Practical and Secure Policy-based Chameleon Hash for Redactable Blockchains

1 Security Analysis of the Composite Assumption

Theorem 1. *Let* $(\epsilon_1, \epsilon_2, \epsilon_T) : \mathbb{Z}_q \to \{0,1\}^*$ *be three random encodings (injective functions) where* $\mathbb{Z}_q$ *is a prime field, and the encoding of group elements are* $\mathbb{G} = \{\epsilon_1(a) | a \in \mathbb{Z}_q\}, \mathbb{H} = \{\epsilon_2(b) | b \in \mathbb{Z}_q\}, \mathbb{G}_T = \{\epsilon_T(c) | c \in \mathbb{Z}_q\}$. *If* $(a, b, c) \xleftarrow{R} \mathbb{Z}_q$ *and encodings* $\epsilon_1, \epsilon_2, \epsilon_T$ *are randomly chosen, we define the advantage of the adversary in solving the composite assumption with at most* $\mathcal{Q}$ *queries to the group operation oracles* $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ *and the bilinear pairing* $\hat{\mathsf{e}}$ *as*

$$
\begin{aligned}
|\mathtt{Adv}_{\mathcal{A}}^{Com}(\lambda) &= \Pr[\mathcal{A}(q, \epsilon_1(1), \epsilon_1(a_1), \epsilon_1(a_2), \epsilon_1(a_1 \cdot \alpha_1^{\ell+1}), \epsilon_1(a_2 \cdot \alpha_2^{\ell+1}), \\
&\quad \{\epsilon_1(\alpha_1), \cdots, \epsilon_1(\alpha_1^{\ell})\}, \{\epsilon_1(\alpha_2), \cdots, \epsilon_1(\alpha_2^{\ell})\}, \epsilon_1(s), \epsilon_1(t_0), \epsilon_1(t_1), \epsilon_2(1), \epsilon_T(1)) \\
&= w : (a_1, a_2, \alpha_1, \alpha_2, s \xleftarrow{R} \mathbb{Z}_q, w \in (0,1), t_w = (\alpha_1^{\ell} + \alpha_2^{\ell}), t_{1-w} = s)] - 1/2| \\
&\leq \frac{2(\ell+2)(2\ell+9)^2}{q}
\end{aligned}
$$

Proof. Let $\mathcal{S}$ play the following game for $\mathcal{A}$. $\mathcal{S}$ maintains three polynomial-sized dynamic lists: $L_1 = \{(p_i, \epsilon_{1,i})\}, L_2 = \{(q_i, \epsilon_{2,i})\}, L_T = \{(t_i, \epsilon_{T,i})\}$. The $p_i \in \mathbb{Z}_q[A_1, A_2, AL_1, AL_2, S, T_0, T_1]$ are 7-variate polynomials over $\mathbb{Z}_q$, such that $p_0 = 1, p_1 = A_1, p_2 = A_2, p_3 = A_1 \cdot AL_1^{\ell+1}, p_4 = A_2 \cdot AL_2^{\ell+1}, \{p_u = AL_1^u\}_{u=1}^{\ell}, \{p_v = AL_2^v\}_{v=1}^{\ell}, p_{2\ell+5} = T_0, p_{2\ell+6} = T_1$. $\mathcal{S}$ also generates $q_0 = 1, t_0 = 1$. Then, $\mathcal{S}$ sets three lists as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^{2\ell+6}, L_2 = (q_0, \epsilon_{2,0}), L_T = (t_0, \epsilon_{T,0})$, where $(\{\epsilon_{1,i}\}_{i=0}^{2\ell+6} \in \{0,1\}^*, \{\epsilon_{2,0}\} \in \{0,1\}^*, \{\epsilon_{T,0}\} \in \{0,1\}^*)$ are arbitrary distinct strings.

At the beginning of the game, $\mathcal{S}$ sends the encoding strings $(\{\epsilon_{1,i}\}_{i=0,\cdots,2\ell+6}, \epsilon_{2,0}, \epsilon_{T,0})$ to $\mathcal{A}$. Next, $\mathcal{S}$ simulates the group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ and the bilinear pairing $\hat{\mathsf{e}}$ as follows. We assume that all requested operands are obtained from $\mathcal{S}$.

- $\mathcal{O}_1$: The group operation involves two operands $\epsilon_{1,i}, \epsilon_{1,j}$. Based on these operands, $\mathcal{S}$ searches the list L_1 for the corresponding polynomials p_i and p_j. Then, $\mathcal{S}$ perform the polynomial addition or subtraction $p_l = p_i \pm p_j$ depending on whether multiplication or division is requested. If p_l is in the list L_1, then $\mathcal{S}$ returns the corresponding ϵ_l to $\mathcal{A}$. Otherwise, $\mathcal{S}$ uniformly chooses $\epsilon_{1,l} \in \{0,1\}^*$, where $\epsilon_{1,l}$ is unique in the encoding string L_1, and appends the pair $(p_l, \epsilon_{1,l})$ into the list L_1. Finally, $\mathcal{S}$ returns $\epsilon_{1,l}$ to $\mathcal{A}$ as the answer. Group operation queries in $\mathbb{H}, \mathbb{G}_T$ (i.e., $\mathcal{O}_2, \mathcal{O}_T$) are treated similarly.
- $\hat{\mathsf{e}}$: The group operation involves two operands $\epsilon_{T,i}, \epsilon_{T,j}$. Based on these operands, $\mathcal{S}$ searches the list L_T for the corresponding polynomials t_i and t_j. Then, $\mathcal{S}$

perform the polynomial multiplication $t_l = t_i \cdot t_j$. If t_l is in the list L_T, then $\mathcal{S}$ returns the corresponding $\epsilon_{T,l}$ to $\mathcal{A}$. Otherwise, $\mathcal{S}$ uniformly chooses $\epsilon_{T,l} \in \{0,1\}^*$, where $\epsilon_{T,l}$ is unique in the encoding string L_T, and appends the pair $(t_l, \epsilon_{T,l})$ into the list L_T. Finally, $\mathcal{S}$ returns $\epsilon_{T,l}$ to $\mathcal{A}$ as the answer.

