Cryptography of Hyperledger Indy

Kyle Huang

February 5, 2020

1 Syntax of Hyperledger Indy

The first four steps are similar to the register operation, and the last four steps look like login.[1]

1. Issuer determines a credential schema $\mathcal{S}$: the type of cryptographic signatures used to sign the credentials, the number l of attributes in a credential, the indices $\mathcal{A}_h \subset [1, l] = \{1, 2, ..., l\}$ of hidden attributes, the public key P_k, the non-revocation credential attribute number l_r and non-revocation public key P_r. Then he publishes it on the ledger and announces the attribute semantics.
2. Holder retrieves the credential schema from the ledger and sets the hidden attributes.
3. Holder requests a credential from issuer. He sends hidden attributes in a blinded form to issuer and agrees on the values of known attributes $\mathcal{A}_k \leftarrow [1, l] \backslash \mathcal{A}_h$.
4. Issuer returns a credential pair (C_p, C_{NR}) to holder. The first credential contains the requested l attributes. The second credential asserts the non-revocation status of the first one. Issuer publishes the non-revoked status of the credential on the ledger.
5. Holder approaches verifier. Verifier sends the Proof Request $\mathcal{E}$ to holder. The Proof Request contains the credential schema $\mathcal{S}_E$ and disclosure predicates $\mathcal{D}$. The predicates for attribute m and value V can be of form $m = V$, $m < V$, or $m > V$. Some attributes may be asserted to be the same: $m_i = m_j$.
6. Holder checks that the credential pair he holds satisfies the schema $\mathcal{S}_E$. He retrieves the non-revocation witness from the ledger.
7. Holder creates a proof $\mathcal{P}$ that he has a non-revoked credential satisfying the proof request $\mathcal{E}$ and sends it to verifier.
8. Verifier verifies the proof.

[1] All content refers to Hyperledger Indy HIPE.

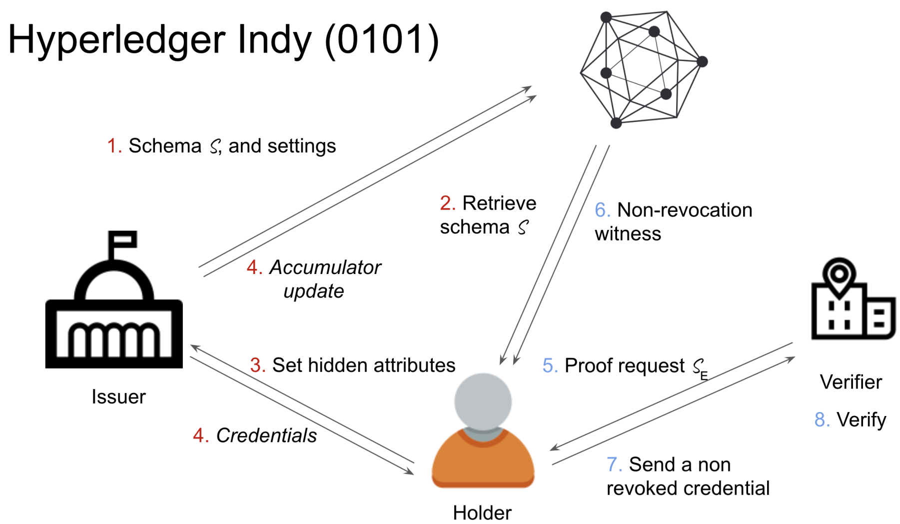

Figure 1: Syntax of Hyperledger Indy

Symbol	Definition
$\mathcal{S}$	Schema, the empty data form with only fields.
(l, l_r)	**Attributes** number and **non-revocation** credential attribute number.
L	The volume of a non-revocation list.
l_a	Message length for all attributes. In Sovrin, $l_a = 256$.
$(\mathcal{A}_k, \mathcal{A}_h)$	The indices of **known attributes** and **hidden attributes** respectively.
	By default, $\{1, 3\} \subset \mathcal{A}_h$ and $\{2\} \subset \mathcal{A}_k$.
(P_k, P_r)	Public keys of **primary credentials** and **non-revocation credentials** resp.
$\mathcal{P}_1$	Correctness proof of P_k.
$(i, \mathcal{H})$	The **index** and **identifier** of a holder in the issuer's view.
(V, acc_V)	The **indices** and **accumulator** of the current non-revocation list.
(C_P, C_{NR})	The **primary credential** and the **non-revocation credential**.

