

# Introduction to Attribute Based Searchable Encryption

Dalia Khader

► **To cite this version:**

Dalia Khader. Introduction to Attribute Based Searchable Encryption. 15th IFIP International Conference on Communications and Multimedia Security (CMS), Sep 2014, Aveiro, Portugal. pp.131-135, 10.1007/978-3-662-44885-4\_11 . hal-01404203

**HAL Id: hal-01404203**

**<https://hal.inria.fr/hal-01404203>**

Submitted on 28 Nov 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



# Introduction to Attribute Based Searchable Encryption

Dalia Khader  
daliakhader@googlemail.com

**Abstract.** An Attribute Based Searchable Encryption Scheme (ABSE) is a public key encryption with keyword search (PEKS) where each user owns a set of attributes, and the senders decide on a policy. The policy is a function of these attributes expressed as a predicate and determines, among the users of the system, who is eligible to decrypt and search the ciphertext. Only members who own sufficient attributes to satisfy that policy can send the server a valid search query. In our work we introduce the concept of a secure ABSE by defining the functionalities and the relevant security notions.

**Keywords:** PEKS, Attribute Based Encryption, Public Key Cryptography

## 1 Introduction

Searchable encryption (SE) is an encryption scheme that supports keyword based retrieval of documents. The main challenge of SE is to allow third parties to search the ciphertexts without giving them decrypting capabilities. This has been an active research area for more than a decade. Song et al. [5] proposed the first scheme that enables searchability in symmetric encryption while Boneh et al. [2] introduced a scheme for public key encryption with keyword search (PEKS).

Searchable encryption schemes assume that the user sending the search query owns the decryption key and that the sender has to know the identity of the user querying the data in order to encrypt using the corresponding encryption key. This raises the question, what if the encrypted data is shared between several receivers and is kept in a remote shared storage that is not trusted for confidentiality?

Attribute-Based Encryption (ABE) [4] addresses this problem. An ABE is a scheme in which each user is identified by a set of attributes, and some function of those attributes, the policy, is used to decide on decryption capabilities. The two types of ABE schemes are: key-policy and ciphertext-policy [3, 1].

This paper defines a new primitive attribute based searchable encryption (ABSE). In ABSE senders decide on a policy that determines user's eligibility not only for decrypting but also for searching the data. Unlike existing proposals in the literature [6], ours is based on a hybrid system of key and cipher policy which gives more flexibility, a strong security, and allows for multi-authorities.

## 2 Formal Definition of ABSE

To define an ABSE we introduce the five entities involved: a central authority  $\mathcal{TTP}$  who sets up the system, a server  $\mathcal{S}$  where all encrypted data is uploaded to. An encryptor  $\mathcal{E}$  who uploads the data and sets the policy. The querier  $\mathcal{Q}$  who wants to search the server and download documents. Many attribute authorities  $\mathcal{AT}$  each responsible of a set of attributes and that give out private keys to users owning these attributes.

**Definition 1.** An Attribute Based Searchable Encryption Scheme consists of the following probabilistic polynomial time algorithms:  $\mathbf{ABSE} := (TSetup, AddUser, ASetup, AttG, PrdG, PrdVK, PrdQT, ABSE, TrpG, TEST)$

