StrongPCH: Strongly Accountable Policy-based Chameleon Hash for Blockchain Rewriting

I. Security Analysis of New Assumptions

In this section, we present the security analysis of the proposed new assumptions, including the extended DLIN (eDLIN), Computational Bilinear Diffie-Hellman (CBDH), and Decisional Bilinear Diffie-Hellman (DBDH). We prove the security in group $\mathbb{G}$, along with the bilinear maps $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$. For simplicity, we do not consider the isomorphism $\varphi : \mathbb{H} \rightarrow \mathbb{G}$ (i.e., $\mathbb{G} \neq \mathbb{H}$), and we do not include the group elements from group $\mathbb{H}$ in the following theorems. One can easily add these two functions into the following theorems.

Theorem 1: Let $(\epsilon_1, \epsilon_2, \epsilon_T) : \mathbb{Z}_q \rightarrow \{0,1\}^*$ 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, then we define the advantage of the adversary in solving the eDLIN 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

$$
\begin{aligned}
|\mathtt{Adv}^{\mathrm{eDLIN}}_{\mathcal{A}}(\lambda) \quad &= \quad \Pr[\mathcal{A}(q, \epsilon_1(1), \epsilon_1(a_1), \epsilon_1(a_2), \\
&\quad\quad \epsilon_1(1/a_1), \epsilon_1(1/a_2), \epsilon_1(a_1 s_1), \epsilon_1(a_1 s_2), \\
&\quad\quad \epsilon_1(a_2 s_1), \epsilon_1(a_2 s_2), \epsilon_1(s_1/a_1), \epsilon_1(s_2/a_2), \\
&\quad\quad \epsilon_1(z), \epsilon_1(t_0), \epsilon_1(t_1), \epsilon_2(1), \epsilon_T(1)) \\
&= \quad w : (a_1, a_2, s_1, s_2, z, s \xleftarrow{R} \mathbb{Z}_q, w \in (0,1), \\
&\quad\quad t_w = z(s_1 + s_2), t_{1-w} = s)] \\
&\quad\quad -1/2| \leq \frac{10(\mathcal{Q}+15)^2}{q}
\end{aligned}
$$

Proof 1: 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 $p_i \in \mathbb{Z}_q[A_1, A_2, S_1, S_2, Z, S, T_0, T_1]$ are 8-variate polynomials over $\mathbb{Z}_q$, such that $p_0 = 1, p_1 = A_1, p_2 = A_2, p_3 = A_1^{q-2}, p_4 = A_2^{q-2}, p_5 = A_1 \cdot S_1, p_6 = A_1 \cdot S_2, p_7 = A_2 \cdot S_1, p_8 = A_2 \cdot S_2, p_9 = A_1^{q-2} \cdot S_1, p_{10} = A_2^{q-2} \cdot S_2, p_{11} = T_0, p_{12} = T_1$. $\mathcal{S}$ also generates $q_0 = 1, t_0 = 1$. Besides, $(\{\epsilon_{1,i}\}_{i=0}^{12} \in \{0,1\}^*, \{\epsilon_{2,0}\} \in \{0,1\}^*, \{\epsilon_{T,0}\} \in \{0,1\}^*)$ are arbitrary distinct strings, $\mathcal{S}$ then sets those pairs $(p_i, \epsilon_{1,i})$ as L_1. Therefore, the three lists are initialised as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^{12}, L_2 = (q_0, \epsilon_{2,0}), L_T = (t_0, \epsilon_{T,0})$.

At the beginning of the game, $\mathcal{S}$ sends the encoding strings $(\{\epsilon_{1,i}\}_{i=0,\cdots,12}, \epsilon_{2,0}, \epsilon_{T,0})$ to $\mathcal{A}$. After this, $\mathcal{S}$ 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., $\mathcal{O}_2, \mathcal{O}_T$) is 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 $a_1, a_2, s_1, s_2, z, s \in \mathbb{Z}_q$, generates $t_b = z(s_1 + s_2)$ and $t_{1-b} = s$. $\mathcal{S}$ sets $A_1 = a_1, A_2 = a_2, S_1 = s_1, S_2 = s_2, Z = z, S = s, T_0 = t_b, T_1 = t_{1-b}$. The simulation by $\mathcal{S}$ is perfect (and disclose nothing to $\mathcal{A}$ about b) unless the `abort` event happens. Thus, we bound the probability of event `abort` by analyzing the following cases:

