I. PROPOSED ECC-CP-ABE SCHEME

We propose a novel ECC-based CP-ABE scheme known as Ecc-bAsed ScalablE Revocation (EASER) CP-ABE. The scheme comprises of a trusted server which performs the setup and key distribution. It also used a trusted proxy server that maintains a Revocation List of revoked users and generates proxy components to assist in partial decryption.

- **Trusted server** The trusted server of the system performs the setup, key generation and encryption of the data. It retains details for registered users, such as user identification, name and also issues the user's secret keys.
- **Proxy Server** It maintains a revocation list of revoked user identities. It also maintains partial secret keys of all users as discussed in the keygen step. It sends proxy component Q to the user to assist in partial decryption, such that if a user is present in the revocation list, the decryption fails for only the revoked users. The other valid users can decrypt uninterruptedly without any requirement of re-encryption or re-distribution of keys.

A. *Phases of the EASER CPABE scheme:*

The scheme comprises the following phases: Setup, Encryption, KeyReg and Decryption.

B. *Setup phase*

The setup phase is the initialisation phase. The size n of the Universe of attributes U identifies how many different attributes are present in the user secret keys. A bit 0 or 1 denotes an attribute is present or absent. For $n = 4$, there are 4 attributes in a users key and system policy that encrypts the data. A sample attribute definition can be: {*IT Professional?*, *joined before march 2016?*, *In an ongoing project?*, *Access to top secret files?*}. A sample attribute set can be $\mathbb{A} = \{0010\}$ In the same way a sample system policy for encryption can be $\mathbb{P} = \{0010\}$. In this case, since the attributes associated with user's secret key satisfy the system's access policy, the user can successfully decrypt the ciphertext using the secret keys denoted by attribute set $\mathbb{A}$. If the Policy is $\mathbb{P} = \{0010\}$, then the user will not be able to decrypt, since the attribute set A does not satisfy the policy $\mathbb{P}$.

- **SP1:** Choose the parameters for an Elliptic Curve group $\mathbb{G} = \{p, E_{\mathrm{p}}(a, b), P\}$, where P is the base point on the curve and p is some large prime number which defines the field Z_{p} of the curve. The order of the curve generated must be prime in order to allow ECC point division. It is best to follow the standard NIST ECC curves which provide the best security and also adhere to all requirements of the EASER scheme.
- **SP2** : Within the finite field of p generate three random numbers α, k_1 and k_2 such that $\{\alpha, k_1, k_2\} \neq 0$. ~~Calculate the values of P_{i}, U_{i} and~~ V_{i} $\forall i \in 0, 1 \ldots n$ as follows:

$$P_{\mathrm{i}} = \alpha_{\mathrm{i}} P \tag{1}$$
$$U_{\mathrm{i}} = k_1 \alpha^{\mathrm{i}} P \tag{2}$$
$$V_{\mathrm{i}} = k_2 \alpha^{\mathrm{i}} P \tag{3}$$

- **SP3** : Choose any 4 trapdoor hash functions H_1, H_2, H_3 and H_4 defined as follows :

$$H_1, H_4 \quad : \{0,1\}^* \longrightarrow {Z_{\mathrm{p}}}^* \tag{4}$$
$$H_2 \quad : \{0,1\}^* \longrightarrow \{0,1\}^{|\sigma|} \tag{5}$$
$$H_3 \quad : \{0,1\}^* \longrightarrow \{0,1\}^{|\mathrm{m}|} \tag{6}$$

where σ is some large string generated randomly, M is the message to be encrypted ~~and~~ $|\ .\ |$ is the operator to find the length of a string ~~within it and~~ $\{0,1\}^{\mathrm{p}}$ denotes a random stream of binary digits of length p and $*$ denotes any number from $0 \longrightarrow \infty$.

- **SP4:** Using the above values~~m~~ it generates the the global secret key *GSK* and global public key *GPK* $\forall i \in 0, 1, \ldots, n$ as follows:

$$GSK = \{\alpha, k_1, k_2\} \tag{7}$$
$$GPK = \{\mathbb{G}, P_{\mathrm{i}}, V_{\mathrm{i}}, U_{\mathrm{i}}, H_1, H_2, H_3, H_4\} \tag{8}$$

To remove these: k_1

C. *KeyReg Phase*

This phase assigns User-ID(*UID*$_j$) to user j. It takes as input the *GSK*, the *GPK* and the ~~Employee ID(*EID*$_j$)~~ or credentials of a user to generate a user secret key k_{u}.

