Survey of Attribute-Based Encryption

D. Boatman[1]
[1]Computer Science and Engineering, Arkansas State University, Jonesboro, Arkansas, USA

Abstract - *The sole purpose of a cryptosystem is to provide a high level of security, as well as adequate time efficiency on complex data. The most expressive form of security would be access control, but symmetric encryption algorithms are not able to support this. Attribute-base encryption allows for the encryption and decryption of data in a role-based fashion, allowing only the user of adequate role to view the data. There are two forms of attribute-based encryption policies: key-policy and ciphertext policy. In which key-policy allows the access policy is mapped onto the user's secret key and ciphertext-policy allows the ciphertext is mapped onto the access policy. An analysis through time is given, starting from the first whisperings of attribute-based systems to the current most efficient version.*

Keywords: Attribute, Role-based, Bilinear, Encryption, Access Structure

1 Introduction

The sole purpose of a cryptosystem is to provide a high level of security, as well as adequate time efficiency on complex data. Private key, or asymmetric, cryptography provides just that.

Although, there is one issue presented. What if you want to change the access requirements for a given system? You have a few options of manually doing this, but they are not relatively appealing. Your first option would be to assign yourself as a mediator, which would entail decrypting all the files that you are wanting to share and sending them through a secure channel to the intended receiver. This would work, but it would become a tedious task.

Another option would be to share your private key with the person you are wanting to allow decryption. The negative side-effect of this is that now the receiver has access to all your encrypted data!

A more appealing form would be to allow a set of attributes, or roles that can decrypt data based on the role. Attribute-based encryption sets to solve this issue in a computationally efficient and provably secure manner. In this system, the door is open to incredibly expressive constructions which allow for different levels of security. For example, you could create multiple different keys that decrypt the same message, and if the user trying to decrypt does not have the correct roles to decrypt, the message cannot be found.

In this paper, the background of the initial findings of ABE, which was built off Identity-based encryption, is explained. From there the mathematical background that is needed to understand the fundamentals of each algorithm is displayed. A detailed analysis of four of the main constructions in this system, starting with the first proposal by Shamir[1] which begged the question of the feasibility of this type of cryptosystem. The constructions then follow a chronological order of new research findings, up until the most efficient proposal given by Waters [3]. Following this, the topic of collusion is more closely analyzed.

2 Background

Before approaching attribute-based encryption, there is an abundance of pre-requisite information. One should understand the fundamentals of asymmetric encryption, as well as fuzzy identity-based encryption which laid the foundation for the beginnings of attribute-based encryption.

As well as the above, one must understand the mathematical foundations of which ABE is built upon, such as bilinear mapping, access structures, and linear secret sharing schemes.

2.1 Symmetric cryptosystems

The basis for symmetric encryption is the fact that there is only one key throughout the entire encryption process. This secret key must be used to encrypt a given plaintext, then then this key must then be sent out alongside the ciphertext to the receiver who must use the same key to decrypt the message.

For example, let Alice hold a plaintext message for which she wants to send to Bob. However, she only wants Bob to be able to read this message. She will use a symmetric encryption algorithm and input one secret key that only she currently has access to. She will then send the encrypted ciphertext to Bob, along with the secret key that she used to encrypt. Bob will then use the reverse of the algorithm that Alice to encrypt, in order to decrypt his message. Bob will not have access to the message. If Bob wanted to send the message back to Alice, he could subsequently follow the same process that Alice used.

One benefit of this type of encryption is its speed and efficiency. You only ever need to hold a minimum of one key to send across the functions of encryption and decryption. Also, the decryption process is the exact reverse of the encryption process. This makes for a simplistic algorithm.

A drawback to this type of system is that there is only one secret key, and this key must be shared among all parties for which they need to decrypt. The key must be sent through a provably secure channel, otherwise the sender risks the chance of an attacker stealing the key and subsequently gaining access to all the encrypted messages with that key.