Table 1: Symbol table

2 Practical construction

2.1 Overview

1. $(sk_I, pk_I, state_V, epoch_V) \leftarrow \texttt{setup}(l, L)$
2. $\texttt{obtainCert}(\mathcal{U}(pk_I, \mathcal{A}_h, \{m_j\}_{\forall j \in \mathcal{A}_h}), \mathcal{I}(sk_I, \mathcal{A}_k, \{m_j\}_{\forall j \in \mathcal{A}_k}, {state_V}^{old}, {epoch_V}^{old}, i))$
   (a) Update $epoch_{V \cup \{i\}}$ on the ledger.
   (b) Holder $\mathcal{U}$ acqires credentials (C_P, C_{NR}, wit_i) where $C_P \leftarrow \texttt{sign}(sk_I, \{m_j\}_{\forall j \in \mathcal{A}_h \cup \mathcal{A}_k})$ and $C_{NR} \leftarrow \texttt{sign}(sk_I, (V \cup \{i\}))$.
3. $epoch_V \leftarrow \texttt{updateEpoch}()$ # By verifier. In our case, $epoch_V$ is on the ledger so everyone can check.
4. $wit_i \leftarrow \texttt{updateWitness}(\mathcal{U}({wit_i}^{old}), \mathcal{I}(state_V))$
5. $\texttt{True/False} \leftarrow \texttt{verify}(\mathcal{U}(C_P, C_{NR}, wit_i), \mathcal{V}(epoch_V))$

Algorithm 1 $\texttt{setup}(l, L)$

$\triangleright$ For primary credential
$p', q' \leftarrow_R \{0,1\}^{1536}$ $\triangleright$ p' and q' are prime; $|p'| = |q'| = 1536$
$p \leftarrow 2p' + 1$; $q \leftarrow 2q' + 1$; $n \leftarrow pq$ $\triangleright$ p and q are prime
$t \leftarrow_R \mathbb{Z}_n^*$; $S \leftarrow t^2 \pmod n$
$x_z \leftarrow_R \mathbb{Z}_{p'q'}^*$, $Z \leftarrow S^{x_z} \pmod n$
$\{x_{r_i} \leftarrow_R \mathbb{Z}_{p'q'}^*, R_i \leftarrow S^{x_{r_i}} \pmod n\}_{\forall i \in [1,l]}$
$P_k \leftarrow (n, S, Z, \{R_i\}_{\forall i \in [1,l]})$
$\triangleright$ The proof of correctness for P_k
$\tilde{x}_z \leftarrow_R \mathbb{Z}_{p'q'}^*$, $\tilde{Z} \leftarrow S^{\tilde{x}_z} \pmod n$
$\{\tilde{x}_{r_i} \leftarrow_R \mathbb{Z}_{p'q'}^*, \tilde{R}_i \leftarrow S^{\tilde{x}_{r_i}} \pmod n\}_{\forall i \in [1,l]}$
$c \leftarrow H_1(Z || \tilde{Z} || \{R_i, \tilde{R}_i\}_{\forall i \in [1,l]})$ $\triangleright$ H_1 is by default SHA2-256
$\hat{x}_z \leftarrow \tilde{x}_z + c \cdot x_z$; $\{\hat{x}_{r_i} \leftarrow \tilde{x}_{r_i} + c \cdot x_{r_i}\}_{\forall i \in [1,l]}$
$\mathcal{P}_1 \leftarrow (c, \hat{x}_z, \{\hat{x}_{r_i}\}_{\forall i \in [1,l]})$
$\triangleright$ For non-revocation credential
$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ $\triangleright$ pick a type-III pairing where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$
$g \leftarrow_R \mathbb{G}_1$; $g' \leftarrow_R \mathbb{G}_2$
$h, h_0, h_1, h_2, \tilde{h} \leftarrow_R \mathbb{G}_1$; $u, \hat{h} \leftarrow_R \mathbb{G}_2$
$sk, x, r \leftarrow_R \mathbb{Z}_q^*$; $pk \leftarrow g^{sk}$; $y \leftarrow \hat{h}^x$
$\triangleright$ accumulator settings
$z \leftarrow e(g, g')^{r^{L+1}}$; $V \leftarrow \emptyset$; $acc_V \leftarrow 1$
$state_V \leftarrow (V, \{g_i, g'_i\}_{\forall i \in [1,2L] \setminus \{L+1\}})$; $epoch_V \leftarrow (V, acc_V)$
$P_r \leftarrow (h, h_0, h_1, h_2, \tilde{h}, pk, \hat{h}, u, y, z)$
$sk_I \leftarrow (p, q, sk, x, r)$, $pk_I \leftarrow (P_k, \mathcal{P}_1, P_r)$
return $(sk_I, pk_I, state_V, epoch_V)$