After querying at most $\mathcal{Q}$ times of corresponding oracles, $\mathcal{A}$ terminates and outputs a guess $b' = \{0,1\}$. At this point, $\mathcal{S}$ chooses random $a_1, a_2, \alpha_1, \alpha_2, s \in \mathbb{Z}_q$, generates $t_b = (\alpha_1^{\ell+1} + \alpha_2^{\ell+1})$ and $t_{1-b} = s$. $\mathcal{S}$ sets $A_1 = a_1, A_2 = a_2, AL_1 = \alpha_1, AL_2 = \alpha_2, S = s, T_0 = t_b, T_1 = t_{1-b}$. The simulation is perfect unless the `abort` event happens. Thus, we bound the probability of event `abort` by analyzing the following cases:

1. $p_i(a_1, a_2, \alpha_1, \alpha_2, s, t_0, t_1) = p_j(a_1, a_2, \alpha_1, \alpha_2, s, t_0, t_1)$: The polynomial $p_i \neq p_j$ due to the construction method of L_1, and $(p_i - p_j)(a_1, \cdots)$ is a non-zero polynomial of degree $[0, 1, \cdots, \ell+2]$ (ℓ+2 is produced by $A_1 \cdot AL_1^{\ell+1}$). The maximum degree of $A_1 \cdot AL_1^{\ell+1}(p_i - p_j)(a_1, \cdots)$ is ℓ+2. By using Lemma 1 in [7], we have $\Pr[(p_i - p_j)(a_1, \cdots) = 0] \leq \frac{\ell+2}{q}$ and thus $\Pr[p_i(a_1, \cdots) = p_j(a_1, \cdots)] \leq \frac{\ell+2}{q}$. Therefore, we have the `abort` probability is $\Pr[\mathsf{abort}_1] \leq \frac{\ell+2}{q}$.
2. $q_i(a_1, \cdots) = q_j(a_1, \cdots)$: The polynomial $q_i \neq q_j$ due to the construction method of L_2, and $(q_i - q_j)(a_1, \cdots)$ is a non-zero polynomial of degree 0. The maximum degree is "0" since the list L_2 contains a single string $\epsilon_{(2,0)}$ only (note that we do not include group elements in $\mathbb{H}$). Therefore, the `abort` probability is "0".
3. $t_i(a_1, \cdots) = t_j(a_1, \cdots)$: The polynomial $t_i \neq t_j$ due to the construction method of L_3, and $(t_i - t_j)(a_1, \cdots)$ is a non-zero polynomial of degree $[0, 1, \cdots, \ell+2]$. The maximum degree of $(A_1 \cdot AL_1^{\ell+1}(t_i - t_j)(a_1, \cdots)$ is also ℓ+2. Therefore, we have $\Pr[(t_i - t_j)(a_1, \cdots) = 0] \leq \frac{\ell+2}{q}$ and thus $\Pr[t_i(a_1, \cdots) = t_j(a_1, \cdots)] \leq \frac{\ell+2}{q}$.

By summing over all valid pairs (i, j) in each case (i.e., at most $2\binom{2\ell+9}{2}$ pairs), we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \leq 2\binom{2\ell+9}{2} \cdot \left(\frac{\ell+2}{q} + \frac{\ell+2}{q}\right) \\ &\leq \frac{2(\ell+2)(2\ell+9)^2}{q}.\end{aligned}$$

2 Chameleon Hash Function

A chameleon hash function consists of the following algorithms [6].

– KeyGen: It takes a security parameter λ as input, outputs a chameleon key pair $(\mathtt{sk}, \mathtt{pk})$.

– Hash: It takes the chameleon public key $\mathtt{pk}$, and a message $m \in \mathcal{M}$ as input, outputs a chameleon hash h, randomness r. Note that $\mathcal{M} = \{0,1\}^*$ denotes a general message space.
– Verify: It takes the chameleon public key $\mathtt{pk}$, a message m, a chameleon hash h and randomness r as input, outputs a bit $b \in \{0,1\}$.
– Adapt: It takes the chameleon secret key $\mathtt{sk}$, messages m, m', chameleon hash h and randomness r as input, outputs a new randomness r'.

Chameleon hash function should ensure indistinguishability and collision-resistance. Indistinguishability means that the randomness associated with a chameleon hash does not reveal whether it was obtained from directly hashing or computed using the chameleon secret key. The collision-resistance means that adversaries cannot find two pairs (m, r) and (m', r') that map to the same chameleon hash h. Since the definition of indistinguishability and collision-resistance are various in the literature [3–6], we focus on a standard one [4] to prove the security of the proposed FB-PCH.