1) $p_i(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = p_j(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$: The polynomial $p_i \neq p_j$ due to the construction method of L_1, and $(p_i - p_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$ is a non-zero polynomial of degree $[0,2]$ or $q-2$ ($q-2$ is produced by A_1^{q-2}). We have $A_1 \cdot A_1^{q-2} = A_1^{q-1} \equiv 1(\mod q)$, so $A_1 \cdot A_1^{q-2} \cdot S_1 \equiv A_1 \cdot S_1 (\mod q)$ and the maximum degree of $A_1 \cdot S_1 (p_i - p_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$ is 4. By using Lemma 1 in [1], we have $\Pr[(p_i - p_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = 0] \leq \frac{4}{q}$ and thus $\Pr[p_i(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = p_j(a_1, a_2, s_1, s_2, z, s, t_0, t_1)] \leq \frac{4}{q}$. Therefore, we have the `abort` probability is $\Pr[\mathsf{abort}_1] \leq \frac{4}{q}$.
2) $q_i(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = q_j(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$: The polynomial $q_i \neq q_j$ due to the construction method of L_2, and $(q_i - q_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$ is a non-zero polynomial of degree 0. The `abort` probability is "0" (i.e., the maximum degree is "0" since the list L_2 contains a single string $\epsilon_{(2,0)}$ only). Recall that we do not include elements from group $\mathbb{H}$ here.
3) $t_i(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = t_j(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$: The polynomial $t_i \neq t_j$ due to the construction method of L_3, and $(t_i - t_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$ is a non-zero polynomial of degree 0, 1, 2, or $2q-4$ ($2q-4$ is produced by $A_1^{q-2} \cdot A_2^{q-2} \cdot S_1 \cdot S_2$, and we denote it as $A^{2q-4} \cdot S^2$).

Since $A^2 \cdot A^{2q-4} = (A^{q-1})^2 \equiv 1(\mod q), A^2 \cdot A^{2q-4} \cdot S^2 \equiv (A \cdot S)^2(\mod q)$, so the maximum degree of $(A_1 \cdot S_1)^2(t_i - t_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1)$ is 6. Therefore, we have $\Pr[(t_i - t_j)(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = 0] \leq \frac{6}{q}$ and thus $\Pr[t_i(a_1, a_2, s_1, s_2, z, s, t_0, t_1) = t_j(a_1, a_2, s_1, s_2, z, s, t_0, t_1)] \leq \frac{6}{q}$.

By summing over all valid pairs (i, j) in each case (i.e., at most $2\binom{\mathcal{Q}+15}{2}$ pairs), we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \\ &\leq 2\binom{\mathcal{Q}+15}{2} \cdot (\frac{4}{q} + \frac{6}{q}) \leq \frac{10(\mathcal{Q}+15)^2}{q}.\end{aligned}$$

Theorem 2: 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. ϵ_1 maps all $a \in \mathbb{Z}_q$ to the string representation $\epsilon_1(g^a)$ of $g^a \in \mathbb{G}$. Similarly, ϵ_2 for $\mathbb{H}$ and ϵ_T for $\mathbb{G}_{\mathbb{T}}$. 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 CBDH 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

$$\begin{aligned}|\mathtt{Adv}_{\mathcal{A}}^{\text{CBDH}}(\lambda) \quad = \quad & \Pr[\mathcal{A}(q, \epsilon_1(1), \epsilon_1(a), \epsilon_1(b), \epsilon_2(1) \\ & \epsilon_2(a), \epsilon_2(b), \epsilon_2(c), \epsilon_2(ab), \epsilon_2(1/ab)) \\ & = \epsilon_T(c/ab)]| \leq \frac{12(\mathcal{Q}+10)^2}{q}\end{aligned}$$

Proof 2: Let $\mathcal{S}$ play the following game with $\mathcal{A}$. $\mathcal{S}$ 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 $p_i \in \mathbb{Z}_q[A, B]$ are 2-variate polynomials over $\mathbb{Z}_q$, such that $p_0 = 1, p_1 = A, p_2 = B$. $q_i \in \mathbb{Z}_q[A, B]$ are 2-variate polynomials over $\mathbb{Z}_q$, such that $q_0 = 1, q_1 = A, q_2 = B, q_3 = C, q_4 = AB, q_5 = AB^{q-2}$. $t_i \in \mathbb{Z}_q[A, B, C]$ are 3-variate polynomials over $\mathbb{Z}_q$, such that $t_0 = C(AB)^{q-2}$. Besides, $\{\epsilon_{1,i}\}_{i=0}^2 \in \{0,1\}^*, \{\epsilon_{2,i}\}_{i=0}^5 \in \{0,1\}^*, \{\epsilon_{T,0}\} \in \{0,1\}^*$ are arbitrary distinct strings. Therefore, the three lists are initialised as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^2, L_2 = \{(q_i, \epsilon_{2,i})\}_{i=0}^5, L_T = (t_0, \epsilon_{T,0})$.

