Hidden Vector Encryption

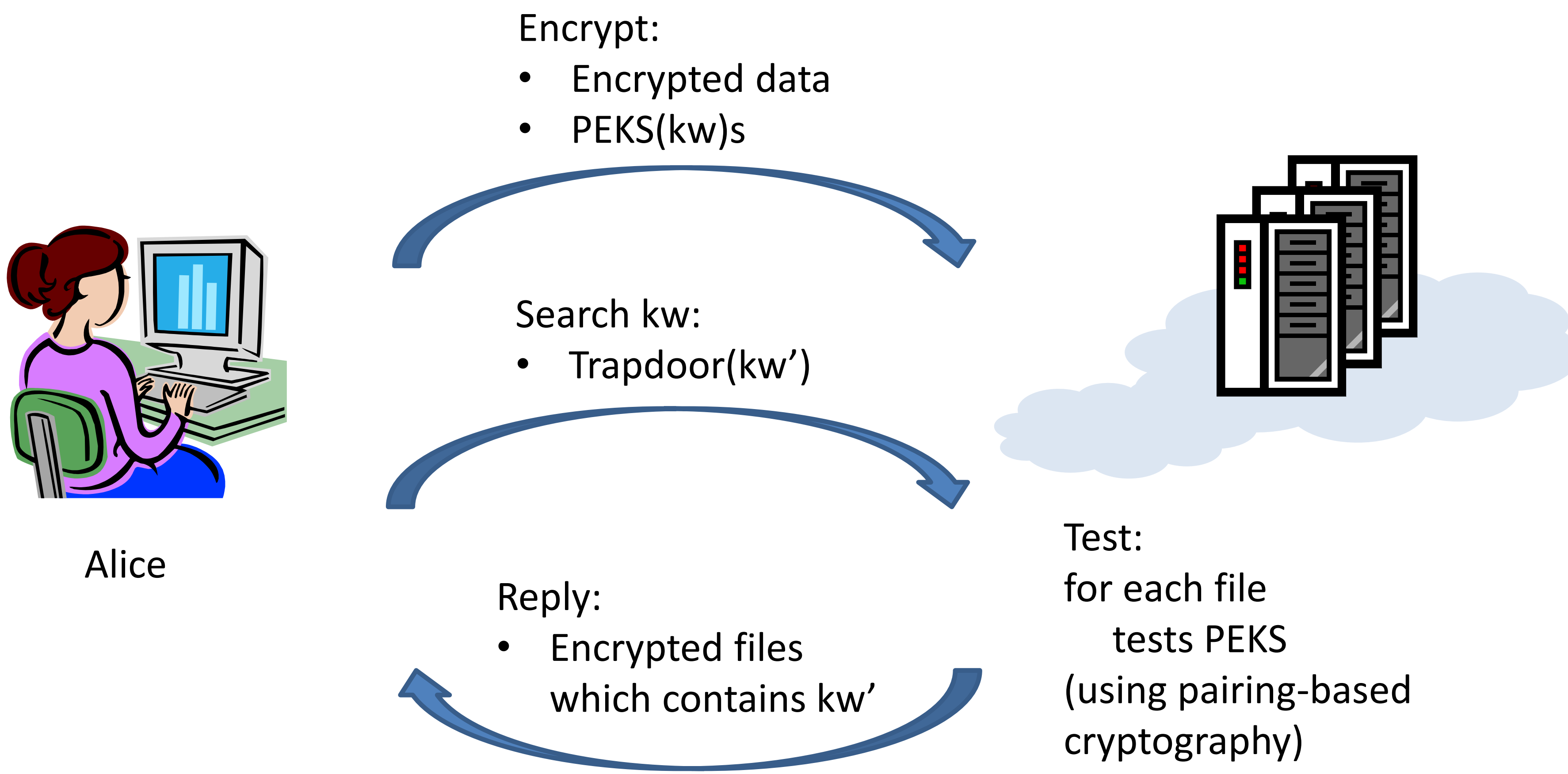
陳榮傑
交通大學資工系
Cryptanalysis Lab
9/24/2012

Searchable Encryption

- Public-key Encryption with Keyword Search (PEKS)
- Public-key Encryption with Conjunctive field Keyword search (PECK)
- Hidden Vector Encryption (HVE)

PEKS

kw : keyword



Server gains no knowledge about kw or the file content stored on the Cloud Storage

Search Keyword

PEKS INDEX

File 1:

- PEKS(id)
- PEKS(title)
- PEKS(author1)
- PEKS(author2)
- ⋮

Trapdoor(author: Brent Waters)

File 2:

- PEKS(id)
- PEKS(title)
- PEKS(author)
- PEKS(keyword1)
- PEKS(keyword2)
- ⋮

Trapdoor(author: Brent Waters)

$$PEKS = (g^r, H_2(t)), \qquad t = e(H_1(KW), h^r), \qquad h = g^\alpha$$

$$Trapdoor = H_1(KW)^\alpha$$

$$Server\ tests\ each\ PEKS\ whether \quad H_2(e(H_1(KW)^\alpha, g^r) = H_2(t)$$

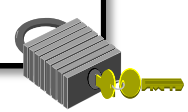
Disadvantages of PEKS

- Consider a system containing *n* documents, each of which has at most *s* keywords
  - Efficiency
    - One keyword search compares at most *ns* times.
  - Limited search capability
    - Conjunctive searches will leak information.
  - Traceable trapdoor
    - A trapdoor of a specific keyword is fixed.