Algorithm 2 $\texttt{verify}_{P_k}(l, P_k, \mathcal{P}_1)$

$(n, S, Z, \{R_i\}_{\forall i \in [1,l]}) \leftarrow P_k$; $(c, \hat{x}_z, \{\hat{x}_{r_i}\}_{\forall i \in [1,l]}) \leftarrow \mathcal{P}_1$
$\tilde{Z} \leftarrow Z^{-c} S^{\hat{x}_z}$; $\{\tilde{R}_i \leftarrow R_i^{-c} S^{\hat{x}_{r_i}}\}_{\forall i \in [1,l]} \pmod n$
return $c == H_1(Z || \tilde{Z} || \{R_i, \tilde{R}_i\}_{\forall i \in [1,l]})$

2.2 Setup

Issuer generates the key pair (sk_I, pk_I), $state_V$ and $epoch_V$ through $\texttt{setup}$ (Algorithm 1); then, he keeps $(sk_I, state_V)$ in a secret manner and publishes $(\mathcal{S}, \mathcal{A}_h, l_r, pk_I, epoch_V)$ to the ledger. Let $(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$, everyone can verify the correctness of P_k through $\texttt{verify}_{P_k}(l, P_k, \mathcal{P}_1)$ (Algorithm 2).

$$(sk_I, pk_I, state_V, epoch_V) \leftarrow \texttt{setup}(l, L) \tag{1}$$

2.3 Credential Issuance

$$(C_P, C_{NR}, state_V, epoch_V, wit_i) \leftarrow \texttt{ObtainCert}(\mathcal{U}(pk_I, \mathcal{A}_h, \{m_j\}_{\forall j \in \mathcal{A}_h}), \mathcal{I}(sk_I, \mathcal{A}_k, \{m_j\}_{\forall j \in \mathcal{A}_k}, state_V{}^{old}, epoch_V{}^{old}, i))$$

Let $i < L$ and $\mathcal{H}$ be the index and identifier of the holder in the issuer's system, respectively. The holder acquires the schema $\mathcal{S}$, indices $\mathcal{A}_h$ and public keys pk_I from the ledger in addition to a random number n_0 and the identifier $\mathcal{H}$ from the issuer; then he sets the hidden attribute $\{m_i\}_{\forall i \in \mathcal{A}_h}$. The credential issuance process is interactive, which follows:

1. The holder requests a credential query req by excuting Algorithm 3, $\texttt{credential}_{req}$; then, he keeps (v', s') private and sends req to the issuer.