At the beginning of the game, $\mathcal{S}$ sends the encoding strings $(\{\epsilon_{1,i}\}_{i=0,\cdots,2}, \{\epsilon_{2,i}\}_{i=0,\cdots,5}, \epsilon_{T,0})$ to $\mathcal{A}$. After this, $\mathcal{S}$ simulates the group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ and the bilinear pairing $\hat{\mathsf{e}}$ using the same method as described in Theorem 1. After querying at most $\mathcal{Q}$ times of corresponding oracles, $\mathcal{A}$ terminates and outputs $\epsilon_T(c/ab)$. At this point, $\mathcal{S}$ chooses random $a, b, c \in \mathbb{Z}_q$. $\mathcal{S}$ sets $A = a, B = b, C = c$. The simulation by $\mathcal{S}$ is perfect unless the `abort` event happens. Thus, we bound the probability of event `abort` by analyzing the following cases:

1) $p_i(a, b, c) = p_j(a, b, c)$: The polynomial $p_i \neq p_j$ due to the construction method of L_1, and $(p_i - p_j)(a, b, c)$ is a non-zero polynomial of degree $[0, 1]$. The maximum degree of $(p_i - p_j)(a, b, c)$ is 1. By using Lemma 1 in [1], we have $\Pr[(p_i - p_j)(a, b, c) = 0] \leq \frac{1}{q}$ and thus $\Pr[p_i(a, b, c) = p_j(a, b, c)] \leq \frac{1}{q}$. So, we have the `abort` probability is $\Pr[\mathsf{abort}_1] \leq \frac{1}{q}$.
2) $q_i(a, b, c) = q_j(a, b, c)$: The polynomial $q_i \neq q_j$ due to the construction method of L_2, and $(q_i - q_j)(a, b, c)$ is a non-zero polynomial of degree $[0, 2]$, or q-2 (q-2 is produced by AB^{q-2}). Since $AB \cdot AB^{q-2} = AB^{q-1} \equiv 1(\mod q)$, we have $AB(q_i - q_j)(a, b, c)$ is non-zero polynomial of degree $[0, 4]$, so the `abort` probability is bounded by $\Pr[\mathsf{abort}_2] \leq \frac{4}{q}$.
3) $t_i(a, b, c) = c/ab$: The degree of p_i is $[0, 1]$, and the the degree of q_i is $[0, 4]$. Since $CAB \cdot (AB)^{q-2} \equiv CAB(\mod q)$, we have $CAB(t_i - t_j)(a, b, c)$ is non-zero polynomial of degree $[0, 7]$, So, we have $\Pr[(t_i - t_j)(a, b, c) = 0] \leq \frac{7}{q}$ and thus $\Pr[\mathsf{abort}_3] \leq \frac{7}{q}$.

By summing over all valid pairs (i, j) in each case (i.e., at most $\binom{\mathcal{Q}_{\epsilon_1}+3}{2} + \binom{\mathcal{Q}_{\epsilon_2}+6}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}$ pairs), and $\mathcal{Q}_{\epsilon_1} + \mathcal{Q}_{\epsilon_2} + \mathcal{Q}_{\epsilon_T} = \mathcal{Q} + 10$, we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \\ &\leq [\binom{\mathcal{Q}_{\epsilon_1}+3}{2} + \binom{\mathcal{Q}_{\epsilon_2}+6}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}] \\ &\cdot (\frac{1}{q} + \frac{4}{q} + \frac{7}{q}) \leq \frac{12(\mathcal{Q}+10)^2}{q}.\end{aligned}$$

Theorem 3: 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. ϵ_1 maps all $a \in \mathbb{Z}_q$ to the string representation $\epsilon_1(g^a)$ of $g^a \in \mathbb{G}$. Similarly, ϵ_2 for $\mathbb{H}$ and ϵ_T for $\mathbb{G}_{\mathbb{T}}$. If $(a, b, d, e, f, \{c_i, l_i\}) \xleftarrow{R} \mathbb{Z}_q$ and encodings $\epsilon_1, \epsilon_2, \epsilon_T$ are randomly chosen, we define the advantage of the adversary in solving the DBDH 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

$$\begin{aligned}|\mathtt{Adv}_{\mathcal{A}}^{\text{DBDH}}(\lambda) \quad = \quad & \Pr[\mathcal{A}(q, \epsilon_1(1), \epsilon_1(a), \epsilon_1(b), \epsilon_1(f), \\ & \epsilon_1(ed), \{\epsilon_1(ec_i), \epsilon_1(el_i)\} \\ = \quad & w : (a, b, d, e, f \xleftarrow{R} \mathbb{Z}_q, \{c_i, l_i\} \in \mathbb{Z}_q, \\ & w \in (0, 1), t_w = ab + edc_i, \\ & t_{1-w} = fb + edl_i)] - 1/2| \\ \leq \quad & \frac{6(\mathcal{Q}+11)^2}{q}\end{aligned}$$

Proof 3: Let $\mathcal{S}$ play the following game with $\mathcal{A}$. $\mathcal{S}$ 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 $p_i \in \mathbb{Z}_q[A, B, C, D, E, F, L, T_0, T_1]$ are 9-variate polynomials over $\mathbb{Z}_q$, such that $p_0 = 1, p_1 = A, p_2 = B, p_3 = F, p_4 = ED, p_5 = EC_i, p_6 = EL_i, p_7 = T_w, p_8 = T_{1-w}$. $\mathcal{S}$ also generates $q_0 = 1, t_0 = 1$. Besides, $\{\epsilon_{1,i}\}_{i=0}^8 \in \{0,1\}^*, \epsilon_{2,0} \in \{0,1\}^*, \epsilon_{T,0} \in \{0,1\}^*$ are arbitrary distinct strings. Therefore, the three lists are initialised as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^8, L_2 = (q_0, \epsilon_{2,0}), L_T = (t_0, \epsilon_{T,0})$.

