# Accountable Fine-Grained Blockchain Rewriting in the Permissionless Setting

Yangguang Tian, Bowen Liu, Yingjiu Li, Pawel Szalachowski, Jianying Zhou

## I. Security Analysis of New 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. $\epsilon_1$ maps all $a \in \mathbb{Z}_q$ to the string representation $\epsilon_1(g^a)$ of $g^a \in \mathbb{G}$. Similarly, $\epsilon_2$ for $\mathbb{H}$ and $\epsilon_T$ for $\mathbb{G}_\mathbb{T}$. If $(a, b, c, d, \{z_i\}_{i\in[1,q']}) \xleftarrow{\text{R}} \mathbb{Z}_q$ and encodings $\epsilon_1, \epsilon_2, \epsilon_T$ are randomly chosen, then we define the advantage of the adversary in solving the $q'$-type with at most $\mathcal{Q}$ queries to the group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ and the bilinear pairing $\hat{\text{e}}$ as

$$\begin{aligned}
\texttt{Adv}_{\mathcal{A}}^{q'\text{-}type}(\lambda) &= |\Pr[\mathcal{A}(q, \epsilon_1(1), \epsilon_1(a), \epsilon_1(c), \epsilon_1((ac)^2), \\
&\quad \epsilon_1(abd), \epsilon_1(d/ab), \epsilon_1(z_i), \epsilon_1(acz_i), \\
&\quad \epsilon_1(ac/z_i), \epsilon_1(a^2cz_i), \epsilon_1(b/z_i^2), \\
&\quad \epsilon_1(b^2/z_i^2), \epsilon_1(acz_i/z_j), \epsilon_1(bz_i/z_j^2), \\
&\quad \epsilon_1(abcz_i/z_j), \epsilon_1((ac)^2z_i/z_j), \\
&\quad \epsilon_2(1), \epsilon_2(b), \epsilon_2(abd), \epsilon_2(abcd), \\
&\quad \epsilon_2(d/ab), \epsilon_2(c), \epsilon_2(cd/ab), \\
&= b : (a, b, c, d, \{z_i\}_{i\in[1,q']}, s \xleftarrow{\text{R}} \mathbb{Z}_q, b \in (0,1), \\
&\quad t_b = abc, t_{1-b} = s)] \\
&\quad -1/2| \le \frac{16(\mathcal{Q}+q'+22)^2}{q}
\end{aligned}$$

*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 $p_i \in \mathbb{Z}_q[A, B, C, D, Z_i, Z_j, T_0, T_1]$ are 8-variate polynomials over $\mathbb{Z}_q$ (note that $i \neq j$), such that $p_0 = 1, p_1 = A, p_2 = C, p_3 = (AC)^2, p_4 = ABD, p_5 = D/AB, p_6 = Z_i, p_7 = ACZ_i, p_8 = AC/Z_i, p_9 = A^2CZ_i, p_{10} = B/Z_i^2, p_{11} = B^2/Z_i^2, p_{12} = ACZ_i/Z_j, p_{13} = BZ_i/Z_j^2, p_{14} = ABCZ_i/Z_j, p_{15} = (AC)^2Z_i/Z_j, q_0 = 1, q_1 = B, q_2 = ABD, q_3 = ABCD, q_4 = D/AB, q_5 = C, q_6 = CD/AB, p_{16} = T_0, p_{17} = T_1, t_0 = 1$, and $(\{\epsilon_{1,i}\}_{i=0}^{16} \in \{0,1\}^*, \{\epsilon_{2,i}\}_{i=0}^{5} \in \{0,1\}^*, \{\epsilon_{T,0}\} \in \{0,1\}^*)$ are arbitrary distinct strings. Therefore, the three lists are initialized as $L_1 = \{(p_i, \epsilon_{1,i})\}_{i=0}^{17}, L_2 = \{(q_i, \epsilon_{2,i})\}_{i=0}^{6}, L_T = (t_0, \epsilon_{T,0})$.