We use a modified version of discrete logarithm (DL)-based CH to construct our FB-PCH. The intention is to prevent key exposure attacks [2]: anyone can extract the trapdoor given two pairs (m, r) and (m', r') that map to the same chameleon hash h. Specifically, CH is associated with a public key $\mathtt{pk} = g^{\mathtt{sk}}$. Hashing a message m outputs $h = \mathtt{pk}^m g^r$ and $\xi = (R = g^r, \Pi)$, where Π is a non-interactive zero-knowledge (NIZK) for the value $\log(R)$. In Verify, the hash value h can be verified by checking if $h = \mathtt{pk}^m R$ and if Π is valid. In Adapt, $r' = (m - m') \cdot \mathtt{sk} + r$ and $\xi' = (R' = g^{r'}, \Pi')$ are computed, where Π' is a NIZK for $\log(R')$. The modified CH scheme achieves indistinguishability and collision-resistance. The construction is a combination of a classic DL-based CH scheme and a NIZK proof. The NIZK proof is used to prove the knowledge of randomness, and we rely on its simulation-sound extractability property to prove collision-resistance.

Theorem 2. *The modified DL-based CH scheme is collision-resistant if the classic DL-based CH scheme is collision-resistant, and the NIZK proof is simulation extractable.*

Proof. Let $\mathcal{S}$ denote an attacker against the classic DL-based CH scheme, who is given a chameleon public key $\mathtt{pk}^*$, a hash oracle and an adapt oracle, aims to find a collision which was not simulated by the adapt oracle. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

– $\mathcal{S}$ sets the chameleon public key as $\mathtt{pk}^*$.
– For a hash query on a message m, $\mathcal{S}$ forwards m to his hash oracle and receives a hash value h and a randomness r, where $h = (\mathtt{pk}^*)^m \cdot g^r$. Then, $\mathcal{S}$ returns (h, ξ) to $\mathcal{A}$, where a NIZK proof $\xi = (R = g^r, \Pi)$. For an adapt query, $\mathcal{S}$ obtains a new randomness r' from his adapt oracle and returns (m', h, ξ') to $\mathcal{A}$, where $\xi' = (R' = g^{r'}, \Pi')$.
– If $\mathcal{A}$ outputs a collision $(m^*, \xi^*, m'^*, \xi'^*, h^*)$, such that the chameleon hash is valid $h^* = (\mathtt{pk}^*)^{m^*} \cdot R^* = (\mathtt{pk}^*)^{m'^*} \cdot R'^*$, and the NIZK proofs (ξ^*, ξ'^*)

are valid, $\mathcal{S}$ extracts (r^*, r'^*) from (ξ^*, ξ'^*) due to NIZK's simulation-sound extractability. We require that either (h^*, m^*, ξ^*) or (h^*, m'^*, ξ'^*) must be a fresh collision, i.e., one that was never queried adapt oracle. Then, $\mathcal{S}$ outputs $(m^*, r^*, m'^*, r'^*, h^*)$ as a collision to the CH scheme; otherwise, $\mathcal{S}$ aborts the game.

Theorem 3. *The modified DL-based CH scheme is indistinguishable if the classic DL-based CH is indistinguishable.*

Proof. Let $\mathcal{S}$ denote an attacker against the DL-based CH, who is given a chameleon public key $\mathtt{pk}^*$ and a $\mathsf{HashOrAdapt}$ oracle, aims to break the classic CH's indistinguishability. $\mathcal{S}$ sets the chameleon public key as $\mathtt{pk}^*$.

If $\mathcal{A}$ submits two messages (m_0, m_1) to $\mathcal{S}$, $\mathcal{S}$ obtains a chameleon hash (h_w, r_w) from his $\mathsf{HashOrAdapt}$ oracle on messages (m_0, m_1), where $w \in [0, 1]$. Then, $\mathcal{S}$ generates a NIZK proof $\xi_w = (R_w = g^{r_w}, \Pi_w)$, and returns (m_w, h_w, ξ_w) to $\mathcal{A}$. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the classic CH's indistinguishability.

3 Correctness of the Decryption.

We provide the correctness of the decryption process. Suppose that the user holds a decryption key in the form of $\mathtt{sk}_{(\delta,t)} = (\mathtt{sk}_0, \{\mathtt{sk}_y\}_{y\in\delta}, \mathtt{sk}')$, where $\mathtt{sk}_0 = (h^{b_1\cdot r_1}, h^{b_2\cdot r_2}, h^r, h^{r+r'})$, and $\mathtt{sk}'_3 = g^{d_3} \cdot g^{-\sigma'} \cdot F(t)^{r+r_x+r'}$ (i.e., the user is not revoked at time t).

– The ciphertext associated with $y_1 \in \delta$ and $y_2 = \{1, 2, 3\}$ is calculated as

