Cours Crypto (11)

Florent Guépin

December 2019

1 Public-key encryption with keyword search (PEKS)

First introduced by Boneh-DiCrescenso-Ostrovsky-Persiazo (2004).
Given a keypair (pk,sk), sk allows deriving T_w for a specific keyword w (eg. "urgent"). Given T_w, gateway can test if a ciphetext c encrypts w while learning nothing else. Given

$$\big(\mathrm{Enc}(pk, M), \mathrm{PEKS}(pk, w_1), \ldots, \mathrm{PEKS}(pk, w_l)\big),$$

T_w can test if $w \in \{w_1, \cdots, w_l\}$ while learning nothing else (no interaction with the holder of sk is required).

Definition: A PEKS scheme is a tuple (Keygen, Enc, Trapdoor,Test) of efficient algorithms such that:

- **Keygen(1^λ)** : given security parameter λ, outputs a key pair (pk,sk)
- **Enc(pk,w)** : Given pk and a keyword $w \in \{0,1\}^*$, outputs a cyphetext c.
- **Trapdoor(sk,w)** : Given secret key sk and keyword, outputs a trapdoor T_w.
- **Test(pk,T_w,c)** : Given pk, a ciphertext c and a trapdoor T_w, outputs 0 or 1.

Notion of Correctness: If (pk,sk) $\leftarrow$ Keygen(1^λ), for any w, $c \leftarrow$ Enc (pk,w) and $T_w \leftarrow$ Trapdoor(sk,w), we have Test(pk,T_w,c) = 1.

Definition: A PEKS scheme provides semantic security if no PPT adversary has noticeable advantage in this game.

1. Challenger generates (pk,sk) $\leftarrow$ Keygen(1^λ) and gives pk to adversary A.
2. A can adaptively choose keywords w and obtain $T_w \leftarrow$ Trapdoor(sk,w) from Challenger.
3. A chooses w_0, w_1 such that it did not obtain T_{w_0}, T_{w_1} so far. Challenger flips a coin $d \leftarrow U(\{0,1\})$ and gives $c \leftarrow$ Enc(pk,w_d) to A.
4. A can made more queries for keywords $w \notin \{w_0, w_1\}$.
5. A outputs $d' \in \{0,1\}$ and wins if $d' = d$. Adv(A) = $|\mathbb{P}[d' = d] - \frac{1}{2}|$

1.1 PEKS implies IBE

Let a PEKS scheme (Keygen, Enc, Trapdoor, Test) we build an IBE out of it.

- **Setup**(1^λ): Run (pk,sk) $\leftarrow$ PEKS.Keygen(1^λ). Outputs $mpk = pk$ and $msk = sk$.

- **Keygen**(msk,ID): Given an identity ID, compute $T_{ID||0} \leftarrow$ PEKS.Trapdoor($sk, ID||0$) and $T_{ID||1} \leftarrow$ PEKS.Trapdoor($sk, ID||1$) output $sk_{ID} = (T_{ID||0}, T_{ID||1})$
- **Encrypt**(mpk,ID,μ): To encrypt $\mu \in \{0,1\}$ under ID, compute $c \leftarrow$ PEKS.Enc($pk, ID||\mu$)
- **Decrypt**(mpk,SK_{ID},c): Parse SK_{ID} as $(T_{ID||0}, T_{ID||1})$. It Test(pk, $T_{ID||0}$, c)=1 output $\mu = 0$. If Test(pk, $T_{ID||1}$, c)=1 output $\mu = 1$. In any other case, output $\perp$.

Lemma: The IBE scheme provides IND-ID-CPA security if the PEKS scheme is semantically secure.

1.2 PEKS from bilinear maps (BDOP, 2004)

Construction based on the Boneh-Francklin IBE.

- **Keygen**(1^λ) :
  1. Choose groups (G,G$_T$) of prime order $p > 2^\lambda$ with a bilinear map $e : G \times G \to G_T$, and a generator $g \leftarrow U(G)$.

  2. Choose $\alpha \leftarrow U(\mathbb{Z}_p^*)$ and compute $g_1 = g^\alpha$ and choose a hash function $H : \{0,1\}^* \to G$. Output pk=((G,G$_T$),g,g_1,H) and $sk = \alpha$.
- **Trapdoor**(sk,w) : Given $sk = \alpha \in \mathbb{Z}_p$ and $w \in \{0,1\}^*$, compute $T_w = H(w)^\alpha$.

- **Enc**(pk,w) : To encrypt $w \in \{0,1\}^*$, choose $r \leftarrow U(\mathbb{Z}_p)$ and compute $c = (c_1, c_2) = (g^r, e(g_1, H(w))^r)$.

- **Test**(pk,T_w,c) : Given $c = (c_1, c_2) \in G \times G_T$, return 1 if $c_2 = e(c_1, T_w)$ and 0 otherwise.