At the beginning of the game, $\mathcal{S}$ sends the encoding strings $(\{\epsilon_{1,i}\}_{i=0,\cdots,8}, \epsilon_{2,0}, \epsilon_{T,0})$ to $\mathcal{A}$. After this, $\mathcal{S}$ simulates the group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ and the bilinear pairing $\hat{\mathsf{e}}$ using the same method as described in Theorem 1. After querying at most $\mathcal{Q}$ times of corresponding oracles, $\mathcal{A}$ terminates and outputs a guess $w' = \{0, 1\}$. At this point, $\mathcal{S}$ chooses random $a, b, d, f, c_i, l_i \in \mathbb{Z}_q$, generates $t_w = ab + edc_i$ and $t_{1-w} = fb + edl_i$. $\mathcal{S}$ sets $A = a, B = b, D = d, E = e, F = f, C_i = c_i, L_i = l_i, T_0 = t_w, T_1 = t_{1-w}$. The simulation by $\mathcal{S}$ is perfect unless the `abort` event happens. Thus, we bound the probability of event `abort` by analyzing the following cases:

1) $p_i(a,b,c,\cdots) = p_j(a,b,c,\cdots)$: The polynomial $p_i \neq p_j$ due to the construction method of L_1, and $(p_i - p_j)(a,b,c,\cdots)$ is a non-zero polynomial of degree $[0,3]$. The maximum degree of $(p_i - p_j)(a,b,c,\cdots)$ is 3. By using Lemma 1 in [1], we have $\Pr[(p_i - p_j)(a,b,c,\cdots) = 0] \leq \frac{3}{q}$ and thus $\Pr[p_i(a,b,c,\cdots) = p_j(a,b,c,\cdots)] \leq \frac{3}{q}$. So, we have the abort probability $\Pr[\mathsf{abort}_1] \leq \frac{3}{q}$.
2) $q_i(a,b,c,\cdots) = q_j(a,b,c,\cdots)$: The polynomial $q_i \neq q_j$ due to the construction method of L_2, and $(q_i - q_j)(a,b,c,\cdots)$ is a non-zero polynomial of degree 0. The abort probability is "0" (i.e., the maximum degree is "0" as L_2 contains a single string $\epsilon_{(2,0)}$ only).
3) $t_i(a,b,c,\cdots) = t_j(a,b,c,\cdots)$: Since the degree of p_i is $[0,3]$, we have $\Pr[(t_i - t_j)(a,b,c,\cdots) = 0] \leq \frac{3}{q}$ and $\Pr[\mathsf{abort}_3] \leq \frac{3}{q}$.

By summing over all valid pairs (i,j) in each case (i.e., at most $\binom{\mathcal{Q}_{\epsilon_1}+9}{2} + \binom{\mathcal{Q}_{\epsilon_2}+1}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}$ pairs), and $\mathcal{Q}_{\epsilon_1} + \mathcal{Q}_{\epsilon_2} + \mathcal{Q}_{\epsilon_T} = \mathcal{Q} + 11$, we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \\ &\leq [\binom{\mathcal{Q}_{\epsilon_1}+9}{2} + \binom{\mathcal{Q}_{\epsilon_2}+1}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}] \\ &\cdot (\frac{3}{q} + \frac{3}{q}) \leq \frac{6(\mathcal{Q}+11)^2}{q}.\end{aligned}$$

II. Security Analysis of StrongPCH

Theorem 4: The StrongPCH scheme is indistinguishable if the CH scheme is indistinguishable, and the Σ scheme is anonymous.

Proof 4: We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{\mathsf{SPCH}}$ denote the advantage of the adversary in game $\mathbb{G}_i$. Assume that $\mathcal{A}$ issues at most $n(\lambda)$ hash queries, which include a $\mathsf{HashOrAdapt}$ query (we call it g-th query).