$$\begin{aligned} A = {} & \hat{\mathtt{e}}(\mathtt{H}(y_1 11)^{s_1\cdot\gamma_i} \cdot \mathtt{H}(y_1 12)^{s_2\cdot\gamma_i}, h^{b_1\cdot r_1}) \cdot \hat{\mathtt{e}}(\mathtt{H}(y_1 21)^{s_1\cdot\gamma_i} \cdot \mathtt{H}(y_1 22)^{s_2\cdot\gamma_i}, h^{b_2\cdot r_2}) \\ & \cdot\hat{\mathtt{e}}(\mathtt{H}(y_1 31)^{s_1\cdot\gamma_i} \cdot \mathtt{H}(y_1 32)^{s_2\cdot\gamma_i}, h^r) \cdot \underline{\hat{\mathtt{e}}(F(t)^s, h^{r+r'}) \cdot \hat{\mathtt{e}}(F(t), h^{r_x\cdot s})} \\ = {} & \hat{\mathtt{e}}(\mathtt{H}(y_1 11)^{s_1\cdot b_1\cdot r_1\cdot\gamma_i}, h) \cdot \hat{\mathtt{e}}(\mathtt{H}(y_1 12)^{s_2\cdot b_1\cdot r_1\cdot\gamma_i}, h) \cdot \hat{\mathtt{e}}(\mathtt{H}(y_1 21)^{s_1\cdot b_1\cdot r_2\cdot\gamma_i}, h) \\ & \cdot\hat{\mathtt{e}}(\mathtt{H}(y_1 22)^{s_2\cdot b_2\cdot r_2\cdot\gamma_i}, h) \cdot \hat{\mathtt{e}}(\mathtt{H}(y_1 31)^{s_1\cdot r\cdot\gamma_i}, h) \cdot \hat{\mathtt{e}}(\mathtt{H}(y_1 32)^{s_2\cdot r\cdot\gamma_i}, h) \cdot \underline{\hat{\mathtt{e}}(F(t)^s, h^{r_x+r+r'})} \end{aligned}$$

– The first pairing in B is calculated as

$$\begin{aligned} & \hat{\mathtt{e}}(\mathtt{H}(\pi(i)11)^{b_1\cdot r_1\cdot\gamma_i/a_1} \cdot \mathtt{H}(\pi(i)21)^{b_2\cdot r_2\cdot\gamma_i/a_1} \cdot \mathtt{H}(\pi(i)31)^{r\gamma_i/a_1} \cdot g^{\sigma_i\cdot\gamma_i/a_1} \cdot (g^{d_1})^{\mathbf{M}_{(i,1)}\cdot\gamma_i} \\ & \cdot \prod_{j=2}^{n_2} [\mathtt{H}(0j11)^{b_1\cdot r_1/a_1} \cdot \mathtt{H}(0j21)^{b_2\cdot r_2/a_1} \cdot \mathtt{H}(0j31)^{r/a_1} \cdot g^{\sigma'_j/a_1}]^{\mathbf{M}_{(i,j)}\cdot\gamma_i}, h^{a_1\cdot s_1}) \end{aligned}$$

– The second pairing in B is calculated as

$$\begin{aligned} & \hat{\mathtt{e}}(\mathtt{H}(\pi(i)12)^{b_1\cdot r_1\cdot\gamma_i/a_2} \cdot \mathtt{H}(\pi(i)22)^{b_2\cdot r_2\cdot\gamma_i/a_2} \cdot \mathtt{H}(\pi(i)32)^{r\cdot\gamma_i/a_2} \cdot g^{\sigma_i\cdot\gamma_i/a_2} \cdot (g^{d_2})^{\mathbf{M}_{(i,1)}\cdot\gamma_i} \\ & \cdot \prod_{j=2}^{n_2} [\mathtt{H}(0j12)^{b_1\cdot r_1/a_2} \cdot \mathtt{H}(0j22)^{b_2\cdot r_2/a_2} \cdot \mathtt{H}(0j32)^{r/a_2} \cdot g^{\sigma'_j/a_2}]^{\mathbf{M}_{(i,j)}\cdot\gamma_i}, h^{a_2\cdot s_2}) \end{aligned}$$

– The third pairing in B is calculated as

$$\hat{\mathsf{e}}(\underline{g^{d_3} \cdot g^{-\sigma'} \cdot F(t)^{r+r_x+r'}} \cdot \prod_{j=2}^{n_2} (g^{-\sigma'_j})^{\mathbf{M}_{(i,j)} \cdot \gamma_i}, h^s)$$

– By multiplying above three parings in B, we have

$$\begin{aligned} B = & \hat{\mathsf{e}}(\mathtt{H}(\pi(i)11)^{b_1 \cdot r_1 \cdot s_1 \cdot \gamma_i}, h) \cdot \hat{\mathsf{e}}(\mathtt{H}(\pi(i)21)^{b_2 \cdot r_2 \cdot s_1 \cdot \gamma_i}, h) \cdot \hat{\mathsf{e}}(\mathtt{H}(\pi(i)31)^{r \cdot s_1 \cdot \gamma_i}, h) \\ & \cdot \hat{\mathsf{e}}(\mathtt{H}(\pi(i)12)^{b_1 \cdot r_1 \cdot s_2 \cdot \gamma_i}, h) \cdot \hat{\mathsf{e}}(\mathtt{H}(\pi(i)22)^{b_2 \cdot r_2 \cdot s_2 \cdot \gamma_i}, h) \cdot \hat{\mathsf{e}}(\mathtt{H}(\pi(i)32)^{r \cdot s_2 \cdot \gamma_i}, h) \\ & \cdot \hat{\mathsf{e}}(g^{d_1 \cdot a_1 \cdot s_1}, h) \cdot \hat{\mathsf{e}}(g^{d_2 \cdot a_2 \cdot s_2}, h) \cdot \hat{\mathsf{e}}(g^{d_3 \cdot s}, h) \cdot \hat{\mathsf{e}}(F(t)^{r+r_x+r'}, h^s) \end{aligned}$$

where σ_i and σ'_j in B were cancelled out when multiplying, and recall that $\sum_{i \in I} \gamma_i \cdot \mathbf{M}_i = (1, 0, \cdots, 0)$.