Correctness : $e(g_1, H(w))^r = e(g^\alpha, H(w))^r = e(g^r, H(w)^\alpha)$

Théorème 1. *The scheme provides semantic security in the ROM under the Decision Bilinear Diffie-Hellman assuption.*

Proof. Let A a PEKS adversary with advantage ε we build a DBDA distinguisher with $\Omega(\frac{\varepsilon}{Q_T}$, where Q_T is the number of trapdoor queries. Algorithm B inputs (g,g^a,g^b,g^c,T) and uses A to decide if $T = e(g,g)^{abc}$ or $T \sim U(G_T)$. B defines $g_1 = g^a$ and runs A on input of $p_k = (g, g_1 = g^a, H)$ and simulates A's view.

- **H-queries** : on a query H(w_r), B returns the previously deifned value if it exists. Otherwise, B flips a based coin $\delta_{w_r} \in \{0,1\}$ such that $\mathbb{P}[\delta_{w_i} = 0] = \frac{1}{Q_T+1}$, where Q_T is the number of trapdoor queries.
  1. If $\delta_{w_i} = 0$, B returns H(w_r) = $(g^b) \cdot g^{\gamma_i}$ for a random $\gamma_i \leftarrow U(\mathbb{Z}_p)$ and keeps γ_i for later use.

  2. If $\delta_{w_i} = 1$, B returns H(w_i) = g^{γ_i} for a random $\gamma_i \leftarrow U(\mathbb{Z}_p)$ kept for later use.

- **Trapdoor queries**: When A queries T_{w_i}, we assume w.l.o.g. that H(w_i) was asked before.

1. If $\mathrm{H}(w_i) = (g^b)\cdot g^{\gamma_i}$ (ie. $\delta_{w_i} = 0$), B fails and outputs random bit.

2. If $\mathrm{H}(w_i) = g^{\gamma_i}$ (ie. $\delta_{w_i} = 1$), B returns $T_{w_i} = H(w_i)^a = (g^a)^{\gamma_i}$.

- **Challenge**: A chooses $w_0, w_1 \in \{0,1\}^*$ such that T_{w_0}, T_{w_1} were not revealed. We assume that $\mathrm{H}(w_0)$,$\mathrm{H}(w_1)$ were asked.

  1. If B replied to $\mathrm{H}(w_0)$,$\mathrm{H}(w_1)$ by setting $\delta_{w_0} = \delta_{w_1} = 1$, B fails and outputs a random bit.

  2. Let a random $d \in \{0,1\}$ such that $\delta_{w_d} = 0$. We have $\mathrm{H}(w_d) = (g^b)\cdot g^{\gamma^*}$ for some $\gamma^* \in \mathbb{Z}_p$ known to B. Then, B computes and returns $c = (g^c, T \cdot e(g^c, g^a)^{\gamma^*})$.

- **Output**: A outputs $d' \in \{0,1\}$. If $d' = d$, B returns 1 (meaning $T = e(g,g)^{abc}$) else : B returns 0 (meaning $T \sim U(G_T)$).

Let events E_1: B does not abort on Trapdoor queries and E_2 : B does not abort in Challenge phase.

Claim 1 : $\mathbb{P}[\mathrm{E}_1] \geq \frac{1}{\exp(1)}$

Proof. (claim 1) δ_{w_i} are independent and identically distributed variables with binomial distribution $\rightarrow$ $\mathbb{P}[\mathrm{E}_1] \geq (1 - \frac{1}{Q_T+1})^{Q_T} \geq \frac{1}{\exp(1)}$ □

Claim 2 : $\mathbb{P}[\mathrm{E}_2] \geq \frac{1}{Q_T}$.

If B does not abort and $T = e(g,g)^{abc}$, then $c = (g^c, e(g_1, H(w_d))^c)$ is a valid encryption of w_d. If B does not abord and $T \sim U(G_T)$, then $c \sim U(G \times G_T)$ is independent of $d \in \{0,1\}$. □

Rermak : The scheme uses and anonymity property in the Boneh-Franklin IBE.

1.3 Consistency notions

Right keyword consistency: For all $\lambda \in \mathbb{N}$, $w \in \{0,1\}^*$, $\mathbb{P}$[Test(p_k,Trapdoor(s_k,w), Enc(p_k,w))=1] =1, where proba is taken over the randomness of Keygen, Trapdoor, Enc and Test.

Perfect Consistency: For all $\lambda \in \mathbb{N}$ and distinct $w, w' \in \{0,1\}^r$, $\mathbb{P}$[Test(p_k,Trapdoor(s_k,w'), Enc(p_k,w))=1]=0 where the probability is taken over the randomness of Keygen, Trapdoor, Enc and Test.