$$(req, (v', s')) \leftarrow \texttt{credential}_{req}(\mathcal{S}, \mathcal{A}_h, \{m_i\}_{\forall i \in \mathcal{A}_h}, n_0, \mathcal{H}, pk_I) \tag{2}$$

2. On receiving req from the holder, the issuer firstly verifies req through $\texttt{verify}_{req}(pk_I, req)$ (Algorithm 4). If it passes, the issuer runs Algorithm 5, $\texttt{credential}_{res}(i, state_V{}^{old}, \mathcal{H}, \{m_i\}_{\forall i \in \mathcal{A}_k}, sk_I, req)$, to generate parameters $(res, state_V, epoch_V)$. Finally, the issuer stores the holder's information and index i in issue's local database, updates its own $state_V$; then, he updates $epoch_V$ on the ledger and returns res to the holder.

$$(res, state_V, epoch_V) \leftarrow \texttt{credential}_{res}(i, state_V{}^{old}, \mathcal{H}, \{m_i\}_{\forall i \in \mathcal{A}_k}, sk_I, req) \tag{3}$$

3. While receiving response res from the issuer, the holder excutes Algorithm 6 $\texttt{credential}_{finish}(pk_I, (v', s'), req, res)$ to do some verifications. If all verifications pass, the holder keeps returned credential (C_P, C_{NR}) and witness wit_i.

$$(C_P, C_{NR}) \leftarrow \texttt{credential}_{finish}(pk_I, (v', s'), req, res) \tag{4}$$

Algorithm 3 $\texttt{credential}_{req}(\mathcal{S}, \mathcal{A}_h, \{m_i\}_{\forall i \in \mathcal{A}_h}, n_0, \mathcal{H}, pk_I)$

$\triangleright$ primary credential

$\{\tilde{m}_i \leftarrow_R \{0,1\}^{593}\}_{\forall i \in \mathcal{A}_h}$; $(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$
$v' \leftarrow_R \{0,1\}^{3152}$; $\tilde{v}' \leftarrow_R \{0,1\}^{3488}$
$(n, S, Z, \{R_i\}_{\forall i \in [1,l]}) \leftarrow P_k$
$U \leftarrow S^{v'} \prod_{\forall i \in \mathcal{A}_h} R_i^{m_i}$; $\tilde{U} \leftarrow S^{\tilde{v}'} \prod_{\forall i \in \mathcal{A}_h} R_i^{\tilde{m}_i}$
$c = H(U||\tilde{U}||n_0)$; $n_1 \leftarrow_R \{0,1\}^{80}$
$\hat{v} \leftarrow \tilde{v} + c \cdot v$; $\{\hat{m}_i \leftarrow \tilde{m}_i + c \cdot m_i\}_{\forall i \in \mathcal{A}_h}$

$\triangleright$ non-revocation credential

$(h, h_0, h_1, h_2, \tilde{h}, pk, \hat{h}, u, y, z) \leftarrow P_r$
$s' \leftarrow_R \mathbb{Z}_q^*$, $U_r \leftarrow h_2^{s'}$
$req \leftarrow (U, c, \hat{v}', \{m_i\}_{\forall i \in \mathcal{A}_h}, n_1, U_r)$
return $(req, (v', s'))$

Algorithm 4 $\texttt{verify}_{req}(pk_I, req)$

$(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$
$(n, S, Z, \{R_i\}_{\forall i \in [1,l]}) \leftarrow P_k$; $(U, c, \hat{v}', \{m_i\}_{\forall i \in \mathcal{A}_h}, n_1, U_r) \leftarrow req$
$\tilde{U} \leftarrow U^{-c} S^{\hat{v}'} \prod_{\forall i \in \mathcal{A}_h} S^{\hat{m}_i} R_i^{-c} \pmod n$
return $c == H(U||\tilde{U}||n_0)$

Algorithm 5 $\texttt{credential}_{res}(i, state_V{}^{old}, \mathcal{H}, \{m_i\}_{\forall i \in \mathcal{A}_k}, sk_I, req)$