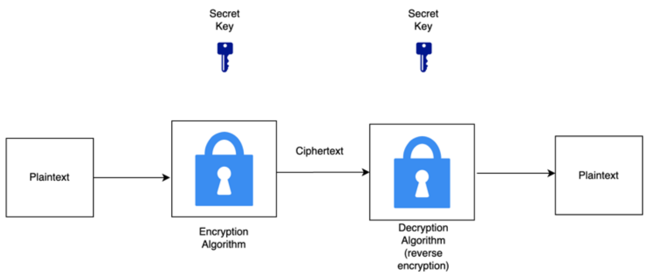

Figure 1: Symmetric Encryption

2.2 Asymmetric cryptosystems

The basic idea for asymmetric encryption is that there is one key for encryption and a separate key for decryption. There are a few assumptions when putting together an asymmetric cryptosystem, one being that an attacker should not be able to find the decrypting key with simply the encryption algorithm and the encrypting key. There must also be separate algorithms for encryption and decryption, along with a set of keys that are used for encryption and decryption without overlap. Lastly, the sender and receiver must not share any matching keys.

Encryption in this system can be described as the following: Let Bob have a plaintext message of which he wants to send to Alice. Additionally, he only wants Alice to be able to decrypt this message. Bob achieves this by using Alice's public key to encrypt his message, then he transmits the message through a channel to Alice. Alice then users her private key to decrypt the plaintext message.

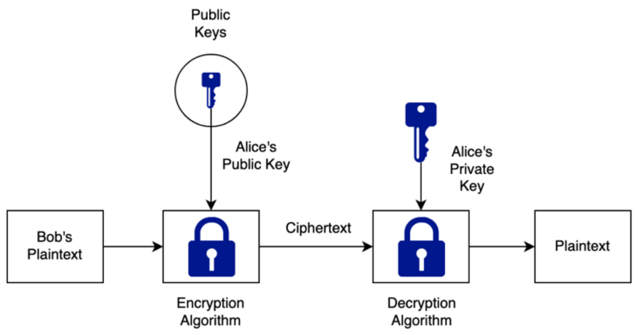

Figure 2: Asymmetric Encryption

For this cryptosystem to be secure, there are many protocols to follow. The most important being that one key should be deemed a private key, and this key should never be shared amongst other parties. In this system, there is no need to share this key. Also, it must be impossible to decrypt a ciphertext if the private key is kept private. Compounding this assurance, an attacker should not be able to break the system by determining a comparison key by having the algorithm, one of the keys, and different ciphertexts returned from the encryption algorithm.

Asymmetric encryption was established to break the constraints of the typical symmetric scheme key sharing. Unlike symmetric cryptosystems, if an attacker gains one of the key pairs, they will be unable to decrypt the message.

2.3 Identity-based encryption

2.3.1 First Proposal

Identity-based encryption (IBE) was first proposed by Shamir [1], in which he proposed an encryption scheme based off a public key system which allowed for a sender to encrypt a message and subsequently decrypt a message independent of the two operations. This negates the need for the random generation of public/private key pairs by allowing the user to create their public key based off identifying strings, such as email and network address.

In this system, the identifiers must uniquely map to one user. Once the public key is assigned, a secret key is then generated regarding this key.

Since a user's encryption key is their identity, *i*, and the decryption key is generated from *i* and a random seed k, there is no need for another channel between the two entities. See figure 3 for a visual representation of this. This type of encryption can be applied to many network systems, namely one in which they want information to be readily available to certain sets of users without the need to distribute multiple pairs of public and private keys.