Lemma 1. *The BDOP PEKS is not perfectly consistent.*

Proof. There exist $w, w' \in \{0,1\}^*$ such that $w \neq w'$ and $\mathrm{H}(w) = \mathrm{H}(w')$ and thus $\mathrm{H}(w)^\alpha = \mathrm{H}(w')^\alpha$. □

Computational consistency: A PEKS is computationally consistent if no PPT adversary has noticeable advantage in this game:

1. Challenger generates(pk,sk)$\leftarrow$ Keygen(1^λ), gives pk to A.

2. A chooses $w, w' \in \{0,1\}^*$, Challenger computes $c \leftarrow \text{Enc}(pk,w)$ and $T_{w'} \leftarrow \text{Trapdoor}(sk,w')$. If $w \neq w'$ and Test(pk,$T_{w'}$,c)=1, A wins. $\text{Adv}^{consist}(\text{A}) = \mathbb{P}[\text{A wins}]$.

Remark :

- Perfect consistency : Adv(A) = 0 for any unbounded A.
- Statistical consistency : Adv(A) $\leq$ negl(λ), for any unbounded A.

Théorème 2. *The BDOP PEKS is computationally consistent.*

Proof. Let $w_1, \cdots, w_{Q_H}$ the keywords queried to H($\cdot$). Let WSET = $\{w_1, \cdots, w_{Q_H}\} \cup \{w, w'\}$. Let E the event that there exist $\overline{w}, \overline{w'} \in$ WSET such that H($\overline{w}$)=H($\overline{w'}$).
$\text{Adv}^{consist}(\text{A}) = \mathbb{P}[\text{A wins} \wedge \text{E}] + \mathbb{P}[\text{A wins} \wedge \overline{E}] \leq \mathbb{P}[\text{E}] \leq \frac{(Q_H+2)^2}{|G|} < \frac{(Q_H+2)^2}{2^\lambda} \implies \text{H}(\overline{w}) \neq \text{H}(\overline{w'})$ so $\text{H}(\overline{w})^\alpha \neq \text{H}(\overline{w'})^\alpha$ and so $e(g^r, H(\overline{w})^\alpha) \neq e(g^r, H(\overline{w'})^\alpha)$. ☐

1.4 PEKS and anonymous IBE

Definition: An IBE provides anonymity (ANON-ID-CPA) if no PPT adversary has noticeable advantage of this game:

1. Challenger generates (mpk, msk) $\leftarrow$ Setup(1^λ and gives mpk to A.
2. A makes key queries: it chooses ID and obtains $SK_{ID} \leftarrow$ Keygen(msk,ID).
3. A chooses M and ID_0, ID_1 that were not submitted to Keygen(msk,$\cdot$). Challenger flips a coin $d \xleftarrow{U(\{0,1\})}$ and returns $c \leftarrow$ IBE(mpk,ID_d,M).
4. A makes more queries for identites $ID \notin \{ID_0, ID_1\}$.
5. A outputs $d' \in \{0,1\}$ and wins if $d' = d$.

Anonymous IBE implies PEKS:

Failed attempt :

- **Keygen(1^λ)** : Run (mpk,msk) $\leftarrow$ IBE.Setup(1^λ). Output (pk,sk) = (mpk,msk).
- **Trapdoor(sk,w)** : Output $T_w \leftarrow$ IBE.Keygen(msk,w).
- **Enc(pk,w)** : Compute $c \leftarrow$ IBE.Encrypt(mpk,w,0^λ).
- **Test(pk,T_w,c)** : Return 1 if IBE.Decrypt(mpk,T_w,c) = 0

Problem: Does not ensure computational consistency in general.

Solution: Encrypt a random string instead of 0^λ.

- **Keygen(1^λ)** : Run (mpk,msk) $\leftarrow$ IBE.Setup(1^λ). Output (pk,sk) = (mpk,msk).

- **Trapdoor(sk,w)**: Output $T_w \leftarrow$ IBE.Keygen(msk,w).

- **Enc(pk,w)**: To encrypt $w \in \{0,1\}^*$,

  1. Compute R $\leftarrow U(\{0,1\}^\lambda)$.
  2. Compute $c^{IBE} \leftarrow$ IBE.Encrypt(mpk,w,R).

  output $c =$(R,c^{IBE})

- **Test(pk,T_w,c)**: Given $c =$(R,c^{IBE}) and T_w, return 1 if $R =$IBE.Decrypt(mpk,T_w,c). Otherwise, return 0.

Théorème 3. *If the IBE scheme is IND-ID-CPA, the PEKS is computationally consistent.*
If the IBE scheme is ANON-ID-CPA, hte PEKS is semantically secure.

Proof. Let a consistency adversary A. We build an IND-ID-CPA adversary B against the IBE. B receives mpk from its IBE challenges and gives $pk = mpk$ to A. A outputs w, w'. B chooses $R_0, R_1 \leftarrow U(\{0,1\}^\lambda)$ and sends (w,R_0,R_1) to its challenger who replies $c^* \leftarrow$ IBE.Encrypt(mpk,w,R_d) for a random bit $d \in \{0,1\}$. B obtains $T_{w'} \leftarrow$ IBE.Keygen(msk,w') from its challenger.
If $R_1 =$ IBE.Decrypt(mpk,T'_w,c^*), then B returns 1 (guess for $d \in \{0,1\}$), else B returns 0. So, Adv(B) = $\varepsilon - 2^{-\lambda}$. □ □