$\mathbf{TSetup}(k) \rightarrow (\mathbf{PP}, \mathbf{UMK})$ : Run by $\mathcal{TTP}$ to set up the system. Takes a security parameter $k$ and outputs public parameters $PP$ and a user master key $UMK$ which is kept secret.
$\mathbf{AddUser}(\mathbf{PP}, \mathbf{UMK}) \rightarrow (\mathbf{RK}_i, \mathbf{SK}_i)$ : Run by $\mathcal{TTP}$ every time a user registers with the system. It outputs a registration key $RK_i$ that will be used to register with attribute authorities and servers. It outputs $SK_i$ that is secret to the user and will be used in creating trapdoors.
$\mathbf{ASetup}(\mathbf{PP}) \rightarrow (\mathbf{AMK}_j, \mathbf{APK}_j)$ : Run by $\mathcal{AT}$ to set up the attribute authority. It outputs an attribute master key $AMK_j$ which is secret to $\mathcal{AT}$ and is used to create attribute private keys when users register. It also outputs an attribute public key $APK_j$ which is used in building the policies and is public to all.
$\mathbf{AttG}(\mathbf{RK}_i, \mathbf{AMK}_j) \rightarrow \mathbf{ASK}_{i,j}$ : Run by $\mathcal{AT}$ to register a user $i$ , and outputs an attribute private key $ASK_{i,j}$ that will be used in proving possession of attribute $j$ .
$\mathbf{PrdG}(\Psi, \mathcal{AP}) \rightarrow (\mathbf{ST}_\Psi, \mathbf{IT}_\Psi)$ : Given a predicate $\Psi$ and a list of attribute public keys $\mathcal{AP} = \{APK_j\}_{j=1}^m$ , the algorithm generates a searching token $ST_\Psi$ that will be used in creating trapdoors and an indexing token $IT_\Psi$ used for creating searchable ciphertext.
$\mathbf{PrdVK}(\Psi) \rightarrow \mathbf{VT}_\Psi$ : Run by $\mathcal{S}$ . For each predicate in the system the server creates a verification token $VT_\Psi$ that is kept secret to the server.
$\mathbf{PrdQT}(\Psi, \mathbf{VT}_\Psi, \mathbf{RK}_i, \mathcal{AP}) \rightarrow \mathbf{QT}_{i,\Psi}$ : Run by the $\mathcal{S}$ . Given a predicate verification token $VT_\Psi$ and a registration key $RK_i$ , the server outputs a query token $QT_{i,\Psi}$ that allows the user $i$ to search for keywords encrypted under the predicate $\Psi$ .
$\mathbf{ABSE}(\mathbf{W}, \Psi, \mathbf{IT}_\Psi) \rightarrow \mathbf{E}_{\Psi, \mathbf{W}}$ : Run by $\mathcal{E}$ . For a keyword $W$ and under token $IT_\Psi$ create a searchable ciphertext $E_{\Psi, W}$ .
$\mathbf{TrpG}(\mathbf{W}, \Psi, \mathbf{QT}_{i,\Psi}, \mathbf{ST}_\Psi, \mathbf{SK}_i, \mathcal{AS}_i) \rightarrow \mathbf{T}_{\Psi, \mathbf{W}}$ : Run by $\mathcal{Q}$ . Given a keyword, a query token, a searching token, a user secret key and a set of user private attribute keys $\mathcal{AS}_i = \{ASK_{i,j}\}_{j=1}^m$ , output a trapdoor $T_{\Psi, W}$ .
$\mathbf{TEST}(\mathbf{E}_{\Psi, \mathbf{W}}, \mathbf{T}_{\Psi, \mathbf{W}}, \mathbf{VT}_\Psi, \mathbf{RK}_i) \rightarrow \{0, 1\}$ : Run by the $\mathcal{S}$ . Given a searchable ciphertext, a trapdoor, a verification token and a registration key output 1 if the user satisfies the predicate and if the keyword is found, otherwise output 0.

**On The Security of ABSE** The security notions of an ABSE are: correctness, security against Attribute Based Chosen Keyword Attack (ACKA) and security against Attributes Forgeability Attacks (AFA). We need three game models to define these notions (See Figures 1(a), 1(b), 1(c)) where the adversary is given access to certain oracles and a trace of the responses is recorded. Both are explained below.

$CUL$ Corrupted Users	$HUL$ Honest Users	$CRK$ Corrupted $RK_i$
$CRK$ Corrupted $RK_i$	$HA$ Honest $\mathcal{AT}$	$CA$ Corrupted $\mathcal{AT}$
$CASK$ Revealed $ASK_{i,j}$	$TrapL$ Queried trapdoors	$HASK$ Non-revealed $ASK_{i,j}$
$PredL$ List of $(ST_\Psi, IT_\Psi)$	$RQTL$ Revealed $QT_{i,\Psi}$	$QTL$ Non-revealed $QT_{i,\Psi}$
$VTL$ Non-revealed $VT_\Psi$	$RVTL$ Revealed $VT_\Psi$	

**AddUsr** : Adds user  $i$  to the system by running  $AddUser$ , and adding  $(RK_i, SK_i)$  to  $HUL$ .

**UsrCpt** : Corrupts user  $i$  by revealing  $(RK_i, SK_i)$  and adding them to  $CUL$  and  $CRK$ .

**RKCpt** : Partially corrupts user  $i$  by revealing registration key  $RK_i$  and adding it to  $CRK$ .

**AddAtt** : Adds an honest attribute authority  $j$  to the system by running  $ASetup$ , computing  $(APK_j, AMK_j)$  and publishing  $APK_j$ .

**AMKCpt** : Corrupts attribute authority  $j$  by revealing  $AMK_j$  and adding to  $CA$ .

**AddASK** : Runs  $AttG$  to compute  $ASK_{i,j}$ , and adds it to  $HASK$ .

**ASKCpt** : Corrupts an attribute private key by revealing  $ASK_{i,j}$  and adding it to  $CASK$ .

**TrapO** : The challenger generates a trapdoor for certain keyword  $W$  using a querying token  $QT_{i,\Psi}$  and searchable token  $ST_\Psi$  on behalf of user  $i$  with set of attributes  $\mathcal{AS}_i$ . The list  $TrapL$  is updated with all the information used as input and as output to the algorithm  $TrpG$ .