- $\mathbb{G}_0$: This is original game for indistinguishability.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except that in the g-th hash query, simulator $\mathcal{S}$ directly hashes a message $(b, \mathsf{r}) \leftarrow \mathsf{Hash}_{\mathsf{CH}}(\mathtt{pk}^*, m)$, without calculating the chameleon hash and randomness (b, r) using the Adapt algorithm. Below we show the difference between $\mathbb{G}_0$ and $\mathbb{G}_1$ is negligible if the CH scheme is indistinguishable.
  Let $\mathcal{S}$ ben an attacker against 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 the master key pairs and user's key pairs honestly. $\mathcal{S}$ sets the chameleon public key of the g-th hash query as $\mathtt{pk}^*$. If $\mathcal{A}$ submits two messages (m_0, m_1, Λ) to $\mathcal{S}$ in the g-th query, $\mathcal{S}$ first obtains a chameleon hash (b_w, r_w) from his $\mathsf{HashOrAdapt}$ oracle on messages (m_0, m_1). Then, $\mathcal{S}$ generates a signature $\sigma \leftarrow \mathsf{Sign}_{\Sigma}(\mathtt{sk}, c)$, and a ciphertext $C \leftarrow \mathsf{Enc}_{\mathsf{ABET}}(\mathtt{mpk}_{\mathsf{ABET}}, \perp, \Lambda)$. Note that the signed message c and the verification key apk can be generated using $\mathtt{pk}^*$ and user's secret key $\mathtt{sk}$. Eventually, $\mathcal{S}$ returns $(m_w, b_w, \mathsf{r}_w, C, apk, c, \sigma)$ to $\mathcal{A}$. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the CH's indistinguishability. Hence, we have
$$\left|\mathtt{Adv}_0^{\mathsf{SPCH}} - \mathtt{Adv}_1^{\mathsf{SPCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda). \tag{1}$$
- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ replaces the encrypted secret key $\mathtt{sk}$ in C by $\perp$ (i.e., empty value). Below we show the difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is negligible if the ABET scheme is semantically secure.
  Let $\mathcal{S}$ denotes an attacker against ABET, who is given a public key $\mathtt{pk}^*$, a key generation oracle and a decryption oracle, aims to break ABET's semantic security. $\mathcal{S}$ sets the game for $\mathcal{A}$ by creating users with the corresponding key pairs $\{(\mathtt{sk}, \mathtt{pk})\}$. $\mathcal{S}$ randomly chooses a user as attribute authority, and sets his public key as $\mathtt{pk}^*$.
  $\mathcal{S}$ simulates the g-th hash query as follows. First, $\mathcal{S}$ sends two messages $(M_0, M_1) = (\mathtt{sk}, \perp)$ to his challenger, and obtains a ciphertext $C^* \leftarrow \mathsf{Enc}(\mathtt{pk}^*, M_w, \Lambda)$. Second, $\mathcal{S}$ simulates a hash-randomness (b, r) and a message-signature (c, σ) according to the protocol specification. Eventually, $\mathcal{S}$ returns a tuple $(m, b, \mathsf{r}, C^*, apk, c, \sigma)$ to $\mathcal{A}$. Note that $\mathcal{S}$ can honestly answer $\mathcal{A}$'s key generation and decryption queries. If the encrypted message is the secret key $\mathtt{sk}$, the simulation is consistent with $\mathbb{G}_1$; Otherwise, the simulation is consistent with $\mathbb{G}_2$. If the advantage of $\mathcal{A}$ is significantly different in $\mathbb{G}_1$ and $\mathbb{G}_2$, $\mathcal{S}$ can break ABET's semantic security.
$$\left|\mathtt{Adv}_1^{\mathsf{SPCH}} - \mathtt{Adv}_2^{\mathsf{SPCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda). \tag{2}$$
- $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that $\mathcal{S}$ outputs a random bit if a **Link** event happens where $\mathcal{A}$ links a message-signature (c, σ) to an honest signer. Since the underlying signature Σ is anonymous, the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible, we have
$$\left|\mathtt{Adv}_2^{\mathsf{SPCH}} - \mathtt{Adv}_3^{\mathsf{SPCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\Sigma}(\lambda). \tag{3}$$
  Combining the above results together, we have
$$\mathtt{Adv}_{\mathcal{A}}^{\mathsf{SPCH}}(\lambda) \quad \leq \quad n(\lambda)(\mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\Sigma}(\lambda)).$$

Theorem 5: The StrongPCH scheme is collision-resistant if the ABET scheme is semantically secure, and the CH scheme is collision-resistant.

Proof 5: We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 3$ and let $\mathtt{Adv}_i^{\mathsf{SPCH}}$ denote the advantage of the adversary in game $\mathbb{G}_i$. Assuming that $\mathcal{A}$ issues at most q queries to the $\mathsf{Hash'}_{\mathsf{SPCH}}$ oracle at each game.