$\triangleright$ primary credential

$(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$; $m_2 \leftarrow H(i||\mathcal{H})$ $\triangleright$ $\mathcal{H}$ is the identifier of holder, like ID
$v'' \leftarrow \{0,1\}^{2723}$; $e'' \leftarrow \{0,1\}^{596}$ $\triangleright$ $|v''| = 2723$, $|e| = 596$ and e is prime
$(n, S, Z, \{R_i\}_{\forall i \in [1,l]}) \leftarrow P_k$; $(U, c, \hat{v}', \{m_i\}_{\forall i \in \mathcal{A}_h}, n_1, U_r) \leftarrow req$
$Q \leftarrow Z(US^{v''} \prod_{\forall i \in \mathcal{A}_k} R_i^{m_i})^{-1} \pmod n$; $r' \leftarrow_R \mathbb{Z}_{p'q'}^*$
$A \leftarrow Q^{e^{-1} \pmod{p'q'}}$; $\hat{A} \leftarrow Q^{r'} \pmod n$
$c' \leftarrow H(Q||A||\hat{A}||n_1)$; $s_e \leftarrow r' - c'e^{-1}$
$R_c \leftarrow (\{m_i\}_{\forall i \in \mathcal{A}_k}, A, e, v'', s_e, c')$

$\triangleright$ non-revocation credential

$s'', c \leftarrow_R \mathbb{Z}_q^*$; $(V^{old}, \{g_i, g_i'\}_{\forall i \in [1,2L] \backslash \{L+1\}})) \leftarrow state_V{}^{old}$
$(h, h_0, h_1, h_2, \tilde{h}, pk, \hat{h}, u, y, z) \leftarrow P_r$, $(sk, x, r) \leftarrow sk_I$
$\sigma \leftarrow (h_0 h_1^{m_2} U_r g_i h_2^{s''})^{(x+c)^{-1}}$; $\sigma_i \leftarrow g'^{(sk + r^i)^{-1}}$; $u_i \leftarrow u^{r^i}$
$w \leftarrow \prod_{\forall j \in V, j \neq i} g'_{L+1+i-j}$; $V \leftarrow V^{old} \cup \{i\}$, $acc_V \leftarrow \prod_{\forall i \in V} g'_{L+1-j}$
$wit_i \leftarrow (\sigma_i, u_i, g_i, w, V)$; $R_r \leftarrow (I_A, \sigma, c, s'', wit_i, g_i, g_i', i)$
$res \leftarrow (acc_V, \mathcal{H}, R_c, R_r)$
$state_V \leftarrow (V, \{g_i, g_i'\}_{\forall i \in [1,2L] \backslash \{L+1\}})$; $epoch_V \leftarrow (V, acc_V)$
return $(res, state_V, epoch_V)$

Algorithm 6 $\texttt{credential}_{finish}(pk_I, (v', s'), req, res)$

$(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$; $(n, S, Z, \{R_i\}_{\forall i \in [1,l]}) \leftarrow P_k$;
$(h, h_0, h_1, h_2, \tilde{h}, pk, \hat{h}, u, y, z) \leftarrow P_r$
$(U, c, \hat{v}', \{m_i\}_{\forall i \in \mathcal{A}_h}, n_1, U_r) \leftarrow req$; $(acc_V, \mathcal{H}, R_c, R_r) \leftarrow res$
$(\{m_i\}_{\forall i \in \mathcal{A}_k}, A, e, v'', s_e, c') \leftarrow R_c$; $(I_A, \sigma, c, s'', wit_i, g_i, g'_i, i) \leftarrow R_r$
$(\sigma_i, u_i, g_i, w, V) \leftarrow wit_i$; $m_2 \leftarrow H(i||\mathcal{H})$
$v \leftarrow v' + v''$; $s \leftarrow s' + s''$
$Q \leftarrow Z(S^v \prod_{\forall i \in (\mathcal{A}_k \cup \mathcal{A}_h)} R_i^{m_i})^{-1} \pmod n$
$\hat{A} \leftarrow A^{c' + s_e \cdot e}$
if $e(g_i,\, acc_V)(e(g,\, w))^{-1} \neq z$ **then**
    return null
else if $e(pk \cdot g_i,\, \sigma_i) \neq e(g,\, g')$ **then**
    return null
else if $e(\sigma,\, y \cdot \hat{h}^c) \neq e(h_0 h_1^{m_2} h_2^s \cdot g_i,\, \hat{h})$ **then**
    return null
else if e is not prime OR $e \notin [2^{596}, 2^{596} + 2^{119}]$ **then**
    return null
else if $Q \neq A^e$ **then**
    return null
else if $c' \neq H(Q||A||\hat{A}||n_1)$ **then**
    return null
else
    $C_P \leftarrow (\{m_i\}_{\forall i \in (\mathcal{A}_h \cup \mathcal{A}_k)}, A, e, v)$; $C_{NR} \leftarrow (I_A, \sigma, c, s, wit_i, g_i, g'_i, i)$
    return (C_P, C_{NR})
end if