– Eventually, B/A is calculated as below

$$\begin{aligned} B/A &= \hat{\mathsf{e}}(g^{d_1 \cdot a_1 \cdot s_1}, h) \cdot \hat{\mathsf{e}}(g^{d_2 \cdot a_2 \cdot s_2}, h) \cdot \hat{\mathsf{e}}(g^{d_3 \cdot (s_1+s_2)}, h) \\ &= \hat{\mathsf{e}}(g, h)^{s_1 \cdot (d_1 \cdot a_1 + d_3)} \cdot \hat{\mathsf{e}}(g, h)^{s_2 \cdot (d_2 \cdot a_2 + d_3)} \\ &= T_1^{s_1} \cdot T_2^{s_2}. \end{aligned}$$

4 Security Analysis of FB-ABE

Theorem 4. *The proposed FB-ABE scheme is semantically secure if the proposed composite assumption is held in the asymmetric pairing groups.*

Proof. We assume a simulator $\mathcal{S}$ whose goal is to the break the security of the composite assumption. $\mathcal{S}$ chooses a challenge time $t^* = t_1^* || t_2^* || \cdots || t_\ell^*$ and a challenge identity id^*.

– $\mathcal{S}$ simulates the public parameters as $g_c = \bar{g}^{\gamma} \cdot g^{\alpha_1^\ell}$, $\bar{g} = g^{z_c}$, $T_{1/2} = \hat{\mathsf{e}}(g, h)^{d_{1/2} \cdot a_{1/2}}$. $\hat{\mathsf{e}}(g_c, h^{\alpha_1})$, $g_0 = \bar{g}^{\delta} \cdot g^{\alpha_1^\ell \cdot t_1^*} \cdots g^{\alpha_1 \cdot t_\ell^*} \cdot g^{\alpha_1 \cdot \mathtt{H}_q(id^*)}$, $g_1 = \bar{g}^{\gamma_1} / g^{\alpha_1^\ell}, \cdots, g_\ell = \bar{g}^{\gamma_\ell} / g^{\alpha_1}$, where $d_{1/2}, \gamma, \gamma_1, \cdots \gamma_\ell, \delta, z_c \in \mathbb{Z}_q$ are chosen by $\mathcal{S}$. Since $\mathcal{S}$ can easily break the composite problem if $g^{\alpha_1^{\ell+1}}$ is unknown, we use it to simulate the updated decryption keys.
  $\mathcal{S}$ also generates other parameters honestly according to the scheme's description. By setting up the parameters as such, $\mathcal{S}$ implicitly sets $d_3 = \alpha_1$, and $g_c^{d_3} = \bar{g}^{\alpha_1 \cdot \gamma} g^{\alpha_1^{\ell+1}}$. Below, we mainly focus on the $g_c^{d_3}$-related simulations.
– $\mathcal{S}$ simulates the key update, break in, and decryption queries as follows.
  1. KeyUp. $\mathcal{S}$ just keep track of the present time t.
  2. We consider two cases in simulating users' decryption keys for decryption queries. The first case is $t \neq t^*$. We show $\mathcal{S}$ can simulate a decryption key at time $t = t_1 || \cdots || t_{\bar{k}} || \cdots || t_\ell$, where $\bar{k} \in [1, \ell]$. Note that $t_{\bar{k}} \neq t_{\bar{k}}^*$, which implies that $\bar{k}$ is not prefix of t^*, and $\bar{k}$ is the smallest index at time t.

In this case, $\mathcal{S}$ first chooses $z \in \mathbb{Z}_q$, and sets $r = \frac{\alpha_1^{\bar{k}}}{t_{\bar{k}} - t_{\bar{k}}^*} + z$. Then, $\mathcal{S}$ computes key components $(\mathtt{sk}_{(0,3)}, \mathtt{sk}_3')$ as

$$(h^r, \frac{g_c^{d_3}}{g_x} \cdot g^{-\sigma'} \cdot \underline{(g_0 \cdot g_1^{t_1} \cdots g_{\bar{k}}^{t_{\bar{k}}} \cdot g_\ell^{\mathtt{H}_q(id)})^r}, g_{\bar{k}+1}^r, \cdots, g_\ell^r) \tag{1}$$

This is a well-formed key at time $t = t_1 || \cdots || t_{\bar{k}}$, where g_x, σ' are chosen by $\mathcal{S}$. We show how to calculate the underlined (U) term in (1).

$$\begin{aligned} U &= [\bar{g}^\delta \cdot g^{\alpha_1^\ell \cdot t_1^*} \cdots g^{\alpha_1 \cdot t_\ell^*} \cdot g^{\alpha_1 \cdot \mathtt{H}_q(id^*)} (\bar{g}^{\gamma_1} / g^{\alpha_1^\ell})^{t_1} \cdots (\bar{g}^{\gamma_{\bar{k}}} / g^{\alpha_{\ell-\bar{k}+1}})^{t_{\bar{k}}} \cdot (\bar{g}^{\gamma_\ell} / g^{\alpha_1})^{\mathtt{H}_q(id)}]^r \\ &= [\bar{g}^{\delta + \Sigma_{i=1}^{\bar{k}} t_i \cdot \gamma_i + \gamma_\ell \cdot \mathtt{H}_q(id)} \cdot \prod_{i=1}^{\bar{k}-1} g_{\ell-i+1}^{t_i^* - t_i} \cdot g_{\ell-\bar{k}+1}^{t_{\bar{k}}^* - t_{\bar{k}}} \cdot \prod_{i=\bar{k}+1}^{\ell} g_{\ell-i+1}^{t_i^*} \cdot g^{\alpha_1 \cdot [\mathtt{H}_q(id^*) - \mathtt{H}_q(id))}]^r \\ &= Z \cdot g_{\ell-\bar{k}+1}^{r(t_{\bar{k}}^* - t_{\bar{k}})} \end{aligned}$$