- $\mathbb{G}_0$: This is original game for collision-resistant.
- $\mathbb{G}_1$: 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 $\mathsf{Hash'}$ oracle which returns the chameleon hash $(\mathtt{pk}_{\mathsf{CH}}, m^*, b^*, \mathsf{r}^*, \cdots)$. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s attacking query does not occur in the g-th query. Therefore, we have
$$\mathtt{Adv}_0^{\mathsf{SPCH}} = q \cdot \mathtt{Adv}_1^{\mathsf{SPCH}} \tag{4}$$
- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that in the g-th query, the encrypted message $\mathtt{sk}_{\mathsf{CH}}$ is replaced by $\perp$ (i.e., empty value). The difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is negligible if the ABET scheme is semantically secure. The reduction is using the same method described in previous game $\mathbb{G}_2$. Hence, we have
$$\left|\mathtt{Adv}_1^{\mathsf{SPCH}} - \mathtt{Adv}_2^{\mathsf{SPCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda). \tag{5}$$

- $\mathbb{G}_3$: 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 $(\mathtt{pk}_{\mathsf{CH}}, m'^*, b^*, \mathsf{r}'^*, \cdots)$. Below we show that the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible if the CH is collision-resistant.

  Let $\mathcal{S}$ denote an attacker against CH with collision-resistance, who is given 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 up $\mathtt{pk}_{\mathsf{CH}} = \mathtt{pk}^*$ and completes the remainder of Setup honestly.
  - $\mathcal{S}$ simulates the response to the Hash' query as $(m, b, \mathsf{r}, C, apk, c, \sigma)$, where (m, b, r) is returned from his Hash' oracle, ciphertext C encrypts $\perp$, and the message-signature (c, σ) is generated according to the protocol specification. Similarly, $\mathcal{S}$ can simulate a chameleon hash $(m^*, b^*, \mathsf{r}^*, C^*, apk^*, c^*, \sigma^*)$ at the g-th query. For the adapt query, $\mathcal{S}$ obtains a new randomness r' from his Adapt' oracle and returns $(m', b, \mathsf{r}', C, apk', c', \sigma')$ to $\mathcal{A}$.
  - At some point, if $\mathcal{A}$ outputs a collision $(\mathtt{pk}_{\mathsf{CH}}, m^*, m'^*, b^*, \mathsf{r}^*, \mathsf{r}'^*, C^*, apk^*, c^*, \sigma^*, apk'^*, c'^*, \sigma'^*)$ with all the verification checks hold, $\mathcal{S}$ outputs $(\mathtt{pk}_{\mathsf{CH}}, m'^*, b^*, \mathsf{r}'^*, \cdots)$ as a valid collision to the CH scheme; otherwise, $\mathcal{S}$ aborts the game. Therefore, we have

$$\left|\mathtt{Adv}_2^{\mathsf{SPCH}} - \mathtt{Adv}_3^{\mathsf{SPCH}}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda). \tag{6}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\mathsf{SPCH}}(\lambda) \quad \leq \quad q(\mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda)).$$

Theorem 6: The StrongPCH scheme is strongly accountable if the digital signature scheme Σ is EUF-CMA secure, and the ABET scheme is traceable.

Proof 6: We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 2$ and let $\mathtt{Adv}_i^{\mathsf{SPCH}}$ denote the advantage of the adversary in game $\mathbb{G}_i$.

- $\mathbb{G}_0$: This is original game for strong accountability.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except that $\mathcal{S}$ outputs a random bit if a **Forge** event happens where $\mathcal{A}$ outputs $(m, b, \mathsf{r}, C, apk^*, c^*, \sigma^*)$, such that σ^* is a valid signature under $apk^* = \mathsf{KeyGen}'_{\Sigma}(pp_{\Sigma}, 0, \mathtt{sk}^*)$ and c^*, and σ^* is not previously simulated by $\mathcal{S}$.

  Let $\mathcal{S}$ be a forger against Σ, who is given a public key $\mathtt{pk}^*$ and a signing oracle $\mathcal{O}^{\mathsf{Sign}}$, aims to break the EUF-CMA security of Σ. $\mathcal{S}$ sets the game for $\mathcal{A}$ by creating k users with the corresponding key pairs. $\mathcal{S}$ randomly selects a user and guesses that the **Forge** event will happen to the user. $\mathcal{S}$ sets the user's public key as $\mathtt{pk}^*$. Note that the corresponding verification key apk^* can be computed by $\mathcal{S}$ using the master key pair. $\mathcal{S}$ randomly chooses a user as attribute authority, and sets its master key pair as $(\mathtt{sk}_{\mathsf{ABET}}, \mathtt{pk}_{\mathsf{ABET}})$.