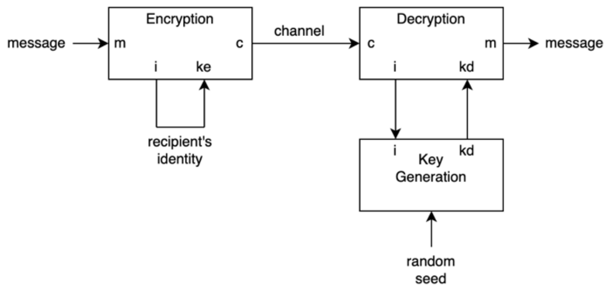

Figure 3: Recreation of Shamir's Proposal

2.3.2 Fuzzy IBE

Furthering research off the original proposal of IBE, Sahai and Waters [2] proposed a deeper form they called fuzzy identity-based encryption. Instead of the traditional form of viewing a user's identity from the form of strings, they proposed a scheme which allowed for viewing identity from a set of descriptive attributes.

As described in their research, inside a F-IBE system a user can decrypt a given ciphertext c, with the secret key for the identity w, that was encrypted with the public key w*'*, only if w and w*'* are within a certain distance specified by the administrator of the system. This provides the system with a very useful tool, error tolerance. Error tolerance pertains to the usefulness of a function when errors are present; with noisy data, an amount of error tolerance is a necessity.

There are more possibilities that open when using a fuzzy IBE system, such as the use of biometric identities and the proposal of attribute-based encryption. The use of biometric identities in encryption schemes, such are fingerprint or face scans, are immensely beneficial to the field of security, but we are more concerned with the possibility of attribute-based encryption.

2.4 Mathematical introductions

The following definitions are from the authors Waters [3] and Bemil [7].

2.4.1 Bilinear Mapping

The following assumptions on groups with computable bilinear maps were presented by Waters [3]:
We establish the following variables G and G_T to be two multiplicative cyclic groups of the prime order p. We assign g to be a generator of G, and e to be a bilinear map indicating that:

$$e = G * G \rightarrow G_T \qquad (1)$$

For e to be mathematically categorized as a bilinear map, it must bilinear and non-degenerative. In order to be bilinear, let all $x, y \in G$ and $a, b \in Z_p$ then we have $e(x^a, y^b) = e(x, y)^{ab}$. For e to be non-degenerative, let $e(g, g) \neq 1$.

2.4.2 Access Structures

The definition of an access structure, Bemil [7], is as follows: Let $\{P_1, \ldots, P_n\}$ be the set of parties. A collection $A \subseteq 2^{\{P_1, \ldots, P_n\}}$ is monotone if B ∈ A and B ⊆ C implies C ∈ A. An access structure can be defined as a monotone collection A of non-empty subsets of $\{P_1, \ldots, P_n\}$. The sets in A are called authorized sets and the sets outside of A are called unauthorized sets.

2.4.3 Linear Secret Sharing Schemes

The definition of Linear secret sharing schemes was given by Bemil [7]. A secret sharing scheme is linear if: The piece of each party is a vector over K, as well as during the generation of the pieces, the dealer chooses independent random variables, denoted $r_1, \ldots, r_L$, each one distributed uniformly over K. Each coordinate of the piece of every party is a linear combination of $r_1, \ldots, r_L$ and the secret s.

3 Attribute-based encryption

3.1 ABE Functions

Every attribute-based encryption system is comprised of a variation of the following algorithms: Setup, Key Generation, Encrypt, and Decrypt. There are many different implementations of these algorithms, but the basic purpose is as follows:

Setup (): The setup function is ran to create an instance of an ABE system. The parameter k is an origin value, which is used to create and return a Master Key, *MK*, and a set of public key attributes, *PK*.

Key Generation (): Once ABE system has been established, a set of attributes S for a particular user and the master key, *MK*, produced from the Setup (). The secret key, *SK*, is then produced.

Encrypt (): A message M is passed, along with the set S', derived from the set of attributes S and a public key *PK*. Returned from this is a generated ciphertext C.

Decrypt (): A ciphertext C is passed with the target S' and following set of attributes, S, for the target, and the secret key SK. To achieve decryption, the result from$| S' \cap S | >= k$ must be true, otherwise the decryption operation will fail.