where Z is shown as follows

$$Z = [\bar{g}^{\delta + \Sigma_{i=1}^{\bar{k}} t_i \cdot \gamma_i + \gamma_\ell \cdot \mathtt{H}_q(id)} \cdot \underline{\prod_{i=1}^{\bar{k}-1} g_{\ell-i+1}^{t_i^* - t_i}} \cdot \prod_{i=\bar{k}+1}^{\ell} g_{\ell-i+1}^{t_i^*} \cdot g^{\alpha_1 \cdot (\mathtt{H}_q(id^*) - \mathtt{H}_q(id))}]^r$$

$\mathcal{S}$ can compute all the terms in Z, and the underlined term in Z is equal to 1 because $t_i = t_i^*$ for all $i < \bar{k}$. The remaining term in $(g_0 \cdot g_1^{t_1} \cdots g_{\bar{k}}^{t_{\bar{k}}})^r$ is $g_{\ell-\bar{k}+1}^{r(t_{\bar{k}}^* - t_{\bar{k}})}$. Since $r = \frac{\alpha_1^{\bar{k}}}{t_{\bar{k}} - t_{\bar{k}}^*} + z$, we can rewrite it as follows

$$g_{\ell-\bar{k}+1}^{r \cdot (t_{\bar{k}}^* - t_{\bar{k}})} = g_{\ell-\bar{k}+1}^{z(t_{\bar{k}}^* - t_{\bar{k}})} \cdot g_{\ell-\bar{k}+1}^{(t_{\bar{k}}^* - t_{\bar{k}}) \frac{\alpha_1^{\bar{k}}}{t_{\bar{k}} - t_{\bar{k}}^*}} = \frac{g_{\ell-\bar{k}+1}^{z(t_{\bar{k}}^* - t_{\bar{k}})}}{g^{\alpha_1^{\ell+1}}}$$

Hence, the second component in (1) is equal to

$$\frac{g_c^{d_3}}{g_x} \cdot g^{-\sigma'} \cdot \underline{(g_0 \cdot g_1^{t_1} \cdots g_{\bar{k}}^{t_{\bar{k}}})^r} = g^{\alpha_1^{\ell+1}} \cdot \frac{\bar{g}^{\alpha_1 \cdot \gamma}}{g_x} \cdot Z \cdot \frac{g_{\ell-\bar{k}+1}^{z(t_{\bar{k}}^* - t_{\bar{k}})}}{g^{\alpha_1^{\ell+1}}} = \frac{\bar{g}^{\alpha_1 \cdot \gamma}}{g_x} \cdot Z \cdot g_{\ell-\bar{k}+1}^{z(t_{\bar{k}}^* - t_{\bar{k}})}$$

To this end, $\mathcal{S}$ can simulate the second component in (1) because the unknown value $g^{\alpha_1^{\ell+1}}$ is cancelled out. The first component h^r in (1), and other components $(g_{\bar{k}+1}^r, \cdots, g_\ell^r)$ can be easily computed by $\mathcal{S}$ since they do not involve $g^{\alpha_1^{\ell+1}}$. This completes the simulation of $g_c^{d_3}$-related key components at time $t \neq t^*$. $\mathcal{S}$ simulates other key components using the same approach described in FAME [1]. Specifically, $\mathcal{S}$ first obtains the composite assumption challenge $(h^{a_1 \cdot \alpha_1^{\ell+1}}, h^{a_2 \cdot \alpha_2^{\ell+1}}, h^{\alpha_1^{\ell+1} + \alpha_2^{\ell+1}})$. Then, $\mathcal{S}$ derives $(h^{a_1 \cdot \alpha_1^{\ell+1} \cdot \bar{r}}, h^{a_2 \cdot \alpha_2^{\ell+1} \cdot \bar{r}}, h^{(\alpha_1^{\ell+1} + \alpha_2^{\ell+1}) \cdot \bar{r}})$ to simulate other key components, where $\bar{r}$ is a randomly chosen value from $\mathbb{Z}_q$.

The second case is $t = t^*$, but $id \neq id^*$. If $\mathcal{A}$ issues a decryption key query on an attribute set δ (i.e., $1 \neq \Lambda^*(\delta)$, $\mathcal{S}$ simulates a decryption

key using the similar approach as above, but using the fact that $id \neq id^*$ instead of $t_{\bar{k}} \neq t^*_{\bar{k}}$. Recall that user's decryption key $\mathtt{sk}'_3$ involves $F(t, id) = g_0 \cdot \prod g_i^{t_i} \cdot g_\ell^{\mathtt{H}_q(id)} \in \mathbb{G}$. $\mathcal{S}$ sets $r = \frac{\alpha_1^\ell}{\mathtt{H}_q(id) - \mathtt{H}_q(id^*)} + z$ in equation (1) for $z \in \mathbb{Z}_q$, and the simulation follows the similar approach as above.

3. Break in. $\mathcal{S}$ simulates a decryption key at break in time $\bar{t}$ using the same method described above (i.e., the first case in simulating users' decryption keys). $\mathcal{S}$ can simulate it since $\bar{t}$ is not prefix of t^*.

Last, $\mathcal{S}$ can easily answer decryption queries and revoke queries.