Searchable Encryption – PECK

Email

Date: 2012/09/24
From: assist@cs.nctu.edu.tw
To: cloud@delta.com.tw
Subject: Delta



PECK(date, from, to, subject)

PECK	date	from	to	subject

Conjunctive equality search for Email:

1. Equality search
   Is Date = 2012/09/24 ?
2. Equality search
   Is From = assist@cs.nctu.edu.tw ?
3. Equality search
   Is Subject = Delta ?

If PECK matches Trapdoor,
then the encrypted email is our target.

Trapdoor(1, 2, 4, date, from, subject)

Trapdoor	..	…	…

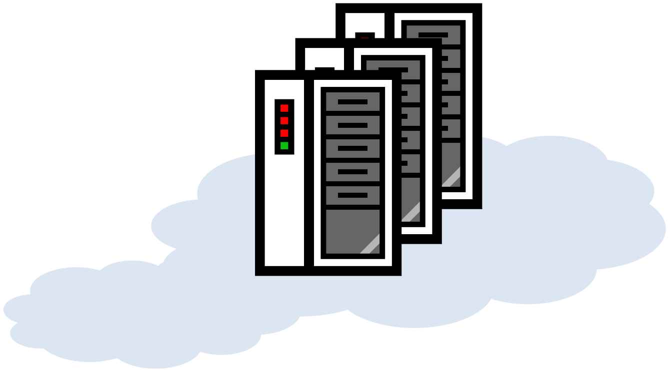
PECK, a variation of PEKS

s: # keyword fields
W_i: the keyword in i^{th} field

Encrypt:
- Encrypted data
- PECK(W_1, … W_s)

Alice

Search $\wedge_{j=1}^{t}(W_{I_j} = \Omega_j)$:
- Trapdoor($\Omega_1, \dots, \Omega_t$)



Reply:
- Encrypted files which contains $\Omega_1, \dots, \Omega_t$

Test:
for each file
tests PECK
(using pairing-based cryptography)

D. J. Park, K. Kim, and P. J. Lee. (2004) Public Key Encryption with Conjunctive Field Keyword Search

Search Keyword in PECK

PECK INDEX

File 1:
PECK(from, to, date, subject) → Trapdoor(1, 4, "assist", "Delta")

File 2:
PECK(from, to, date, subject) → Trapdoor(1, 4, "assist", "Delta")

$$PECK = [\hat{e}(rH(W_1), Y_1), \dots, \hat{e}(rH(W_s), Y_1), rY_2, rP], \qquad Y_1 = s_1P, Y_2 = s_2P$$

$$Trapdoor = [T_1, T_2, I_1, \dots, I_t], \qquad T_1 = \frac{s_1}{s_2 + u}\left(H(\Omega_1) + \cdots + H(\Omega_t)\right), T_2 = u$$

$$Server\ tests\ each\ PECK\ \ if \quad \prod_{i=1}^{t} \hat{e}\left(rH\left(W_{I_j}\right), Y_1\right) = \hat{e}(T_1, rY_2 + T_2rP)$$

PECK – Pros & Cons

- Consider a system containing *n* documents, each of which has exactly *s* keyword fields
- Advantages
  - One keyword search compares *n* times.
  - Conjunctive searches will leak nothing except the searching fields.
  - A random number is added to generate the trapdoor.
- Disadvantages
  - The keyword fields are predefined and thus fixed.
  - Only the equality search is supported.

Hidden Vector Encryption

To enhance search capability

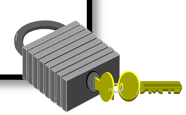
Query Type	
Equality query: $(x_i = a)$	for any $a \in T$
Comparison query: $(x_i \geq a)$	for any $a \in T$
Subset query: $(x_i \in A)$	for any $A \subseteq T$
Equality conjunction: $(x_1 = a_1) \wedge \ldots \wedge (x_w = a_w)$	
Comparison conjunction: $(x_1 \geq a_1) \wedge \ldots \wedge (x_w \geq a_w)$	
Subset conjunction: $(x_1 \in A_1) \wedge \ldots \wedge (x_w \in A_w)$	

D. Boneh and B. Waters. (2007) Conjunctive, Subset, and Range Queries on Encrypted Data

Searchable Encryption – HVE

<u>Classified Document</u>

Classification	2
Year	2008
Author	CWHsieh



Queries for classified documents:

1. Comparison query
   Is Classification < 3 ?
2. Range query
   2006 < Year ≤ 2012 ?
3. Subset query
   Is author one of
   { RJChen,
   CWHsieh,
   LTTsai } ?