3.2 Key-policy ABE

In this policy, the access policy is mapped onto the user's secret key. Following this, the ciphertext to be decrypted is computed in relation to the set of attributes.

3.3 Ciphertext-policy ABE

In this policy, the private key is related to the set of attributes. Alongside this, the ciphertext is mapped onto the access policy.

3 Constructions

3.1 Fuzzy IBE

Before diving into the critical aspects of this construction, a brief overview is stated. The construction of fuzzy identity-based encryption, as presented by Sahai and Waters [2], sets what they call 'identities' as a uniform set of attributes. A random polynomial of specified error tolerant degree, d - 1, is associated with the creation of each private key; the polynomials will be related at point zero, keeping some slight uniformity and relation across user's polynomials.

Once the set of possible attributes is defined, each user has their identities set. With this information, a private key component is created in relation to the user's polynomial. When considering decryption, the user's private key must be within d degrees to perform the decryption operation. The initial setup function is called, which defines universe of size U, which will consist of the first occurrences of integers in the field Z_p*. The cryptosystem's master key will be returned, after choosing randomly from the universe U.

To generate a subsequent private key for a chosen identity, w, inside the universe U, a random polynomial of degree $d - 1$ is chosen, continuing to hold the principle of relation upon point zero.

For encryption of a message, M, with a user's public key w', the value s in the field Z_p is chosen at random. They give the following function that defines the ciphertext:

$$E = \left(w', E' = MY^s, \{E_i = T_i^s\}_{i \in w'}\right) \qquad (2)$$

For decryption of the ciphertext E, that was previously encrypted with the public key w', there must be a private key in relation, w, such that the product of union of w and w' is greater than the specified degree of error tolerance d. If and only if this is satisfied, the ciphertext can be decrypted. Mathematically, they choose an arbitrary element that is in the product of $w \cap w'$, S, in the following equation to decrypt:

$$\frac{E'}{\prod_{i\in S}(e(D_i,E_i))^{\Delta_{i,S}(0)}} \quad (3)$$

3.2 Goyal [2006]

The findings of Goyal [6] can be seen as a furthering of Sahai and Water's[2] previous work. In their previous system, they found it to be unscalable and that there was a definite lack in expressiveness.

In their new system, they focus on labeling the ciphertext when encrypting with the according set of attributes. As for the private key system, each private key is related to an access structure that determines what it should be allowed to decrypt. The access tree for which the key is mapped upon holds the attributes as leaf nodes. Instead of a ciphertext-policy attribute-based system, this would be referred to as a key-policy attribute-based encryption system.

As well as the previous functions stated in the construction of fuzzy identity-based encryption, a function they call Delegation is also used. This allows for a more expressive system which allows a key for the access structure *X*, to create a key for access structure *Y* if *Y* is more expressive than *X*.

Before describing the functions inside this construction, the construction of the underlying access trees must be explained. An access tree is a tree that represents an access structure; each non-leaf node is what they describe as a threshold gate and each leaf node consists of an attribute. The threshold gates could consist of an AND/OR gate signifying the necessity level for each attribute in the leaf nodes.

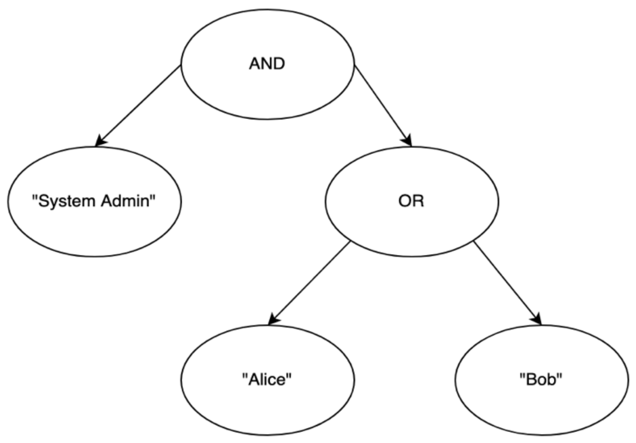

Figure 4: Access Tree Structure

Beginning the functions, the initial setup function defines the universe *U*, which consists of the attributes *i*. Choosing from the field Z_p, assign a random integer for each attribute in *U*. The master key for the system is returned.