- **KP1:** Let $\mathbb{A} = \{a_1, a_2, \ldots, a_{\mathrm{n}}\}$ be the attribute set for the user. Compute the value of the expression:

$$f(\alpha, \mathbb{A}) = \prod_{i=1}^{n} (\alpha + H_4(i))^{(1-i)} \tag{9}$$

The polynomial function $f(x, \mathbb{A})$ is of degree at-most n.

- **KP2** : Within the finite field Z_{p}, generate two random numbers r_{u} and t_{u}, and compute the following values:

$$u_1 = r_u + k_1.f(x, \mathbb{A}) \tag{10}$$
$$u_2 = s_u - k_2.f(x, \mathbb{A}) \tag{11}$$
$$u_3 = r_u k_2 + t_u k_1 \tag{12}$$

~~Send the portion of the keys u_1 and u_2 to the user and u_3 to the proxy server.~~

- **KP3** : Sends the portion of the secret key $u_3.f(x, \mathbb{A})$ to the proxy server. It provides the following identity and portion of the secret keys to the user *j*:

$$k_u = \{\sout{uid_j}, u_1, u_2\} \tag{13}$$

D. *Encryption Phase:*

It takes the plaintext, access policy and GPK as input. Standard AES algorithm encrypt the plaintext and the EASER CPABE scheme encrypts the AES key.

- **EP1:** Generate ~~a~~ random number σ. ~~Now generate~~ another random number $\sigma_{\mathrm{m}} \in \{0,1\}^{|\sigma|}$.
  What is the difference between the two?
- **EP2:** Compute the values of r_{m} and k_{m} according to the expression:

$$r_m = H_1(\mathbb{P}, M, \sigma_m) \tag{14}$$
$$k_m = KDF(r_m \mathbb{P}) \tag{15}$$

- **EP3:** For the access policy $\mathbb{P} = \{b_1, b_2, \ldots, b_n\}$, compute the following:

$$f(x, \mathbb{P}) = \prod_{i=1}^{n} (x + H_4(i))^{(1-b_i)} \quad (16)$$

The polynomial function $f(x, \mathbb{P})$ is of degree at-most n. Let f_i denote the coefficient of x_i in the polynomial function $f(x, \mathbb{P})$.

- **EP4:** Compute the following:

$$P_{m,i} = r_m P_i, i = 1, \ldots, n - |\mathbb{P}| \quad (17)$$

$$K_{1,m} = r_m \sum_{i=0}^{n} f_i U_i \quad (18)$$

$$K_{2,m} = r_m \sum_{i=0}^{n} f_i V_i \quad (19)$$

$$C_{\sigma_m} = H_2(k_m) \oplus \sigma_m \quad (20)$$

$$C_m = H_3(\sigma_m) \oplus M \quad (21)$$

Do not put like this expand equations and put properly in the above:

It can be conveniently proved that K_{1m} and K_{2m} results into $r_m k_1$ $f(\alpha,\mathbb{P})$P and $r_m k_2$ $f(\alpha,\mathbb{P})$P respectively, by putting the values of U_i and V_i in the respective expressions.

E. Decryption Phase:

The decryption phase takes as input Ciphertext along with user's secret key k_u = {~~uid$_j$,~~ u_1, u_2} and generates the plaintext as ~~as~~ output.

Change the following to itemize and equations

DP1: If access policy P is not a subset of attribute set A, then abort.

DP2 : Compute the values U and V as follows:

$U = u_2 k_{1m} = (t_u\text{-}k_2\ f(\alpha, \mathbb{A}))(r_m k_1\ f(\alpha, \mathbb{P})P)$

$V = u_1 k_{2m} = (r_u\text{-}k_1\ f(\alpha, \mathbb{A}))(r_m k_2\ f(\alpha, \mathbb{P})P)$

Send U, V and user id uid_j to the proxy server. The proxy server sends proxy component Q back to the user to assist partial decryption and scalable revocation.

Proxy Server

It maintains a revocation list ~~which comprises of user ids that must revoked users~~. The proxy server generates proxy components and sends them to all users. For revoked users, it modifies the proxy components so that decryption fails. The other valid users get proper proxy components so that description is successful and they access the ciphertext without an interruption.