– In the challenge phase, $\mathcal{S}$ returns a challenge ciphertext $C^* \leftarrow \mathsf{Enc}(\mathtt{mpk}, m_b, \Lambda^*, t^*)$ to $\mathcal{A}$, where ct_0 can be

$$b = 0 : (h^{a_1 \cdot \alpha_1^{\ell+1} \cdot \bar{r'}}, h^{a_2 \cdot \alpha_2^{\ell+1} \cdot \bar{r'}}, h^{(\alpha_1^{\ell+1} + \alpha_2^{\ell+1}) \cdot \bar{r'}}, g^{z_c \cdot (\delta + \sum_{i=1}^{\ell} \gamma_i \cdot t_i^*) \cdot (\alpha_1^{\ell+1} + \alpha_2^{\ell+1})})$$
$$b = 1 : (h^{a_1 \cdot \alpha_1^{\ell+1} \cdot \bar{r'}}, h^{a_2 \cdot \alpha_2^{\ell+1} \cdot \bar{r'}}, h^{s \cdot \bar{r'}}, g^{z_c \cdot (\delta + \sum_{i=1}^{\ell} \gamma_i \cdot t_i^*) \cdot s})$$

Note that $\bar{r'}$ is a randomly chosen value from $\mathbb{Z}_q$.

Since there are at most k system users and $\mathcal{T}$ times, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\text{FB-ABE}}(\lambda) \leq k \cdot \mathcal{T} \cdot \mathtt{Adv}_{\mathcal{S}}^{Com}(\lambda)$$

5 Security Analysis of FB-PCH

Theorem 5. *The FB-PCH scheme is forward/backward-secure collision resistant if the modified DL-based CH scheme is collision resistant, and the FB-ABE scheme is semantically secure.*

Proof. We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{\text{FB-PCH}}$ be the advantage of the adversary in game $\mathbb{G}_i$. We assume that $\mathcal{A}$ issues at most q queries to the Hash oracle.

– $\mathbb{G}_0$: This is original game for forward/backward-secure collision-resistance.
– $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except the following difference: $\mathcal{S}$ randomly chooses $g \in [1, q]$ as a guess for the index of the query to the Hash oracle at time t^* which returns the chameleon hash $(m^*, h^*, \xi^*, C^*, t^*)$. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s attacking query does not occur in the g-th query. Because we assume the upper-bound of time is $\mathcal{T}$, we have

$$\mathtt{Adv}_0^{\text{FB-PCH}} = q \cdot \mathcal{T} \cdot \mathtt{Adv}_1^{\text{FB-PCH}} \tag{2}$$

– $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that in the g-th query, $\mathcal{S}$ replaces the encrypted trapdoor tr^* in C^* by $\perp$ (i.e., empty value). Below we show that the difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is negligible if the FB-ABE is semantically secure.
Let $\mathcal{S}$ be an attacker against the FB-ABE scheme with semantic security, who is given a public key $\mathtt{pk}^*$ and a set of oracles, aims to distinguish between encryption of M_0 and M_1 under an access structure Λ at time t^*. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- $\mathcal{S}$ sets up $\mathtt{mpk} = \mathtt{pk}^*$ and completes the remainder of Setup honestly.
- $\mathcal{S}$ can honestly answer all queries made by $\mathcal{A}$ using the given set of oracles. In the g-th query, $\mathcal{S}$ submits an identity id, two messages $(tr, 0)$, an access structure Λ^* and a time t^* to his challenger and obtains a challenge ciphertext C^*. Eventually, $\mathcal{S}$ returns (m^*, h^*, ξ^*, C^*) to $\mathcal{A}$, where $h^* = (g^{tr})^{m^*} \cdot R^*, \xi^* = (R^* = g^{\mathsf{r}^*}, \Pi^*)$ and Π^* is a NIZK for $\log(R^*)$. Since the trapdoor tr and the randomness r^* are randomly chosen by $\mathcal{S}$, $\mathcal{S}$ can simulate the adapt queries successfully.

If the encrypted message in C^* is tr, then the simulation is consistent with $\mathbb{G}_1$; Otherwise, the simulation is consistent with $\mathbb{G}_2$. Therefore, if the advantage of $\mathcal{A}$ is significantly different in $\mathbb{G}_1$ and $\mathbb{G}_2$, $\mathcal{S}$ can break the semantic security of the FB-ABE scheme. Hence, we have

$$\left|\mathtt{Adv}_1^{\text{FB-PCH}} - \mathtt{Adv}_2^{\text{FB-PCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\text{FB-ABE}}(\lambda). \tag{3}$$

– $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that in the g-th query, $\mathcal{S}$ outputs a random bit if $\mathcal{A}$ outputs a valid collision $(id^*, h^*, m^*, \xi^*, m'^*, \xi'^*, C^*, C'^*, t^*)$. Below we show that the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible if the modified DL-based CH scheme is collision resistant.