2.4 Credential Revocation

The revocation process is quite straightforward. The issuer fetches the current ${epoch_V}^{old}$ from the ledger. Then, he revokes user with index i via Algorithm 7 and updates $epoch_V$ and $state_V$ on the ledger and in its private database, respectively, after running $(state_V, epoch_V) \leftarrow \texttt{revoke}({epoch_V}^{old}, i)$.

Algorithm 7 $\texttt{revoke}({state_V}^{old}, {epoch_V}^{old}, i)$

$(V^{old}, \{g_i, g'_i\}_{\forall i \in [1,2L] \setminus \{L+1\}}) \leftarrow {state_V}^{old}$; $(V^{old}, {acc_V}^{old}) \leftarrow {epoch_V}^{old}$
$V \leftarrow V^{old} \setminus \{i\}$; $acc_V \leftarrow {acc_V}^{old} \cdot (g'_{L+1-j})^{-1}$
$state_V \leftarrow (V, \{g_i, g'_i\}_{\forall i \in [1,2L] \setminus \{L+1\}})$; $epoch_V \leftarrow (V, acc_V)$
return $(state_V, epoch_V)$

2.5 Epoch update

The epoch update is omitted since the epoch is stored on the ledger in Hyperledger Indy so that each node synchronizes the last version of epoch.

2.6 Witness update

While a verifier touches a holder to issue a zero-knowledge proof, the first step of the holder is to update his witness by requesting ${wit_i}^{old}$ to the issuer; and the issuer computes and returns through `WitnessUpdate` (Algorithm 8).

Algorithm 8 $\texttt{updateWitness}({wit_i}^{old}, state_V)$

$(\sigma_i, u_i, g_i, w^{old}, V^{old}) \leftarrow {wit_i}^{old}$; $(V, \{g_i, g'_i\}_{\forall i \in [1,2L] \setminus \{L+1\}}) \leftarrow state_V$
$w \leftarrow w^{old} \prod_{\forall j \in V \setminus V^{old}} g'_{L+1+i-j} / \prod_{\forall j \in V^{old} \setminus V} g'_{L+1+i-j}$
$wit_i \leftarrow (\sigma_i, u_i, g_i, w, V)$
return wit_i

2.7 Credential verification

After updating the witness, the holder generates a $proof \leftarrow \texttt{proof}(C_P, C_{NR}, pk_I)$ (algorithm 9). Let $(commit, opener) \leftarrow proof$ be the generated proof, the holder sends $commit$ to the verifier first, after confirmation from the verifier, the holder sends $opener$ to the verifier to open the aforementioned $commit$; and the verifier is convinced if algorithm 10 returns $\texttt{True} \leftarrow \texttt{verify}_{credential}(proof, epoch_V)$.