**AddPred** : Generates a searchable token and an indexing token  $(ST_\Psi, IT_\Psi)$  for predicate  $\Psi$  by running  $PrdG$  and then updates the list  $PredL$ .

**AddVT** : Runs  $PrdVK$  to obtain a verification token  $VT_\Psi$  and updates  $VTL$ .

**RevealVT** : Corrupting the verification token  $VT_\Psi$  by revealing it and  $RVTL$  is updated.

**AddQT** : Generates a querying token  $QT_{i,\Psi}$  by running  $PrdQT$ , then  $QTL$  is updated.

**RevealQT** : The querying token  $QT_{i,\Psi}$  of user  $i$  and predicate  $\Psi$  is revealed to the adversary and the list  $RQTL$  is updated.

**Ch<sub>b</sub>** : Challenges the adversary to guess whether a trapdoor  $T_b$  ( $b \in \{0, 1\}$ ) was generated for keyword  $W_0$  or  $W_1$ . The adversary chooses the predicate, the set of attributes and the user he would like to be challenged upon.

**Correctness of ABSE.** This property demands that if a searchable encryption  $E_{\Psi,W}$  was produced correctly, i.e. using valid  $IT_\Psi$  and if a trapdoor  $T_{\Psi,W}$  was introduced correctly using valid  $QT_{i,\Psi}$ ,  $ST_\Psi$ ,  $SK_i$ ,  $\mathcal{AS}_i$ , then the  $TEST$  algorithm should return 1 if the predicate is satisfied  $\Psi(\mathcal{AS}_i) = 1$  and the keywords match  $W = W'$ , otherwise the  $TEST$  algorithm should return 0. Figure 1(a) explains the details. Formally, the ABSE is said to be correct if for a security parameter  $k$  and all polynomial time adversaries  $\mathcal{A}$  the following advantage is negligible:  $\text{Adv}_{\mathcal{A}}^{\text{corr}}(k) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{corr}}(k) = 1]|$

**Attribute based Chosen Keyword Attacks.** We define security for an ABSE in the sense of semantic-security. The aim is to ensure that an encryption  $ABSE$  does not reveal any information about keyword  $W$  except to a  $\mathcal{Q}$  who satisfies the policy and can create trapdoors. We define the security against an active attackers  $\mathcal{A}$  whose given access to a set of oracles shown in Figure 1(b). Let the advantage of winning the game be defined as follows:  $\text{Adv}_{\mathcal{A}}^{\text{ACKA}}(k) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{ACKA}_0}(k) = 1] - \Pr[\text{Exp}_{\mathcal{A}}^{\text{ACKA}_1}(k) = 1]|$

An ABSE scheme is said to be secure against an ACKA if for a given security parameter  $k$  and all polynomial time adversary  $\mathcal{A}$  the advantage  $\text{Adv}_{\mathcal{A}}^{\text{ACKA}}(k)$  is negligible.

**Attribute Forgeability Attack.** This security notion captures forgeability of trapdoors where the adversary can produce a trapdoor without having the sufficient attribute set that satisfies the predicate  $\Psi$ . The adversary is given access to the oracles described in Figure 1(c). The challenge is to produce a pair of searchable encryption  $E_{\Psi'}^*$  and trapdoor  $T_{\Psi'}^*$  under predicate  $\Psi'$  such that the  $TEST(E_{\Psi'}^*, T_{\Psi'}^*, VT_{\Psi'}, RK'_i)$  outputs 1 for a given  $RK'_i$ . The definition includes coalition of attributes. Formally, an ABSE scheme is said to be secure against an AFA if for a security parameter  $k$  and all polynomial time adversaries  $\mathcal{A}$  the following advantage is negligible:  $\text{Adv}_{\mathcal{A}}^{\text{AFA}}(k) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{AFA}}(k) = 1]|$

### 3 Conclusion

We define a new ABSE scheme and the security notions required. A working construction and security proofs are provided in a full version of this paper.