At the beginning of the game, $\mathcal{S}$ sends the encoding strings $(\{\epsilon_{1,i}\}_{i=0,\cdots,17}, \{\epsilon_{2,i}\}_{i=0,\cdots,6}, \epsilon_{T,0})$ to $\mathcal{A}$, which includes $q'$+26 strings. Note that the number of encoding string $\epsilon_{1,i}$ is linear to the parameter $q'$. After this, $\mathcal{S}$ simulates the group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$ and the bilinear pairing $\hat{\text{e}}$. 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 $\mathcal{S}$ performs 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 $\epsilon_l$ 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 $\mathcal{O}_2, \mathcal{O}_T$ are treated similarly.
- $\hat{\text{e}}$: 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 $\mathcal{S}$ performs 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 $a, b, c, d, z_i, z_j, s \in \mathbb{Z}_q$ and $t_b = abc$ and $t_{1-b} = s$. $\mathcal{S}$ sets $A = a, B = b, C = c, D = d, Z_i = z_i, Z_j = z_j, T_0 = t_b, T_1 = t_{1-b}$. The simulation by $\mathcal{S}$ is perfect (and reveal 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) $p_i(a, b, c, d, z_i, z_j, t_0, t_1) = p_j(a, b, c, d, z_i, z_j, t_0, t_1)$: The polynomial $p_i \neq p_j$ due to the construction method of $L_1$, and $(p_i - p_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ is a non-zero polynomial of degree $[0, 6]$, or $q$-2 ($q$-2 is produced by $Z_j^{q-2}$). Since $Z_j \cdot Z_j^{q-2} = Z_j^{q-1} \equiv 1(\mod q)$, we have $(AC)^2Z_iZ_j \cdot Z_j^{q-2} \equiv (AC)^2Z_iZ_j(\mod q)$. By using Lemma 1 in [12], we have $\Pr[(p_i - p_j)(a, b, c, d, z_i, z_j, t_0, t_1) = 0] \le \frac{6}{q}$ because the maximum degree of $(AC)^2Z_i/Z_j(p_i - p_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ is 6. So, we have $\Pr[p_i(a, b, c, d, z_i, z_j, t_0, t_1) = p_j(a, b, c, d, z_i, z_j, t_0, t_1)] \le \frac{6}{q}$, and the `abort` probability is $\Pr[\mathsf{abort}_1] \le \frac{6}{q}$.
2) $q_i(a, b, c, d, z_i, z_j, t_0, t_1) = q_j(a, b, c, d, z_i, z_j, t_0, t_1)$: The polynomial $q_i \neq q_j$ due to the construction method of $L_2$, and $(q_i - q_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ is a non-zero polynomial of degree $[0, 4]$, 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

Yangguang Tian is with Department of Computer Science, University of Surrey, Guildford, UK

Bowen Liu, Pawel Szalachowski, and Jianying Zhou are with Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Signapore

Yingjiu Li with Computer and Information Science, University of Oregon, Eugene, US

$CDAB \cdot (AB)^{q-2} \equiv CDAB(\mod q)$. The maximum degree of $CD/AB(q_i - q_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ is 4, so the `abort` probability is $\Pr[\mathsf{abort}_2] \leq \frac{4}{q}$.

3) $t_i(a, b, c, d, z_i, z_j, t_0, t_1) = t_j(a, b, c, d, z_i, z_j, t_0, t_1)$: The polynomial $p_i \neq p_j$ due to the construction method of $L_1$, and $(p_i - p_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ is a non-zero polynomial of degree $[0, 6]$, or $q$-2. Since $(AC)^2 Z_i \cdot Z_j^{q-2}(t_i - t_j)(a, b, c, d, z_i, z_j, t_0, t_1)$ has degree 6, we have $\Pr[(p_i - p_j)(a, b, c, d, z_i, z_j, t_0, t_1) = 0] \leq \frac{6}{q}$. The `abort` probability is $\Pr[\mathsf{abort}_3] \leq \frac{6}{q}$.

By summing over all valid pairs $(i, j)$ in each case (i.e., at most $\binom{\mathcal{Q}_{\epsilon_1}+18}{2} + \binom{\mathcal{Q}_{\epsilon_2}+7}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}$ pairs), and $\mathcal{Q}_{\epsilon_1} + \mathcal{Q}_{\epsilon_2} + \mathcal{Q}_{\epsilon_T} = \mathcal{Q} + q' + 26$, we have the abort probability is

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3] \\ &\leq [\binom{\mathcal{Q}_{\epsilon_1}+18}{2} + \binom{\mathcal{Q}_{\epsilon_2}+7}{2} + \binom{\mathcal{Q}_{\epsilon_T}+1}{2}] \\ &\cdot (\frac{4}{q} + 2\frac{6}{q}) \leq \frac{16(\mathcal{Q}+q'+26)^2}{q}.\end{aligned}$$

## II. Related Works

**Dynamic Proactive Secret Sharing.** Proactive security was first introduced by Ostrovsky and Yung [8], which is refreshing secrets to withstand compromise. Later, Herzberg et al. [6] introduced proactive secret sharing (PSS). The PSS allows the distributed key shares in a SSS to be updated periodically, so that the secret remains secure even if an attacker compromises a threshold number of shareholders in each epoch. However, it did not support dynamic committees because users may join in or leave from a committee dynamically. Desmedt and Jajodia [4] introduced a scheme that redistributes secret shares to new access structure (or new committee). Specifically, a resharing technique is used to change the committee and threshold in PSS. However, the scheme is not verifiable, which disallows PSS to identify the faulty (or malicious) users. The property of verifiability is essential to PSS (i.e., verifiable secret sharing such as Feldman [5]), which holds malicious users accountable. So, the dynamic proactive secret sharing (DPSS) we considered in this work includes verifiability.