  $\mathcal{S}$ simulates a hash query as follows. First, $\mathcal{S}$ chooses a chameleon secret key $\mathtt{sk}_{\mathsf{CH}}$ and generates a ciphertext as $C \leftarrow \mathsf{Enc}(\mathtt{pk}_{\mathsf{ABET}}, \mathtt{sk}_{\mathsf{CH}}, \Lambda)$. Second, $\mathcal{S}$ sends a message $c \leftarrow \mathsf{KeyGen}_{\sigma}(pp, Dlog(\mathtt{pk}^*), \mathtt{sk}_{\mathsf{CH}})$ to his signing oracle $\mathcal{O}^{\mathsf{Sign}}$, and obtains a signature σ. Note that the signed message c can be perfectly simulated by $\mathcal{S}$ due to the homomorphic property of Σ regarding keys and signatures (e.g., $c = \mathtt{pk}^* \cdot h^{\mathtt{sk}_{\mathsf{CH}}}$). Eventually, $\mathcal{S}$ generates a chameleon hash (b, r) according to the protocol specification, and returns a tuple $(m, b, \mathsf{r}, C, \mathtt{apk}, c, \sigma)$ to $\mathcal{A}$. $\mathcal{S}$ records all the simulated message-signature pairs by including them to a set $\mathcal{Q}$. $\mathcal{S}$ also simulates an adapt query honestly using the chameleon secret key $\mathtt{sk}_{\mathsf{CH}}$.

  If **Forge** event occurs, such that $\mathcal{A}$ outputs $(m, b, \mathsf{r}, C, apk^*, c^*, \sigma^*)$, $\mathcal{S}$ will check whether:
  - the verification key apk^* is associated with the challenge user $\mathtt{pk}^*$;
  - the message-signature pair was not previously simulated by $\mathcal{S}$, which is $(c^*, \sigma^*) \notin \mathcal{Q}$;
  - $1 \stackrel{?}{=} \mathsf{Verify}(\mathtt{pk}_{\mathsf{CH}}, m, b, \mathsf{r}, C, apk^*, c^*, \sigma^*)$.

  If all the above conditions hold, $\mathcal{S}$ confirms that it as a successful forgery, and outputs σ^* as its own forgery; Otherwise, $\mathcal{S}$ aborts the game. Since at most k users involved in the game, we have

$$\left|\mathtt{Adv}_0^{\mathsf{SPCH}} - \mathtt{Adv}_1^{\mathsf{SPCH}}\right| \leq k \cdot \mathtt{Adv}_{\mathcal{S}}^{\Sigma}(\lambda). \tag{7}$$

- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_0$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Forge}'$ event happens where $\mathcal{A}$ outputs $(m', b, \mathsf{r}', C, apk^{*'}, c^{*'}, \sigma^{*'})$, such that $\sigma^{*'}$ is a valid signature under apk^* and c^*, and $\sigma^{*'}$ is not previously simulated by $\mathcal{S}$. The reduction is performed using the same method described as above. Hence, we have

$$\left|\mathtt{Adv}_1^{\mathsf{SPCH}} - \mathtt{Adv}_2^{\mathsf{SPCH}}\right| \leq k \cdot \mathtt{Adv}_{\mathcal{S}}^{\Sigma}(\lambda). \tag{8}$$

To this end, $\mathcal{A}$ has no advantage in game $\mathbb{G}_2$. The adversary $\mathcal{A}$ becomes a passive one after the first two games. Since the ABET scheme is traceable, we have

$$\mathtt{Adv}_2^{\mathsf{SPCH}} \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda). \tag{9}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{\mathsf{PCH}^*}(\lambda) \quad \leq \quad 2k \cdot \mathtt{Adv}_{\mathcal{S}}^{\Sigma}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda)$$

III. Security Analysis of ABET

In this section, we show the security analysis of ABET scheme. The semantic security of ABET consists of a set of hybrids, where each hybrid describes how simulator $\mathcal{S}$ interacts with adversary $\mathcal{A}$. The first hybrid is the one where $\mathcal{S}$ and $\mathcal{A}$ interacts according to the original semantic security game. In the second hybrid, we rewrite ABET scheme in a compact form by interpreting the outputs of random oracle appropriately and using the compact group representation [2] to represent group elements. In the following hybrids, the indistinguishability between two hybrids can be either computationally-close or statistically-close. We stress that the security analysis here is similar to the proof described in [3], except the indistinguishability between two hybrids with computationally-close is reduced to the proposed eDLIN assumption. The security reduction on traceability states that, if an adversary $\mathcal{A}$ extracts a message id from a decryption key, there exists an extractor E who can use the extracted message to hold the commitment scheme's extractability.

IV. Security Analysis of Signature Σ

In this section, we present the security analysis of the proposed Σ scheme, including unforgeability, and anonymity.

A. Unforgeability

Theorem 7: The proposed Σ scheme achieves selective EUF-CMA security if the proposed CBDH assumption is held in the asymmetric pairing groups.