A user send uid_j, U and V to the proxy server. The proxy server also maintains a list of user ids as well as the secret key components (u_3, $f(\alpha,\mathbb{A})$) as discussed in the KeyGen phase. The proxy server does the following to generate the proxy components:

DP2.1 : R = U+V

Therefore,

$R = U+V = (t_u - k_2\ f(\alpha, \mathbb{A}))(r_m k_1\ f(\alpha, \mathbb{P})P) + (r_u - k_1\ f(\alpha, \mathbb{A}))(r_m k_2\ f(\alpha, \mathbb{P})P)$

$= t_u r_m k_1\ f(\alpha, \mathbb{P})$ - $r_m k_1 k_2\ f(\alpha, \mathbb{P})\ f(\alpha, \mathbb{A})P$

$+r_u r_m k_2\ f(\alpha, \mathbb{P})$ - $r_m k_1 k_2\ f(\alpha, \mathbb{P})\ f(\alpha, \mathbb{A})P$

$= r_m(r_u k_2 + t_u k_1)\ f(\alpha, \mathbb{P})P$

$= r_m u_3\ f(\alpha, \mathbb{P})P$

DP2.2 : The proxy server checks if the user id uid_3 is registered and also present in the revocation list.

Case No Revocation : For a valid user ~~id~~ calculate proxy component Q as:

$Q = \frac{R}{X_i}$

where X_i ~~is the value mapped to UID in the memory.~~

~~Since,~~ ~~X_i~~= $u_3\ f(\alpha,\mathbb{A})$

$Q = \frac{R}{X_i} = \frac{U+V}{f(\alpha,\mathbb{A})} = \frac{r_m u_3\ f(\alpha,\mathbb{P})P}{u_3\ f(\alpha,\mathbb{A})}$

$= r_m\ F(\alpha,\mathbb{P})$P

where $F(\alpha) = \frac{f(\alpha,\mathbb{P})}{f(\alpha,\mathbb{A})}$

The proxy server returns the proxy component Q to the valid user.

Case Revocation: If user id uid_j is present in the revocation list, then compute the proxy component Q as follows:

$Q = \frac{R}{B}$, where B is some random number such that Q is not same as $r_m\ F(\alpha,\mathbb{P})$P. Return Q to the user so that decryption will fail.

DP3: Evaluate the expression for $F(x)$ where $F(x) = F(x,\mathbb{A},\mathbb{P})$ is defined as

$$F(x,\mathbb{A},\mathbb{P}) = \sum_{i=1}^{n-|\mathbb{P}|} (x+H_4(i))^{c_i}$$

where $c_i=a_i$ - b_i

Let F_i be the coefficient of x_i in $F(x)$. Since $\mathbb{P} \subseteq \mathbb{A}$, $F_0 \geq 1$.

DP4: Calculate the value of W using the expression

$W = \sum_{i=1}^{n-|\mathbb{P}|} F_i P_{m,i}$

$= r_m(\sum_{i=1}^{n-|\mathbb{P}|}(F_i \alpha^i)P$

$= r_m(\sum_{i=1}^{n-|\mathbb{P}|}(F_i \alpha^i + F_0 - F_0)P$

$= r_m((F)_{(\alpha)} - F_0)P$

$= r_m F_\alpha P - r_m F_0 P$

Compute $\frac{1}{F_0}(Q-W)$.
For a valid user:

$\frac{1}{F_0}(Q - W) = \frac{1}{F_0}(r_m F \alpha P - (r_m F(\alpha) - r_m F_0))P$

$= \frac{1}{F_0}(r_m F_0 P)$

$= r_m P$

For a revoked user the above computation fails :
$\frac{1}{F_0}(Q - W) != r_m P$

DP5: Compute :

$\sigma'_m = H_2(KDF(r_m P)) \oplus C_{\sigma_m}$

$M' = C_m \oplus H_3(\sigma'_m)$

$r'_m = H_1(\mathbb{P}, M', \sigma'_m)$

Verify if $r_m P = r_m P$ in phase DP4, then the message M is valid.

F. Fix for Attack

Theorem 1: Proposed EASER CP-ABE scheme is secure and does not allows deriving of system private keys (k_1, k_2) by collusion attack. referemce

Proof: $u_1^i = r_{u_i} + k_1 \ f(\alpha, \mathbb{A})$
$u_2^i = t_{u_i} + k_1 \ f(\alpha, \mathbb{A})$ for i=1,.....l
Here the system is of '2l' linear equations and 2l+2 unknowns. Hence not solvable.