<p>Experiment <math>\text{Exp}_{\mathcal{A}}^{\text{cor}}(k)</math>:</p> <ul style="list-style-type: none"> <li>• <math>(PP, UMK) \leftarrow TSetup(k)</math></li> <li>• <math>HUL, HA, PredL, HASK, QTL, VTL = \phi</math></li> <li>• <math>(SK_i, RK_i, AS_i, QT_{i,\psi}, ST_{\psi}, IT_{\psi}) \leftarrow \mathcal{A}(PP : AddUsrc(\cdot), AddAtt(\cdot), AddASK(\cdot, \cdot), AddPred(\cdot, \cdot), AddQT(\cdot, \cdot), AddVT(\cdot))</math></li> <li>• If <math>[(SK_i, RK_i) \notin HUL] \vee [\exists j \in \Psi \text{ s.t. } (AMK_j, APK_j) \notin HA] \vee [\exists j \in AS_i \text{ s.t. } (ASK_{i,j}) \notin HASK] \vee [VT_{\psi} \notin VTL]</math>: Return 0</li> <li>• <math>ABSE(W, IT_{\psi}) \rightarrow E_{\psi, W}; TrpG(W', QT_{i,\psi}, ST_{\psi}, SK_i, AS_i) \rightarrow T_{\psi, W}</math></li> <li>• If <math>[[\Psi(AS_i) \neq 1] \vee [W \neq W']] \wedge [TEST(E_{\psi, W}, T_{\psi, W}, VT_{\psi}, RK_i) = 0]</math>: Return 1</li> <li>• If <math>[[\Psi(AS_i) = 1] \wedge [W = W']] \wedge [TEST(E_{\psi, W}, T_{\psi, W}, VT_{\psi}, RK_i) = 1]</math>: Return 1</li> <li>• Else Return 0</li> </ul>
---

(a) Correctness Game Model

<p>Experiment <math>\text{Exp}_{\mathcal{A}}^{\text{ACKA}}(k)</math>:</p> <ul style="list-style-type: none"> <li>• <math>(PP, UMK) \leftarrow TSetup(k)</math></li> <li>• <math>CUL, HUL, CRK, HA, CA, CASK, HASK, TrapL, PredL, QTL, RQTL, RVTL, VTL = \phi</math></li> <li>• <math>\hat{b} \leftarrow \mathcal{A}(PP : Usrcpt(\cdot), RKCpt(\cdot), AMKCpt(\cdot), ASKCpt(\cdot, \cdot), RevealQT(\cdot, \cdot), AddVT(\cdot), RevealVT(\cdot), TrapO(\cdot, \cdot, \cdot, \cdot, \cdot), AddUsrc(\cdot), AddAtt(\cdot), AddPred(\cdot, \cdot), AddQT(\cdot, \cdot), AddASK(\cdot, \cdot), Ch(\cdot, \cdot, \cdot, \cdot, \cdot))</math></li> <li>• Return <math>\hat{b}</math></li> </ul>
--

(b) Security against ACKA

<p>Experiment <math>\text{Exp}_{\mathcal{A}}^{\text{AFA}}(k)</math>:</p> <ul style="list-style-type: none"> <li>• <math>(PP, UMK) \leftarrow TSetup(k)</math></li> <li>• <math>CUL, HUL, CRK, HA, CA, CASK, HASK, TrapL, PredL, QTL, RQTL, RVTL, VTL = \phi</math></li> <li>• <math>(T_{\psi'}^*, E_{\psi'}^*, \Psi', RK_i')</math> <math>\leftarrow \mathcal{A}(PP : Usrcpt(\cdot), RKCpt(\cdot), AMKCpt(\cdot), ASKCpt(\cdot, \cdot), RevealQT(\cdot, \cdot), AddVT(\cdot), RevealVT(\cdot), TrapO(\cdot, \cdot, \cdot, \cdot, \cdot), AddUsrc(\cdot), AddAtt(\cdot), AddPred(\cdot, \cdot), AddQT(\cdot, \cdot), AddASK(\cdot, \cdot))</math></li> <li>• If <math>[TEST(E_{\psi'}^*, T_{\psi'}^*, VT_{\psi'}, RK_i') = 0] \vee [\Psi' \subseteq CA] \vee [T_{\psi'}^* \in TrapL] \vee [[VT_{\psi'} \in RVTL] \wedge [\forall j \in \Psi', j \in CASK \cup CA] \vee [[VT_{\psi'} \notin RVTL] \wedge [\exists i \text{ s.t. } \forall j \in AS_i, j \in CASK_i \cup CA \text{ and } RK_i' = RK_i]]]</math>: Return 0.</li> <li>• Else Return 1</li> </ul>
---

(c) Security against AFA

## References

1. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE SSP'07*, pages 321–334.
2. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT'04*, pages 506–522.
3. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS'06*, pages 89–98.
4. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT'05*. LNCS.
5. D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SSP'00*. IEEE.
6. Q. Zheng, S. Xu, and G. Ateniese. Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. *IACR ePrint*.