Proof 7: Let $\mathcal{S}$ denote a CBDH problem solver, who is given $(g^a, g^b, h^a, h^b, h^c, h^{ab}, h^{1/ab})$, aims to compute $\hat{\mathsf{e}}(g,h)^{c/ab}$. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating system users with corresponding key pairs $\{(\mathtt{sk}, \mathtt{pk})\}$, where $\mathtt{pk} = h^{\mathtt{sk}}$. $\mathcal{S}$ randomly selects a user and sets up its verification key as $\mathtt{pk} = h^a$. In particular, $\mathcal{S}$ selects an index g and guesses that the forge event will happen to the g-th query on a challenge message m^* and an ephemeral public key $h^{esk} = h^c$. $\mathcal{S}$ also sets up a master public key as $g^\beta = g^b$, and completes the remainder of the setup honestly.
- $\mathcal{S}$ simulates hash queries as follows. If $\mathcal{A}$ queries the challenge message m^*, $\mathcal{S}$ returns $g^r \cdot g^{b_i}$ to $\mathcal{A}$, where $(r, b_i) \in \mathbb{Z}_q$ are randomly chosen by $\mathcal{S}$. Otherwise, $\mathcal{S}$ returns g^{b_i} to $\mathcal{A}$ as the response to a hash query on message M.

  If $\mathcal{A}$ issues a signing query on a message m, $\mathcal{S}$ simulates a signature as $\sigma = (\sigma', \sigma'') = (g^{b_i \cdot r_i}, h^{\alpha(b_i \cdot r_i - a \cdot b)/b_i})$, and the corresponding verification key is simulated as $\hat{\mathsf{e}}(g^b, \mathtt{pk})^\alpha$. Note that the exponents α, b_i are randomly chosen by $\mathcal{S}$.
- When forge event occurs, i.e., $\mathcal{A}$ outputs $\sigma^* = (\sigma^{*'}, \sigma^{*''})$, where $\sigma^{*'} = g^{ab} \cdot g^{c(r+b_i)}$ and $\sigma^{*''} = h^{\alpha \cdot c}$, $\mathcal{S}$ checks whether:
  - the forging event happens at g-th query;
  - the message-signature pair (m^*, σ^*) was not previously simulated by $\mathcal{S}$.
  - the signature is valid $\hat{\mathsf{e}}(\sigma^{*'}, h^\alpha) = \hat{\mathsf{e}}(g^\beta, \mathtt{pk})^\alpha \cdot \hat{\mathsf{e}}(g^r \cdot g^{b_i}, \sigma^{*''})$.

  If all the above conditions hold, $\mathcal{S}$ confirms that it as a successful forgery from $\mathcal{A}$, and extracts the solution $\hat{\mathsf{e}}(g,h)^{c/ab} = [\hat{\mathsf{e}}(\sigma^{*'}, h^{1/ab})/\hat{\mathsf{e}}(g,h)]^{1/(r+b_i)}$ to the CBDH assumption.

B. Anonymity

Theorem 8: The proposed Σ scheme achieves anonymity if the proposed DBDH assumption is held in the asymmetric pairing groups.

Proof 8: Let $\mathcal{S}$ denote a DBDH problem distinguisher, who is given $(g^a, g^b, g^f, g^{ed}, g^{ec_i}, g^{el_i}, \forall i \in [q], h^a, h^b, h^f, h^{ed}, h^{ec_i}, h^{el_i}, \forall i \in [q])$, and aims to distinguish $T_w = g^{ab} \cdot g^{edc_i}$ and $T_{1-w} = g^{fb} \cdot g^{edl_i}$. We add the corresponding instances from group $\mathbb{H}$ for simulating message-signature pairs, and these extra instances are no help in solving the DBDH problem. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.

- Setup: $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating system users. $\mathcal{S}$ randomly selects two challenge users and sets $\mathtt{pk}_0 = h^a, \mathtt{pk}_1 = h^f$ and generates key pair for other users honestly. $\mathcal{S}$ also sets $g^\beta = g^b$, and computes the remainder of the setup honestly.
- $\mathcal{S}$ simulates user $\mathtt{pk}_0$'s signatures as follows.
  - $\mathcal{S}$ simulates a signature as $\sigma = (\sigma', \sigma'') = (T_0, h^{ec_i \alpha})$ on a message m; Note that the randomness esk is implicitly sets as c_i, and α is chosen by $\mathcal{S}$.
  - $\mathcal{S}$ simulates a verification key as $\hat{\mathsf{e}}(g^b, h^a)^\alpha$, and sets $g^d = \mathtt{H}(m)$.
  - $\mathcal{S}$ returns $(m, \sigma, \hat{\mathsf{e}}(g,h)^{ab\alpha})$ to $\mathcal{A}$.

  $\mathcal{S}$ can simulate user $\mathtt{pk}_1$'s signature using the same method described above.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{S}$ can break the DBDH assumption.

References

[1] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *EUROCRYPT*, 1997, pp. 256–266.

[2] J. Chen, R. Gay, and H. Wee, "Improved dual system abe in prime-order groups via predicate encodings," in *EUROCRYPT*, 2015, pp. 595–624.

[3] S. Agrawal and M. Chase, "Fame: fast attribute-based message encryption," in *CCS*, 2017, pp. 665–682.