Algorithm 9 $\texttt{proof}(C_P, C_{NR}, pk_I)$

$(\{m_i\}_{\forall i \in (\mathcal{A}_h \cup \mathcal{A}_k)}, A, e, v) \leftarrow C_P$; $(I_A, \sigma, c, s, wit_i, g_i, g'_i, i) \leftarrow C_{NR}$
$(\sigma_i, u_i, g_i, w, V) \leftarrow wit_i$; $(P_k, \mathcal{P}_1, P_r) \leftarrow pk_I$
$(h, h_0, h_1, h_2, \tilde{h}, pk, \hat{h}, u, y, z) \leftarrow P_r$; $\rho, r, r', r'', r''', \leftarrow_R \mathbb{Z}_q^*$
$mult \leftarrow c\rho$; $tmp \leftarrow c \cdot open$; $mult' \leftarrow r'' \cdot r$; $tmp' \leftarrow r'' \cdot open$
$C \leftarrow h^{\rho}\tilde{h}^{open}$; $D \leftarrow g^r \tilde{h}^{open'}$; $A \leftarrow \sigma\tilde{h}^{\rho}$; $\mathcal{G} \leftarrow g_i \tilde{h}^r$
$\mathcal{W} \leftarrow w g'^{r'}$; $\mathcal{S} \leftarrow \sigma_i g'^{r''}$; $\mathcal{U} \leftarrow u_i g'^{r'''}$
$commit \leftarrow (C, D, A, \mathcal{G}, \mathcal{W}, \mathcal{S}, \mathcal{U})$
$opener \leftarrow (c, \rho, \{m_j\}_{\forall j \in \mathcal{A}_k}, r, s, open, open', mult, mult', tmp, tmp', r', r'', r''')$
$proof \leftarrow (commit, opener)$
return $proof$

Algorithm 10 $\texttt{verify}_{credential}(proof, epoch_V)$

$(commit, opener) \leftarrow proof$; $(C, D, A, \mathcal{G}, \mathcal{W}, \mathcal{S}, \mathcal{U}) \leftarrow commit$
$(c, \rho, \{m_j\}_{\forall j \in \mathcal{A}_k}, r, s, open, open', mult, mult', tmp, tmp', r', r'', r''') \leftarrow opener$
$(V, acc_V) \leftarrow epoch_V$
$X_1 \leftarrow e(h_0 \cdot \prod_{\forall j \in \mathcal{A}_k} h_j^{m_j} \cdot \mathcal{G},\, \hat{h})$
$X_2 \leftarrow e(A,\, y\hat{h}^c) e(\tilde{h},\, \hat{h}^{r - mult} y^{-\rho})$
if $C \neq h^{\rho}\tilde{h}^{open}$ **then**
  return `False`
else if $1 \neq C^c h^{-mult}\tilde{h}^{-tmp}$ **then**
  return `False`
else if $X_1 \neq X_2 \cdot \prod_{\forall j \in \mathcal{A}_h} e(h_j,\, \hat{h})^{-m_j} \cdot e(h_{l+1},\, \hat{h})^{-s}$ **then**
  return `False`
else if $e(\mathcal{G},\, acc_V) \neq e(g,\, \mathcal{W}) \cdot z \cdot e(\tilde{h},\, acc_V)^r e(g^{-1},\, g')^{r'}$ **then**
  return `False`
else if $D \neq g^r \tilde{h}^{open'}$ **then**
  return `False`
else if $1 \neq D^{r''} g^{-mult'} \tilde{h}^{-tmp'}$ **then**
  return `False`
else if $e(pk \cdot \mathcal{G},\, \mathcal{S}) \neq e(g,\, g') e(pk \cdot \mathcal{G},\, g')^{r''} e(\tilde{h},\, g')^{-mult'} e(\tilde{h},\, \mathcal{S})^r$ **then**
  return `False`
else if $e(\mathcal{G},\, u) \neq e(g,\, \mathcal{U}) e(\tilde{h},\, u)^r e(g^{-1},\, g')^{r'''}$ **then**
  return `False`
else
  return `True`
end if