To encrypt a message *M*, you must also pass along a set of attributes and a public key. They define the following encrypting equation as:

$$E = (\gamma, E' = MY^s, \{E_i = T_i^s\}_{i\in\gamma}) \quad (4)$$

As for key generation, the access tree, T, for which the key is built upon, and the master key MK are needed. Starting in a top-down fashion, assign a polynomial for each node in the tree T. Once the polynomial has been established, set the degree in the usual way of the threshold value minus 1. After the polynomials are defined, the returned value can be described as the following equation:

$$D_x = g^{\frac{q_x(0)}{t_i}} \quad (5)$$

To decrypt in this system, it takes the encrypted message E as input, the private key D for that user, and a node x to be decrypted. In the simplest case when x is a leaf node, it will decrypt if the following function is satisfied:

$$e(D_x, E_i) = e\left(g^{\frac{q_x(0)}{t_i}}, g^{s*t_i}\right) = e(g,g)^{s*q_x(0)}\ if\ i\ \in \gamma$$
(6)

In the case when x is a non-leaf node, the algorithm must recurse through all of the child nodes of the non-leaf node x. It will then store the value from the attempted decryption of each node into a set Fz . If the set is satisfactory, it will then be thrown into the following equation:

$$F_x = \prod_{z\in S_x} F_z^{\Delta_{i,S'_x}(0)} \quad (7)$$

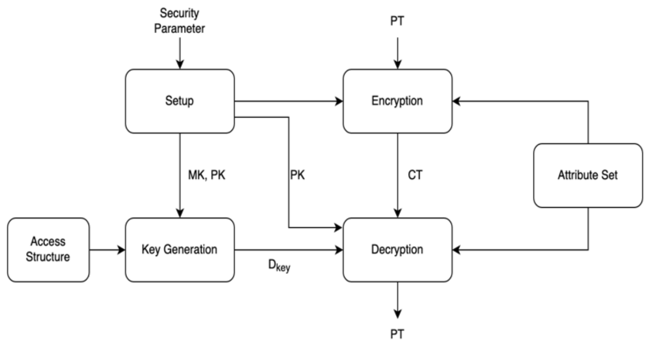

Figure 5: Goyal's System

3.3 Bethencourt

In this work, Bethencourt et al [4], further the research of Goyal[6] and Waters[2] into a different direction. Instead of key-policy attribute-based encryption, they present a ciphertext-policy attribute-based encryption scheme. In this system, the key is associated with the defining attribute set, rather than the ciphertext. Also, the ciphertext is associated with the according policies.

In this construction, the setup function is not uniform to the preceding constructions. Although, the usual assumption of a bilinear group of specified prime order, along with assigning a generator *g*. They differ in the next step of this algorithm, in which they choose two exponents in the field Z_p, denoting as *a, b.* They provide equation (8) to describe the public key, as well as MK = (b, g^a).

$$PK = G_0, g,$$

$$h = g^b, f = g^{1/b}, \quad (8)$$
$$e(g,g)^a$$

As for encryption, a polynomial is chosen for each node in the access tree T for which the message M is mapped under. The process begins at the root node and moves down, assigning a polynomial of the threshold value minus 1.

Alongside this process, in the same top-down fashion as before, a random value inside the field Z_p is assigned as the product of the polynomial function at point zero. To complete the expression, it then picks a random value under each degree in the polynomial. The following equation is used to define this process:

$$CT = (T, \tilde{C} = M_e(g,g)^{as}, C = h^s, \forall_y \in Y: C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)})$$
(9)

For key generation, the set of attributes is passed which then returns a key that relates to the set. A random value in the field Zp is chosen for each attribute, which then generates the subsequent key as:

$$SK = (D = g^{\frac{a+r}{B}}, \forall_j \in S: D_j = g^r * H(j)^{r_j}, D'_j = g^{r_j})$$
(10)

This construction, as well as the previously explained construction, has a delegation function. This function takes the secret key, as well as a set that is inside the set of attributes S for the given secret key. This algorithm provides an additional secret key for the set which is equivalent to the secret key previously provided.