There exist several DPSS schemes in the literature. Wong et al. [14] introduced a verifiable secret redistribution protocol that supports dynamic committee. The proposed protocol allows new shareholders to verify the validity of their shares after redistribution between different committees. Zhou et al. [15] introduced an APSS, a PSS protocol for asynchronous systems that tolerate denial-of-service attacks. Schultz et al. [11] introduced a resharing protocol called MPSS. The MPSS supports mobility, which means the group of shareholders can change during resharing. Baron et al. [1] introduced a DPSS protocol that achieves a constant amortized communication overhead per secret share. In CCS'19, Maram et al. [7] presented a practical DPSS: CHURP. CHURP is designed for open blockchains, and it has very low communication overhead per epoch compared to the existing schemes [14], [15], [11], [1]. Specifically, the total number of bits transmitted between all committee members in an epoch is substantially lower than in existing schemes. Recently, Benhamouda et al. [2] introduced anonymous secret redistribution. The benefit is to ensure sharing and resharing of secrets among small dynamic committees.

DPSS can be used to secure blockchain rewriting, such as $\mu$chain [9]. $\mu$chain relies on encryption with secret sharing (ESS) to hide illegal content, as certain use-cases aim to prevent distribution of illegal content (e.g., child pornography) via the blockchain. ESS allows all the mutable transactions containing illegal content to be encrypted using transaction-specific keys. The transaction-specific keys are split into shares using DPSS [1], and these resulting shares are distributed to a number of miners, which then reshare the keys among all online miners dynamically. In this work, we use KP-ABE with DPSS to ensure blockchain rewiring with fine-grained access control. The master secret key in KP-ABE is split into key shares, and these key shares are distributed to all users in a committee. The key shares can be securely redistributed across dynamic committees. To the best of our knowledge, ours is the first attempt to distribute the master secret key in KP-ABE for decentralized systems.

## III. Security Analysis of ABET

### A. Semantic Security

Informally, an ABE scheme is secure against chosen ciphertext attacks (i.e., CCA-secure) if no group of colluding users can distinguish between encryption of $M_0$ and $M_1$ under an index $j^*$ and a set of attributes $\delta^*$ of an attacker's choice as long as no member of the group is authorized to decrypt on her own. The security experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{S}$ is given in Figure 1.

In this experiment, a challenge index $j^*$ and a challenge set of attributes $\delta^*$ are chosen by attackers at the beginning security experiment. The challenge attribute set $\delta^*$ does not satisfy any queried policy $\Lambda$ and the challenge index $j^* > i$. We define the advantage of the adversary as

$$\mathtt{Adv}_{\mathcal{A}}^{\text{Semantic}}(\lambda) = |\Pr[w = w'] - 1/2|.$$

*Definition 2:* An ABET scheme is semantically secure if for any probabilistic polynomial-time (PPT) $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}^{\text{Semantic}}(\lambda)$ is negligible in $\lambda$.

*Theorem 3:* The proposed ABET scheme is semantically secure in the standard model if the $q'$-type assumption is held in the asymmetric pairing groups.

*Proof 2:* Let $\mathcal{S}$ denote a $q'$-type problem attacker, who is given the terms from the assumption, aims to distinguish $g^{abc}$ and $g^s$. The reduction is performed as follows.

- $\mathcal{S}$ simulates master public key $\mathtt{mpk} = (g, u, v, w, h, \hat{\mathsf{e}}(g, h)^{\alpha}, \{g_1^{\alpha}, \cdots g_k^{\alpha}\}, \{h_1^{\alpha}, \cdots h_k^{\alpha}\}, g^{\beta}, h^{1/\alpha}, h^{\beta/\alpha}, \hat{\mathsf{e}}(g, h)^{\theta/\alpha})$ as follows: $\hat{\mathsf{e}}(g, h)^{\alpha} = \hat{\mathsf{e}}(g^a, h^b)$, $\{g_1^{\alpha}, \cdots g_k^{\alpha}\} = \{g^{abdz_1}, \cdots, g^{abdz_k}\}$, $\{h_1^{\alpha}, \cdots h_k^{\alpha}\} = \{h^{abdz_1}, \cdots, h^{abdz_k}\}$, $h^{1/\alpha} = h^{d/ab}$, $h^{\beta/\alpha} = h^{\beta d/ab}$, $\hat{\mathsf{e}}(g, h)^{\theta/\alpha} = \hat{\mathsf{e}}(g^{\theta}, h^{d/ab})$, and $(u, v, w)$ are simulated using the same method described in [10]. Note that $(\beta, \theta, \{z_1, \cdots, z_k\})$ are randomly chosen by $\mathcal{S}$, and $\alpha$ (or $1/\alpha$) is implicitly assigned as $ab$ (or $d/ab$) from the $q'$-type assumption. $\mathcal{A}$ submits a challenge index $j^* = \{I_1, \cdots, I_j\}$ and a set of attributes $\delta^*$ to $\mathcal{S}$.

