t_w allows testing whether c is an encryption of a given keyword W.

A PEKS scheme consists of the following set of algorithms :

$\mathsf{Keygen}(\lambda)$	:	Given a security parameter $\lambda \in \mathbb{N}$, outputs (PK, SK).
$\mathsf{Trapdoor}(SK, W)$	:	Outputs t_W for the keyword W.
$\mathsf{Encrypt}(PK, W)$	:	Outputs c which encrypts W.
$\mathsf{Test}(PK, t_W, c)$	:	Outputs 0 or 1.

Correctness:

For all $\lambda \in \mathbb{N}$ and $(PK, SK) \leftarrow \mathsf{Keygen}(\lambda)$, if $t_W \leftarrow \mathsf{Trapdoor}(SK, W)$, then :

$$\mathsf{Test}(PK, \mathsf{Encrypt}(PK, W), t_W) = 1 \qquad \text{with high probability}$$

Computational consistency:

For any PPT adversary $\mathcal{A}$, the following experiment outputs 1 with negligible probability:

1. Run $(PK, SK) \leftarrow \mathsf{Keygen}(\lambda)$ and PK is given to $\mathcal{A}$.
2. $\mathcal{A}$ outputs $W, W' \in \{0,1\}^\star$ such that $W \neq W'$.
3. Compute $c \leftarrow \mathsf{Encrypt}(PK, W)$ and $t_{W'} \leftarrow \mathsf{Trapdoor}(SK, W')$. Return 1 if $W \neq W'$ and $\mathsf{Test}(PK, t_{W'}, c) = 1$.

Semantic Security for PEKS (a.k.a. keyword-privacy):

No PTT adversary $\mathcal{A}$ has non-negligible advantage in this game:

1. The challenger generates $(PK, SK) \leftarrow \mathsf{Keygen}(\lambda)$, initializes a set $Q \leftarrow \emptyset$ and gives PK to $\mathcal{A}$.
2. $\mathcal{A}$ makes queries: $\mathcal{A}$ chooses W and obtains $t_W \leftarrow \mathsf{Trapdoor}(SK, W)$. The challenger updates $Q \leftarrow Q \cup \{W\}$.
3. $\mathcal{A}$ chooses distinct keywords $W_0, W_1 \notin Q$ and obtains $c^\star \leftarrow \mathsf{Encrypt}(PK, W_\gamma)$ where $\gamma \xleftarrow{R} 0, 1$.
4. $\mathcal{A}$ makes more queries for keywords $W \notin \{W_0, W_1\}$.
5. $\mathcal{A}$ outputs $\gamma' \in \{0,1\}$ and wins if $\gamma' = \gamma$.

We define the advantage of adversary $\mathcal{A}$ in this game:

$$\mathsf{Adv}_{\mathcal{A}}^{\text{PEKS-IND-CPA}}(\lambda) = |\Pr[\gamma' = \gamma] - \frac{1}{2}|$$

References

[1] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. *EUROCRYPT*, 2004.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *ACM CCS*, 2006.