Lastly is the decryption function for this construction. This is a recursive function that attempts to decrypt all nodes x. A ciphertext, the following secret key mapped to the set of attributes, and a node x from the access tree T, is needed as parameters to this algorithm. In practice, this function will start with the root node and recurse down through the tree. If the tree T is in satisfactory relation to the set S, then it will decrypt and return the plaintext.

3.4 Waters

As per the preceding trend, the work of Waters [3] is compounding the previous works on attribute-based encryption. The main issue that was left following the work of Bethencourt [4] was the fact that it was only proven to be secure in simple generic group situations. While keeping the use of a ciphertext-policy system, Waters did change the method by removing the use of access trees and introducing the mapping of onto a Linear Secret Sharing Scheme matrix instead. They also prove construction under what they call the Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption, instead of a generic group. With this method, they were able to give an efficient and provably secure system, which answers the problems left open from the previous works. This construction consists of the following functions: Setup, Encrypt, Key Generation, and Decrypt.

In the setup function, the distinct number of attributes in the given system is needed. Once it has this information, it then defines a group of prime order *p,* and a generator *g*. Once these are defined, a set of random group elements of size *U* and two random exponents *a* and *b* from the field Z_p are defined. A public and master key is returned from this.

Following the setup is the key generation algorithm. With the master key and the set of attributes as parameters, a random element *t* in the set Z_p is chosen. With this data, the following equation is used to generate a private key:

$$K = g^a g^{bt},$$
$$L = g^t,$$
$$\forall x \in S \; K_x = h_x^t$$
(11)

For the encryption algorithm, the public key *PK*, the message *M* to encrypt, and the LSSS access matrix (*M*, *p*). The LSSS structure consists of an *l * n* matrix, for which the algorithm initially defines the vector $\vec{v} \in Z_p^n$. Starting at 1 and following until it reaches the defined value *l*, the algorithm will calculate $\lambda_i = \vec{v}$, for each row M_i , as well as defining the random values in the field Z_p from 1 to *l.*

In the decryption algorithm, the ciphertext *CT* given from the LSSS access structure (*M, p*) and a private key for the given set *S* is needed. If the provided set and access structure are satisfactory, the following algorithm will be used for decryption:

$$e(C', K)/(\prod_{i \in I}(e(C_i, L)\, e(D_i, K_{p(i)}))^{w_i} =$$
$$e(g,g)^{as} e(g,g)^{bst} / (\prod_{i \in I} e(g,g)^{tb\lambda_i w_i}) = e(g,g)^{as}$$
(12)

Where $I \subset \{1,2, \dots l\} = \{i: p(i) \in S\}$ and the set of constant values that are satisfactory shares of the secret *s* in collusion with *M.*

4 Common ABE Issues

The problem management in the following section took inspiration from Piretti[5] who extensively covered this topic.

4.1 Collusion

Collusion in this sense pertains to act of an attacker combining keys to gain access to data that should otherwise be unattainable. To explain further, let Alice have an abundance of keys for the attribute-based system. There should never be the case in which Alice uses any string of keys together to decrypt a message, holding that none of the keys are valid for decryption in the system. Going further, let Bob also be a user with a key for this system. Bob should not be able to group any set of keys with Alice to decrypt a message that neither of them have access to decrypt.

The most popular technique to provide security against collusion is to randomize the user's secret keys. In this way, the possibility of collusion is negligible.