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{Semantic}}(\lambda)$
$(\mathtt{msk}, \mathtt{mpk}) \leftarrow \mathsf{Setup}(1^\lambda), w \leftarrow \{0,1\}, \mathcal{Q} \leftarrow \emptyset$
$(M_0, M_1, \delta^*, j^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathtt{mpk})$
where $\mathcal{O} \leftarrow \{\mathsf{KeyGen}', \mathsf{Dec}'\}$
and $\mathsf{KeyGen}'(\mathtt{msk}, \cdots)$ on input $\Lambda, i$ :
return $\perp$, if $1 = \Lambda(\delta^*) \wedge i = j^*$
$\mathtt{sk}_{\Lambda_i} \leftarrow \mathsf{KeyGen}(\mathtt{msk}, \Lambda)$
return $\mathtt{sk}_{\Lambda_i}$
and $\mathsf{Dec}'(\cdots)$ on input $C$ :
return $\perp$, if $\mathtt{sk}_{\Lambda_i} \notin \mathcal{Q}$ for some $\mathtt{sk}_{\Lambda_i} \vee C = C^*$
$M \leftarrow \mathsf{Dec}(\mathtt{mpk}, C, \mathtt{sk}_{\Lambda_i})$
return $M$
$w' \leftarrow \mathcal{A}^{\mathcal{O}}(C^*)$, where $C^* \leftarrow \mathsf{Enc}(\mathtt{mpk}, M_w, \delta^*, j^*)$
return 1, if $w' = w$; else, return 0.

Fig. 1: Semantic Security.

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{Ano}}(\lambda)$
$(\mathtt{msk}, \mathtt{mpk}) \leftarrow \mathsf{Setup}(1^\lambda), w \leftarrow \{0,1\}, \mathcal{Q} \leftarrow \emptyset$
$(M, \delta^*, j^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathtt{mpk})$
where $\mathcal{O} \leftarrow \{\mathsf{KeyGen}', \mathsf{Dec}'\}$
$w' \leftarrow \mathcal{A}^{\mathcal{O}}(C^*)$, where $C^* \leftarrow \mathsf{Enc}(\mathtt{mpk}, M, \delta^*, j^* + w)$
return 1, if $w' = w$; else, return 0.

Fig. 2: Ciphertext Anonymity.

- $\mathcal{S}$ simulates decryption keys $\mathtt{sk}_{\Lambda_i} = (\{\mathtt{sk}_\tau\}_{\tau\in[n_1]}, \mathtt{sk}_0, \mathtt{sk}_1, \mathtt{sk}_2)$ (note that $\Lambda_i(\delta^*) \neq 1$) as follows: $\mathtt{sk}_0 = (g^{dt/ab}, g^r)$, $\mathtt{sk}_1 = g^\theta \cdot \widehat{i}^t \cdot g^{\beta\cdot r}$, $\mathtt{sk}_2 = \{g_{i-1}^{abdt}, \cdots, g_1^{abdt}\}$, where $\widehat{i} = g_k^{abdI_1} \cdots g_i^{abdI_i} \cdot g$. Note that $\{\mathtt{sk}_\tau\}_{\tau\in[n_1]}$ is simulated using the same method described in [10], and $(t, r)$ are randomly chosen by $\mathcal{S}$.
- $\mathcal{S}$ simulates challenge ciphertext $C^* = (ct, \{ct_{\tau,1}, ct_{\tau,2}\}_{\tau\in[\delta]}, ct_0, ct_1)$ as follows: $ct = M_b \oplus \mathtt{H}(T) \oplus \mathtt{H}(\hat{\mathsf{e}}(g,h)^{\theta cd/ab})$, $ct_0 = (h^c, h^{cd/ab}, h^{\beta cd/ab})$, and $ct_1 = \widehat{j^c} = h_k^{abcdI_1} \cdots h_j^{abcdI_j} \cdot h$. Note that $\{ct_{\tau,1}, ct_{\tau,2}\}_{\tau\in[\delta]}$ are simulated using the same method described in [10], $s$ is implicitly assigned as $c$ from the $q'$-type assumption, and $T$ can be either $\hat{\mathsf{e}}(g,h)^{abc}$ or $\hat{\mathsf{e}}(g,h)^s$.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, $\mathcal{S}$ can break the $q'$-type problem.

### B. Ciphertext Anonymity

Informally, ciphertext anonymity requires that any third party cannot distinguish the encryption of a chosen message for a first chosen index from the encryption of the same message for a second chosen index without a decryption key. We prove the ABET scheme has selective ciphertext anonymity (i.e., the index is chosen prior to the security experiment). The security experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{S}$ is given in Figure 2.

In this experiment, the decryption key $\mathtt{sk}_{\Lambda_i}$ for decrypting $C^*$ successfully is unknown to $\mathcal{A}$, which implies $1 = \Lambda_i(\delta^*)$, where $i \in [1, q]$ and $q$ denotes the number of queries to KeyGen' oracle. The restriction is that none of the KeyGen' queried pairs $((\Lambda_1, 1), \cdots, (\Lambda_q, q))$ satisfies $1 = \Lambda_i(\delta^*) \wedge i = j^*$. We define the advantage of the adversary as

$$\mathtt{Adv}_{\mathcal{A}}^{\text{Ano}}(\lambda) = |\Pr[w = w'] - 1/2|.$$

*Definition 4:* An ABET scheme is anonymous if for any PPT $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}^{\text{Ano}}(\lambda)$ is negligible in $\lambda$.