If X matches Y, then the encrypted classified document is our target document.

Hidden Attribute Vector X

X	2	2008	CWHsieh

Hidden Attribute Vector Y

Y			

HVE (1/8)

- Predicate function: $P_{\boldsymbol{Y}}(\boldsymbol{X})$

Suppose there are two attribute vectors: $\boldsymbol{X}, \boldsymbol{Y}$ of length $\ell = 5$

	1	2	3	4	5	
$\boldsymbol{X}$	0	1	4	8	6	← Specific values
$\boldsymbol{Y}$	*	*	4	*	*	← * indicate "DON'T CARE"s

$$\Rightarrow P_{\boldsymbol{Y}}(\boldsymbol{X}) = \begin{cases} 1, & X_i = Y_i \; for \; all \; Y_i \neq *, \\ 0, & otherwise. \end{cases}$$

HVE (2/8)

X : attribute

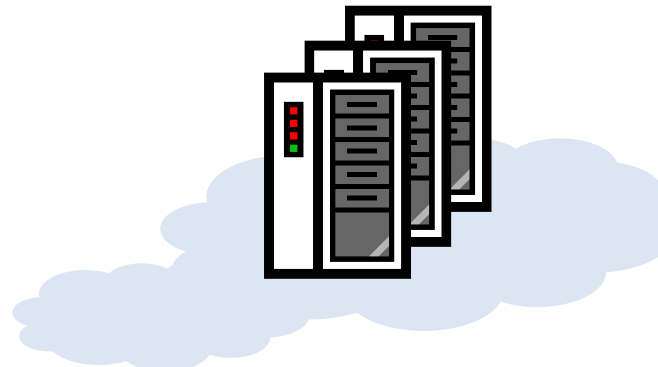
Encrypt:

- Encrypted data
- HVE(X)

Alice

Search Y:

- Trapdoor(Y)



Test:
for each file
tests if $P_{\boldsymbol{Y}}(\boldsymbol{X}) = 1$
(using pairing-based cryptography)

Reply:

- Decrypted files whose HVE $\boldsymbol{X}$ satisfies $P_{\boldsymbol{Y}}(\boldsymbol{X}) = 1$

HVE (3/8)

- Comparison with PECK
- The same as PECK
  - One keyword search compares *n* times.
  - A random number is added to generate the trapdoor.
  - The keyword fields are predefined and thus fixed.
  - Conjunctive searches will leak nothing except the searching fields.
- Advantages
  - Conjunctive, subset, and range searches are supported.
- Disadvantages
  - Search process will decrypt the encrypted data

HVE (4/8)

- Equality(Y_1) / Conjunctive Equality(Y_2)

	1	2	3	4	5
X_1	0	1	4	8	6
X_2	1	3	2	8	5
X_3	0	2	3	8	6
X_4	2	3	1	4	6
Y_1	*	3	*	*	*
Y_2	0	*	*	8	*

HVE (5/8)

- Subset
  - Documents are separated into categories
    - 1: Art
    - 2: Engineering
    - 3: Financial
    - 4: Humanities
    - 5: Novel
  - Each document belongs to one category
  - Search documents in “Art” or “Humanities”

	1	2	3	4	5
X_1	0	1	0	0	0
X_2	0	0	1	0	0
X_3	0	0	0	0	1
X_4	1	0	0	0	0
Y	*	0	0	*	0

HVE (6/8)

- Comparison
  - Documents have an attribute of type integer, such as year
    - 1: 2007
    - 2: 2008
    - 3: 2009
    - 4: 2010
    - 5: 2011
    - 6: 2012
    - 7: 2013
  - Search documents whose $year \leq 2010$

	1	2	3	4	5	6	7		
X_1	0	1	1	1	1	1	1	←	(2008)
X_2	0	0	0	0	0	1	1		(2012)
X_3	0	0	0	1	1	1	1	←	(2010)
X_4	0	0	0	0	0	0	1		(2013)
Y	*	*	*	1	*	*	*		

HVE (7/8)

- Range
  - Documents have an attribute of type integer, such as year
    - 1: 2007
    - 2: 2008
    - 3: 2009
    - 4: 2010
    - 5: 2011
    - 6: 2012
    - 7: 2013
  - Search documents whose $2010 \leq year \leq 2012$

or $2009 < year \leq 2012$

	1	2	3	4	5	6	7	
X_1	0	1	1	1	1	1	1	(2008)
X_2	0	0	0	0	0	1	1	← (2012)
X_3	0	0	0	1	1	1	1	← (2010)
X_4	0	0	0	0	0	0	1	(2013)
Y	*	*	0	*	*	1	*	