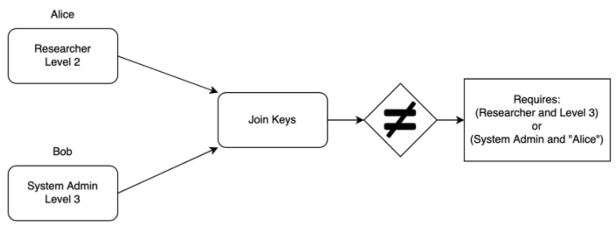
Alice
Researcher Level 2
Bob
System Admin Level 3
Join Keys
≠
Requires: (Researcher and Level 3) or (System Admin and "Alice")

Figure 6: Collusion Resistance

4.2 Key Revocation

The act of revocation is a present issue in both key management and user policy management. In symmetric encryption, this is an easy situation to solve. However, in an attribute-based system, multiple users will share the same attributes and therefore make it difficult to perform a key revocation.

One proposed idea is to create a time constraint on the key. In this example, a key will only be useful for the duration of the time for which it was defined. This solves the issue easily, as it requires for new updated keys in a time frame specified by the security administrator of the system. This does however become a tedious task for the system to maintain an intense number of attribute sets and keys, for which they must update again at the end of the quarter.

4.3 Modifying attributes in users

Given the simplicity of the key generation algorithms, modifying the attributes of a user are incredibly simple. Although, this does require some work from the key administrator.

The first topic of discussion would be to add attributes to a user. In an attribute-based system, all that would need to happen for the user to have additional attributes would be for the key distributor to create another key, but with the set of attributes that has the additional attribute.

The same functionality is used when describing the removal of attributes from a user in this system. All that is necessary to do is run the key generation algorithm, but this time the attribute set would need to be smaller, this way would allow for accurate removal.

4.4 Modifying attributes in policies

In contrast to attributes and users, to add or remove an attribute from a policy in an attribute-based system the encryption and decryption algorithms must be used. Formally, to add an attribute to a policy, you would need to decrypt this object and then encrypt the object with a different set of policies.

5 Conclusion

Attribute-based encryption is a new and efficient system of encrypting data. With this system, you are able to achieve newfound results in the field of encryption. Instead of using symmetric encryption, where you only have one key that is used encrypt and decrypt, you are able to provide a deeper sense of access control.

In this system, we allow a set of attributes, or roles that can decrypt data based on the role. This allows for multiple users to be able to decrypt the same data, with the caveat that they are of the desired access level to obtain the information.

In this paper, there was an overview of the necessary information to broach this type of encryption. It was found pertinent to include the more basic form of encryption, symmetric, and then dive deeper into the foundation of asymmetric encryption.

After these topics, the first happenings of identity-based encryption were displayed by Shamir[1], then when the first actual implementation of a basic form of attribute based encryption by Sahai[2]. This gave way to a multitude of ground breaking research into this field. Following Sahai[2], in chronological order the following constructions of Goyal[6], Bethencourt[4], and Waters[3] were presented.

The material was shown in this format to display the evolution of this type of system; starting with an incredibly basic form in Sahai[2], speaking on the first rendition of key-policy ABE in Goyal[6], then formulation and improving the form of ciphertext-policy ABE in Bethencourt[4] and Waters[3].

6 References

[1] Adi Shamir. Identity-based cryptosystems and signature schemes. Advances in Cryptology, 47-53

[2] Amit Sahai and Brent Waters. 2005. Fuzzy identity-based encryption. Lecture Notes in Computer Science, 457-473.

[3] Brent Waters. 2011. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Public Key Cryptography – PKC 2011 (2011), 53–70.

[4] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy attribute-based encryption. 2007 IEEE Symposium on Security and Privacy (SP '07) (2007).

[5] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. 2010. Secure attribute-based systems. Journal of Computer Security 18, 5 (2010), 799–837

[6] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 2006. Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security - CCS '06 (2006).

[7] Dan Boneh, et al. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys

[8] William Stallings. 2017. Cryptography and Network Security: Principles and Practice. Pearson Education, Inc.