*Theorem 5:* The proposed ABET scheme is anonymous if the Extended Decisional Diffie-Hellman Assumption (eDDH) assumption is held in the asymmetric pairing groups.

*Proof 3:* Let $\mathcal{S}$ denote an eDDH problem distinguisher, who is given $(g, h, g^a, g^b, g^{ab}, h^c, h^{ab}, h^{1/ab}, h^{abc})$, aims to distinguish $h^{c/ab}$ and $h^s$ [13]. The reduction is performed as follows.

- $\mathcal{S}$ simulates master public key $\mathtt{mpk} = (g, u, v, w, h, \hat{\mathsf{e}}(g,h)^\alpha, \{g_1^\alpha, \cdots g_k^\alpha\}, \{h_1^\alpha, \cdots h_k^\alpha\}, g^\beta, h^{1/\alpha}, h^{\beta/\alpha}, \hat{\mathsf{e}}(g,h)^{\theta/\alpha})$ as follows: $\hat{\mathsf{e}}(g,h)^\alpha = \hat{\mathsf{e}}(g,h)^{ab}$, $\{g_1^\alpha, \cdots, g_k^\alpha\} = \{g_1^{ab}, \cdots, g_k^{ab}\}$, $\{h_1^\alpha, \cdots, h_k^\alpha\} = \{h_1^{ab}, \cdots, h_k^{ab}\}$, $h^{1/\alpha} = h^{1/ab}$, $h^{\beta/\alpha} = h^{\beta/ab}$, $\hat{\mathsf{e}}(g,h)^{\theta/\alpha} = \hat{\mathsf{e}}(g,h)^{\theta/ab}$. Note that $\mathcal{S}$ randomly chooses $(u, v, w)$ and $(\beta, \theta, \{z_i\})$, and implicitly sets $\alpha = ab$. $\mathcal{A}$ submits a challenge index $j^* = \{I_1, \cdots, I_j\}$ to $\mathcal{S}$.
- $\mathcal{S}$ simulates a decryption key $\mathtt{sk}_{\Lambda_i} = (\{\mathtt{sk}_\tau\}_{\tau\in[n_1]}, \mathtt{sk}_0, \mathtt{sk}_1, \mathtt{sk}_2)$ with respect to index $i = \{I_1, \cdots, I_i\}$ as follows: $\mathtt{sk}_0 = (g^t, g^{\beta\cdot r} \cdot g^{-\Sigma(z_i I_i)t\cdot b})$, $\mathtt{sk}_1 = g^\theta \cdot g^{a\cdot r}$, where $(t, r) \in \mathbb{Z}_q$ are randomly chosen by $\mathcal{S}$. The components $(\{\mathtt{sk}_\tau\}_{\tau\in[n_1]}, \mathtt{sk}_2)$ are honestly simulated by $\mathcal{S}$. The simulated components $(g^t, g^{\beta\cdot r} \cdot g^{-\Sigma(z_i I_i)t\cdot b}, g^\theta \cdot g^{a\cdot r})$ are correctly distributed, because $g^\theta \cdot g^{a\cdot r} = g^\theta \cdot \widehat{i}^t \cdot g^{a[\beta\cdot r - \Sigma(z_i I_i)t\cdot b]} = g^\theta \cdot \widehat{i}^t \cdot g^{a\cdot\overline{r}}$, where $\overline{r} = \beta \cdot r - \Sigma(z_i I_i)t \cdot b$, and $\widehat{i} = g_k^{abI_1} \cdots g_i^{abI_i} \cdot g$. So, the simulated components $(g^t, g^{\overline{r}}, g^\theta \cdot \widehat{i}^t \cdot g^{a\cdot\overline{r}})$ match the real distribution.
- $\mathcal{S}$ simulates the challenge ciphertext $C^* = (ct, \{ct_{\tau,1}, ct_{\tau,2}\}_{\tau\in[\delta]}, ct_0, ct_1)$ with respect to index $j^*$ as follows: $ct = M \oplus \mathtt{H}(\hat{\mathsf{e}}(g,h)^{abc}) \oplus \mathtt{H}(\hat{\mathsf{e}}(g,T)^\beta)$, $ct_0 = (h^c, T, T^\beta)$, and $ct_1 = h_k^{abcI_1} \cdots h_j^{abcI_j} \cdot h^c$. Note that $\mathcal{S}$ simulates $\{ct_{\tau,1}, ct_{\tau,2}\}_{\tau\in[\delta]}$ honestly, and $T$ can be either $h^{c/ab}$ or $h^s$.

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

## IV. Security Analysis of APC$^2$H

### A. Proof of Theorem 6

*Theorem 6:* The proposed generic framework is indistinguishable if the CH scheme is indistinguishable.