HVE (8/8)

- HVE supports
  - Equality search
  - Conjunctive equality search
  - Subset search
  - Comparison search
  - Range search

  - Conjunctive of equality, subset, and range searches

HVE details (1/2)

Setup(λ) The setup algorithm first chooses random primes $p, q > m$ and creates a bilinear group $\mathbb{G}$ of composite order $n = pq$, as specified in Section 4.1. Next, it picks random elements

$$(u_1, h_1, w_1), \ldots, (u_\ell, h_\ell, w_\ell) \in \mathbb{G}_p^3 , \qquad g, v \in \mathbb{G}_p , \qquad g_q \in \mathbb{G}_q.$$

and an exponent $\alpha \in \mathbb{Z}_p$. It keeps all these as the secret key SK.

It then chooses $3\ell + 1$ random blinding factors in $\mathbb{G}_q$:

$$(R_{u,1}, R_{h,1}, R_{w,1}), \ldots, (R_{u,\ell}, R_{h,\ell}, R_{w,\ell}) \in \mathbb{G}_q \text{ and } R_v \in \mathbb{G}_q.$$

For the public key, PK, it publishes the description of the group $\mathbb{G}$ and the values

$$g_q, \quad V = vR_v, \quad A = e(g, v)^\alpha, \quad \begin{pmatrix} U_1 = u_1 R_{u,1}, & H_1 = h_1 R_{h,1}, & W_1 = w_1 R_{w,1} \\ & \vdots & \\ U_\ell = u_\ell R_{u,\ell}, & H_\ell = h_\ell R_{h,\ell}, & W_\ell = w_\ell R_{w,\ell} \end{pmatrix}$$

The message space $\mathcal{M}$ is set to be a subset of $\mathbb{G}_T$ of size less than $n^{1/4}$.

HVE details (2/2)

Encrypt(PK, $\mathcal{I} \in \mathbb{Z}_m^\ell$, $M \in \mathcal{M} \subseteq \mathbb{G}_T$) Let $\mathcal{I} = (\mathcal{I}_1, \ldots, \mathcal{I}_\ell) \in \mathbb{Z}_m^\ell$. The encryption algorithm works as follows:

- choose a random $s \in \mathbb{Z}_n$ and random Z, $(Z_{1,1}, Z_{1,2}), \ldots, (Z_{\ell,1}, Z_{\ell,2}) \in \mathbb{G}_q$. (The algorithm picks random elements in $\mathbb{G}_q$ by raising g_q to random exponents from $\mathbb{Z}_n$.)
- Output the ciphertext:

$$C = \left(C' = MA^s, \quad C_0 = V^s Z, \quad \begin{pmatrix} C_{1,1} = (U_1^{\mathcal{I}_1} H_1)^s Z_{1,1}, & C_{1,2} = W_1^s Z_{1,2} \\ \vdots & \\ C_{\ell,1} = (U_\ell^{\mathcal{I}_\ell} H_\ell)^s Z_{\ell,1}, & C_{\ell,2} = W_\ell^s Z_{\ell,2} \end{pmatrix} \right)$$

GenToken(SK, $\mathcal{I}_* \in \Sigma_*^\ell$) The key generation algorithm will take as input the secret key and an ℓ-tuple $\mathcal{I}_* = (\mathcal{I}_1, \ldots, \mathcal{I}_\ell) \in \{\mathbb{Z}_m \cup \{*\}\}^\ell$. Let S be the set of all indexes i such that $\mathcal{I}_i \neq *$. To generate a token for the predicate $P_{\mathcal{I}_*}^{\mathrm{HVE}}$ choose random $(r_{i,1}, r_{i,2}) \in \mathbb{Z}_p^2$ for all $i \in S$ and output:

$$\mathrm{TK} = \left(\mathcal{I}_*, \quad K_0 = g^\alpha \prod_{i \in S} (u_i^{\mathcal{I}_i} h_i)^{r_{i,1}} w_i^{r_{i,2}}, \quad \forall i \in S: \quad K_{i,1} = v^{r_{i,1}}, \quad K_{i,2} = v^{r_{i,2}} \right)$$

Query(TK, C) Using the notation in the description of *Encrypt* and *GenToken* do:

- First, compute

$$M \leftarrow C' \;/\; \left(e(C_0, K_0) \;/\; \prod_{i \in S} e(C_{i,1}, K_{i,1})\; e(C_{i,2}, K_{i,2}) \right) \tag{3}$$

- If $M \notin \mathcal{M}$ output $\perp$. Otherwise, output M.