Let $\mathcal{S}$ denote an attacker against the modified DL-based CH, who is given a chameleon public key $\mathtt{pk}^*$, a hash oracle, and an adapt oracle, aims to find a collision which was not simulated by the adapt oracle. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- $\mathcal{S}$ sets the chameleon public key of the g-th query as $\mathtt{pk}^*$, and completes the remainder of Setup honestly.
- $\mathcal{S}$ can answer all adapt queries at different time made by $\mathcal{A}$ by choosing different trapdoors. For the g-th hash query, $\mathcal{S}$ returns (h, ξ, C) to $\mathcal{A}$ as the response to the hash oracle on the hashed message m, where a chameleon hash value $h = (\mathtt{pk}^*)^m \cdot R$, a NIZK proof $\xi = (R = g^{\mathsf{r}}, \Pi)$, and a ciphertext C encrypting "0". For the g-th adapt query, $\mathcal{S}$ returns (m', h, ξ', C') to $\mathcal{A}$, where $\xi' = (R' = g^{\mathsf{r}'}, \Pi')$.
- When $\mathcal{A}$ outputs a collision $(id^*, m^*, \xi^*, m'^*, \xi'^*, h^*, C^*, C'^*, t^*)$, such that the chameleon hash is valid $h^* = (\mathtt{pk}^*)^{m^*} \cdot R^* = (\mathtt{pk}^*)^{m'^*} \cdot R'^*$, and the NIZK proofs (ξ^*, ξ'^*) are valid. We require that either (h^*, m^*, ξ^*, C^*) or $(h^*, m'^*, \xi'^*, C'^*)$ must be a fresh collision, i.e., one that was never queried adapt oracle. Then, $\mathcal{S}$ outputs $(m^*, \xi^*, m'^*, \xi'^*, h^*)$ as a collision to the modified DL-based CH scheme; otherwise, $\mathcal{S}$ aborts the game. Therefore, we have

$$\left|\mathtt{Adv}^{\text{FB-PCH}} - \mathtt{Adv}^{\text{FB-PCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\text{CH}}(\lambda). \tag{4}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\text{FB-PCH}}(\lambda) \leq q \cdot \mathcal{T} \cdot (\mathtt{Adv}_{\mathcal{S}}^{\text{FB-ABE}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\text{CH}}(\lambda)).$$

Theorem 6. *The FB-PCH is indistinguishable if the modified DL-based CH scheme is indistinguishable, and the FB-ABE scheme is semantically secure.*

Proof. We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{\text{FB-PCH}}$ be the advantage of the adversary in game $\mathbb{G}_i$. We assume that $\mathcal{A}$ issues at most q hash queries at each game.

- $\mathbb{G}_0$: This is original game for indistinguishability.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except the following difference: $\mathcal{S}$ randomly chooses $g \in [1, q]$ as a guess for the challenge query at time t^* in the challenge phase. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s attacking query does not occur in the g-th query. Since the upper-bound of time is $\mathcal{T}$, we have

$$\mathtt{Adv}_0^{\text{FB-PCH}} = q \cdot \mathcal{T} \cdot \mathtt{Adv}_1^{\text{FB-PCH}} \tag{5}$$

- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that in the g-th query, $\mathcal{S}$ replaces the encrypted trapdoor tr^* in C^* by $\perp$ (i.e., empty value). By using the same security analysis as described in the above game $\mathbb{G}_2$, we have

$$\left|\mathtt{Adv}_1^{\text{FB-PCH}} - \mathtt{Adv}_2^{\text{FB-PCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\text{FB-ABE}}(\lambda). \tag{6}$$

- $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that in the g-th query, $\mathcal{S}$ directly hashes a message m, instead of calculating the chameleon hash and randomness (h, r) using the trapdoor tr. Below we show the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible if the CH scheme is indistinguishable.
  Let $\mathcal{S}$ denote an attacker against the modified DL-based CH, who is given a chameleon public key $\mathtt{pk}^*$ and a $\mathsf{HashOrAdapt}$ oracle, aims to break the CH's indistinguishability. $\mathcal{S}$ generates a master key pair, and simulates all users' (attribute-based) keys honestly. $\mathcal{S}$ sets the chameleon public key of the g-th query as $\mathtt{pk}^*$.
  In the g-th query, if $\mathcal{A}$ submits $(id, m_0, m_1, \Lambda, \delta, t)$ to $\mathcal{S}$, $\mathcal{S}$ obtains a chameleon hash (h_w, r_w) from his $\mathsf{HashOrAdapt}$ oracle on messages (m_0, m_1), where $w \in [0, 1]$. Then, $\mathcal{S}$ honestly generates a NIZK proof $\xi_w = (R_w = g^{r_w}, \Pi_w)$ and a ciphertext C according the protocol's description (note that $\mathcal{S}$ sets the encrypted trapdoor as $\perp$). Eventually, $\mathcal{S}$ returns (m_w, h_w, ξ_w, C) to $\mathcal{A}$. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break CH's indistinguishability. Hence, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\text{FB-PCH}}(\lambda) \leq \mathtt{Adv}_{\mathcal{S}}^{\text{CH}}(\lambda)).$$

  Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\text{FB-PCH}}(\lambda) \leq q \cdot \mathcal{T} \cdot (\mathtt{Adv}_{\mathcal{S}}^{\text{FB-ABE}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\text{CH}}(\lambda)).$$

Bibliography

[1] S. Agrawal and M. Chase. Fame: fast attribute-based message encryption. In *CCS*, pages 665–682, 2017.

[2] G. Ateniese and B. de Medeiros. On the key exposure problem in chameleon hashes. In *SCN*, pages 165–179, 2004.

[3] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. Redactable blockchain–or–rewriting history in bitcoin and friends. In *EuroS&P*, pages 111–126, 2017.

[4] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig. Chameleon-hashes with ephemeral trapdoors. In *PKC*, pages 152–182, 2017.

[5] D. Derler, K. Samelin, and D. Slamanig. Bringing order to chaos: The case of collision-resistant chameleon-hashes. In *PKC*, pages 462–492, 2020.

[6] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*, 2000.

[7] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.