*Proof 4:* The reduction is executed between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. Assume that $\mathcal{A}$ activates at most $n(\lambda)$ chameleon hashes. Let $\mathcal{S}$ denote a distinguisher against CH, who is given a chameleon public key $\mathtt{pk}^*$ and a HashOrAdapt oracle, aims to break the indistinguishability of CH. In particular, $\mathcal{S}$ is allowed to access the chameleon trapdoor $Dlog(\mathtt{pk}^*)$ [3]. $\mathcal{S}$ randomly chooses $g \in [1, n(\lambda)]$ as a guess for the index of the HashOrAdapt query. In the $g$-th query, $\mathcal{S}$'s challenger directly hashes a message $(h, \mathsf{r}) \leftarrow \mathsf{Hash}(\mathtt{pk}^*, m)$, instead of calculating the chameleon hash and randomness $(h, \mathsf{r})$ using Adapt algorithm.

$\mathcal{S}$ sets up the game for $\mathcal{A}$ by distributing a master secret key to a group of users in a committee. $\mathcal{S}$ can honestly generate secret keys for any modifier associated with an access policy $\Lambda$. If $\mathcal{A}$ submits a tuple $(m_0, m_1, \delta)$ in the $g$-th query, then $\mathcal{S}$ first obtains a chameleon hash $(h_b, \mathsf{r}_b)$ from his HashOrAdapt

oracle. Then, $\mathcal{S}$ simulates a message-signature pair $(c, \sigma)$, and a ciphertext $C$ on message $Dlog(\mathtt{pk}^*)$ according to the protocol specification. Eventually, $\mathcal{S}$ returns $(h_b, \mathsf{r}_b, C, c, \sigma)$ to $\mathcal{A}$. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, $\mathcal{S}$ can break the indistinguishability of CH.

### B. Proof of Theorem 7

*Theorem 7:* The proposed generic framework is adaptively collision-resistant if the ABET scheme is semantically secure, the CH scheme is collision-resistant, and the DPSS scheme has secrecy.

*Proof 5:* We define a sequence of games $\mathbb{G}_i$, $i = 0, \cdots, 4$ and let $\mathtt{Adv}_i^{GF}$ denote the advantage of the adversary in game $\mathbb{G}_i$. Assume that $\mathcal{A}$ issues at most $n(\lambda)$ queries to Hash oracle.

- $\mathbb{G}_0$: This is the original game for adaptive collision-resistance.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ will output a random bit if $\mathcal{A}$ outputs a correct master secret key when no more than $t$ users in a committee are corrupted, and the setup is honest (or the dealer is honest). The difference between $\mathbb{G}_0$ and $\mathbb{G}_1$ is negligible if DPSS scheme has secrecy.

$$\left|\mathtt{Adv}_0^{GF} - \mathtt{Adv}_1^{GF}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{DPSS}}(\lambda). \tag{1}$$

- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except the following difference: $\mathcal{S}$ randomly chooses $g \in [1, n(\lambda)]$ as a guess for the index of the Hash$'$ oracle which returns the chameleon hash $(h^*, m^*, \mathsf{r}^*, C^*, c^*, \sigma^*)$. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s attacking query does not occur in the $g$-th query. Therefore, we have

$$\mathtt{Adv}_1^{GF} = n(\lambda) \cdot \mathtt{Adv}_2^{GF} \tag{2}$$

- $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that in the $g$-th query, the encrypted message $\mathtt{sk}_{\mathsf{CH}}$ in $C^*$ is replaced by "$\perp$" (i.e., an empty value). Below we show that the difference between $\mathbb{G}_2$ and $\mathbb{G}_3$ is negligible if ABET scheme is semantically secure.

  Let $\mathcal{S}$ denote an attacker against ABET with semantic security, who is given a public key $\mathtt{pk}^*$ and a key generation oracle and a decryption oracle, aims to distinguish between encryptions of $M_0$ and $M_1$ associated with a challenge index $j^*$ and a challenge set of attributes $\delta^*$, which are predetermined at the beginning of the game for semantic security. $\mathcal{S}$ simulates the game for $\mathcal{A}$ as follows.
  - $\mathcal{S}$ sets up $\mathtt{mpk}_{\mathsf{ABET}} = \mathtt{pk}^*$ and completes the remainder of Setup honestly, which includes user's key pairs and chameleon key pairs for hashing in CH. $\mathcal{S}$ returns all public information to $\mathcal{A}$.
  - $\mathcal{S}$ can honestly answer the key generation and decryption queries made by $\mathcal{A}$ using his given oracles. The restrictions on those queries are $\Lambda_i(\delta^*) \neq 1 \wedge i \neq j^*$ and $C \neq C^*$, respectively. In the $g$-th query, upon receiving a hash query w.r.t., a hashed message $m^*$ from $\mathcal{A}$. $\mathcal{S}$ first submits two messages (i.e., $[M_0 = \mathtt{sk}_{\mathsf{CH}}, M_1 = \perp]$) to his challenger, and obtains a challenge ciphertext $C^*$ under index $j^*$ and $\delta^*$. Then, $\mathcal{S}$ returns the tuple $(h^*, m^*, \mathsf{r}^*, C^*, c^*, \sigma^*)$ to $\mathcal{A}$. Note that $\mathcal{S}$ can simulate the message-signature pair $(c^*, \sigma^*)$ honestly using user's key pairs. Besides, $\mathcal{S}$ can simulate the adapt query successfully using $\mathtt{sk}_{\mathsf{CH}}$.

  If the encrypted message in $C^*$ is $\mathtt{sk}_{\mathsf{CH}}$, then the simulation is consistent with $\mathbb{G}_2$; Otherwise, the simulation is consistent with $\mathbb{G}_3$. Therefore, if the advantage of $\mathcal{A}$ is significantly different in $\mathbb{G}_2$ and $\mathbb{G}_3$, $\mathcal{S}$ can break the semantic security of the ABET. Hence, we have

$$\left|\mathtt{Adv}_2^{GF} - \mathtt{Adv}_3^{GF}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda). \tag{3}$$

- $\mathbb{G}_4$: This game is identical to game $\mathbb{G}_3$ except that in the $g$-th query, $\mathcal{S}$ outputs a random bit if $\mathcal{A}$ outputs a valid collision $(h^*, m^{*'}, \mathsf{r}^{*'}, C^{*'}, c^{*'}, \sigma^{*'})$, and it was not previously returned by the Adapt$'$ oracle. Below we show that the difference between $\mathbb{G}_3$ and $\mathbb{G}_4$ is negligible if CH is collision-resistant.

  Let $\mathcal{S}$ denote an attacker against CH with collision-resistant, who is given a chameleon public key $\mathtt{pk}^*$ 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 $\mathtt{pk}_{\mathsf{CH}} = \mathtt{pk}^*$ for the $g$-th hash query, and completes the remainder of Setup honestly, which includes user's key pairs and master key pair in ABET. $\mathcal{S}$ returns all public information to $\mathcal{A}$.
  - $\mathcal{S}$ can simulate all queries made by $\mathcal{A}$ except adapt queries. If $\mathcal{A}$ submits an adapt query in the form of $(h, m, \mathsf{r}, C, c, \sigma, m')$, then $\mathcal{S}$ obtains a randomness $\mathsf{r}'$ from his Adapt$'$ oracle, and returns $(h, m', \mathsf{r}', C', c', \sigma')$ to $\mathcal{A}$. In particular, $\mathcal{S}$ simulates the $g$-th hash query as $(h^*, m^*, \mathsf{r}^*, C^*, c^*, \sigma^*)$ w.r.t. a hashed message $m^*$, where $C^* \leftarrow \mathsf{Enc}_{\mathsf{ABET}}(\mathtt{mpk}^*, \perp, \delta^*, j^*)$, $c^*$ is derived from $\perp$ because $Dlog(\mathtt{pk}^*)$ is unknown.
  - If $\mathcal{A}$ outputs a collision $(h^*, m^*, \mathsf{r}^*, C^*, c^*, \sigma^*, m^{*'}, \mathsf{r}^{*'}, C^{*'}, c^{*'}, \sigma^{*'})$ with respect to the $g$-th query, and all relevant checks are succeed, then $\mathcal{S}$ output $(h^*, m^{*'}, \mathsf{r}^{*'}, C^{*'}, c^{*'}, \sigma^{*'})$ as a valid collision to CH; Otherwise, $\mathcal{S}$ aborts the game. Therefore, we have

$$\left|\mathtt{Adv}_3^{GF} - \mathtt{Adv}_4^{GF}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda). \tag{4}$$

Combining the above results together, we have

$$\begin{aligned}\mathtt{Adv}_{\mathcal{A}}^{GF}(\lambda) \quad \leq \quad & n(\lambda) \cdot (\mathtt{Adv}_{\mathcal{S}}^{\mathsf{DPSS}}(\lambda) + \mathtt{Adv}_{\mathcal{S}}^{\mathsf{ABET}}(\lambda) \\ & + \mathtt{Adv}_{\mathcal{S}}^{\mathsf{CH}}(\lambda)).\end{aligned}$$

### C. Proof of Theorem 8

*Theorem 8:* The proposed generic framework is accountable if the $\Sigma$ scheme is EUF-CMA secure, and the DPSS scheme has correctness.

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

- $\mathbb{G}_0$: This is the original game for accountability.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ will output a random bit if all honest users in a committee outputs a master secret key $\mathtt{msk}'$ such that $\mathtt{msk}' \neq \mathtt{msk}$,

and the setup is honest. The difference between $\mathbb{G}_0$ and $\mathbb{G}_1$ is negligible if DPSS scheme has correctness.

$$\left|\mathtt{Adv}_0^{GF} - \mathtt{Adv}_1^{GF}\right| \leq \mathtt{Adv}_{\mathcal{S}}^{\mathsf{DPSS}}(\lambda). \tag{5}$$

- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ will output a random bit if $\mathcal{A}$ outputs a valid forgery $\sigma^*$, where $\sigma^*$ was not previously simulated by $\mathcal{S}$ and the user is honest. The difference between $\mathbb{G}_1$ and $\mathbb{G}_2$ is negligible if $\Sigma$ is EUF-CMA secure.

  Let $\mathcal{F}$ denote a forger against $\Sigma$, who is given a public key $\mathtt{pk}^*$ and a signing oracle $\mathcal{O}^{\mathsf{Sign}}$, aims to break the EUF-CMA security of $\Sigma$. Assume that $\mathcal{A}$ activates at most $n$ users in the system.
  - $\mathcal{F}$ randomly chooses a user in the system and sets up its public key as $\mathtt{pk}^*$. $\mathcal{F}$ completes the remainder of Setup honestly. Below we mainly focus on user $\mathtt{pk}^*$ only.
  - To simulate a chameleon hash for message $m$, $\mathcal{F}$ first obtains a signature $\sigma$ from his signing oracle $\mathcal{O}^{\mathsf{Sign}}$. Then, $\mathcal{F}$ generates chameleon hash and ciphertext honestly because $\mathcal{F}$ chooses the chameleon secret key $\mathtt{sk}_{\mathsf{CH}}$, and returns $(m, h, \mathsf{r}, C, c, \sigma)$ to $\mathcal{A}$. Besides, the message-signature pairs and collisions can be perfectly simulated by $\mathcal{F}$ for any adapt query. $\mathcal{F}$ records all the simulated message-signature pairs by including them to a set $\mathcal{Q}$.
  - When forging attack occurs, i.e., $\mathcal{A}$ outputs $(m^*, h^*, \mathsf{r}^*, C^*, c^*, \sigma^*)$, $\mathcal{F}$ checks whether:
    * the forging attack happens to user $\mathtt{pk}^*$;
    * the ciphertext $C^*$ encrypts the chameleon trapdoor $\mathtt{sk}_{\mathsf{CH}}$;
    * the message-signature pair $(c^*, \sigma^*)$ is derived from the chameleon trapdoor $\mathtt{sk}_{\mathsf{CH}}$;
    * the message-signature pair $(c^*, \sigma^*) \notin \mathcal{Q}$;
    * $1 \leftarrow \Sigma.\mathsf{Verify}(\mathtt{pk}^*, c^*, \sigma^*)$ and $1 \leftarrow \mathsf{Verify}(\mathtt{pk}_{\mathsf{CH}}, m^*, h^*, \mathsf{r}^*)$.

  If all the above conditions hold, $\mathcal{F}$ confirms that it as a successful forgery from $\mathcal{A}$, then $\mathcal{F}$ extracts the forgery via $\sigma \leftarrow M_\Sigma(\mathsf{PP}, \mathtt{pk}^*, \sigma^*, \Delta(\mathtt{sk}))$ due to the homomorphic property of $\Sigma$ (regarding keys and signatures [13]), where $\Delta(\mathtt{sk})$ is derived from $(c, c^*)$. To this end, $\mathcal{F}$ outputs $\sigma$ as its own forgery; Otherwise, $\mathcal{F}$ aborts the game.

$$\left|\mathtt{Adv}_1^{GF} - \mathtt{Adv}_2^{GF}\right| \leq n \cdot \mathtt{Adv}_{\mathcal{F}}^{\Sigma}(\lambda). \tag{6}$$

Combining the above results together, we have

$$\mathtt{Adv}_{\mathcal{A}}^{GF}(\lambda) \quad \leq \quad \mathtt{Adv}_{\mathcal{S}}^{\mathsf{DPSS}}(\lambda) + n \cdot \mathtt{Adv}_{\mathcal{F}}^{\Sigma}(\lambda).$$

## REFERENCES


[1] J. Baron, K. El Defrawy, J. Lampkins, and R. Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. In *ACNS*, pages 23–41, 2015.

[2] F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In *TCC*, pages 260–290, 2020.

[3] D. Derler, K. Samelin, D. Slamanig, and C. Striecks. Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based. In *NDSS*, 2019.

[4] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical report, 1997.

[5] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS*, pages 427–438, 1987.

[6] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO*, pages 339–352, 1995.

[7] S. K. D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song. Churp: Dynamic-committee proactive secret sharing. In *CCS*, pages 2369–2386, 2019.

[8] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *ACM PODC*, pages 51–59, 1991.

[9] I. Puddu, A. Dmitrienko, and S. Capkun. $\mu$chain: How to forget without hard forks. *IACR Cryptology ePrint Archive*, 2017:106, 2017.

[10] Y. Rouselakis and B. Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *CCS*, pages 463–474, 2013.

[11] D. A. Schultz, B. Liskov, and M. Liskov. Mobile proactive secret sharing. In *ACM PODC*, pages 458–458, 2008.

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

[13] Y. Tian, N. Li, Y. Li, P. Szalachowski, and J. Zhou. Policy-based chameleon hash for blockchain rewriting with black-box accountability. In *ACSAC*, pages 813–828, 2020.

[14] T. M. Wong, C. Wang, and J. M. Wing. Verifiable secret redistribution for archive systems. In *First International IEEE Security in Storage Workshop, 2002. Proceedings.*, pages 94–105, 2002.

[15] L. Zhou, F. B. Schneider, and R. Van Renesse. Apss: Proactive secret sharing in asynchronous systems. *ACM (TISSEC)*, 8(3):259–286, 2005.