Detailed Security Proofs for "HNE-CPPMA: A Hybrid NTRU–ECC Conditional Privacy-Preserving Message Authentication Scheme with Dynamic Credential Binding for VANETs"

Feixiang Ye and Zhiquan Liu

I. Security Assumptions

Definition 1 (NTRU Lattice)**.** Let λ_1 denote the dimension of the NTRU ring and let $q_1 > 5$ be a prime modulus. Define the polynomial rings $\mathbb{R} = \mathbb{Z}[x]/(x^{\lambda_1}+1)$ and $\mathbb{R}_{q_1} = \mathbb{Z}_{q_1}[x]/(x^{\lambda_1}+1)$, where $\mathbb{Z}_{q_1}$ denotes the integers modulo q_1. For $f, g \in \mathbb{R}$, where f is invertible modulo q_1, let $h = gf^{-1} \bmod q_1 \in \mathbb{R}_{q_1}^{\times}$. The corresponding NTRU lattice is $\Lambda_h^{q_1} = \{(d_1, d_2) \in \mathbb{R}^2 | d_1 + d_2 h = 0 \bmod q_1\}$, which is a full rank $2\lambda_1$-dimensional lattice generated by $\begin{pmatrix} -C_{\lambda_1}(h) & I_{\lambda_1} \\ q_1 I_{\lambda_1} & O_{\lambda_1} \end{pmatrix}$, where I_{λ_1} and O_{λ_1} are the $\lambda_1 \times \lambda_1$ unit and zero matrices, respectively, and $C_{\lambda_1}(h)$ is the anti-circulant matrix.

Definition 2 (NTRU-SIS)**.** Given (λ_1, q_1, β), $h \in \mathbb{R}_{q_1}^{\times}$ and $\Lambda_h^{q_1}$, the NTRU-SIS$_{\lambda_1,q_1,\beta}$ problem is to find a nonzero pair $(d_1, d_2) \in \Lambda_h^{q_1}$ such that $d_1 + d_2 h = 0 \bmod q_1$ and $\|(d_1, d_2)\| \le \beta$, where $\|\cdot\|$ denotes the Euclidean norm.

For $j \in \{2, 3\}$, let $E_j(a_j, b_j)$ be an elliptic curve with security parameter λ_j, defined by $y^2 = x^3 + a_j x + b_j$ on the finite field $\mathbb{F}_{p_j}$, where $a_j, b_j \in \mathbb{F}_{p_j}$ and $4a_j^3 + 27b_j^2 \neq 0$. Let $\mathbb{G}_j$ be an additive cyclic group with prime order q_j and generator P_j on $E_j(a_j, b_j)$. Scalar multiplication is denoted by $xP_j = P_j + P_j + \cdots + P_j$ (x times), where $x \in \mathbb{Z}_{q_j}^*$. In HNE-CPPMA, $E_2(a_2, b_2)$ is used for chameleon hash binding, while $E_3(a_3, b_3)$ is used for message signing, with $\lambda_2 > \lambda_3$.

Definition 3 (ECDLP)**.** Given $(\mathbb{G}_j, P_j, Q_j)$ over $E_j(a_j, b_j)$, it is computationally difficult for the probability polynomial time ($\mathcal{PPT}$) algorithm to compute $x \in \mathbb{Z}_{q_j}^*$ so that $Q_j = xP_j$ holds.

Definition 4 (CTB-ECDLP assumption)**.** Given $(\mathbb{G}_j, P_j, Q_j = xP_j)$ over $E_j(a_j, b_j)$, it is computationally difficult for a resource-constrained quantum adversary whose running time is bounded by T to compute $x \in \mathbb{Z}_{q_j}^*$ so that $Q_j = xP_j$ holds, where $T < T_Q^{E_j}$.

II. Formal Security Analysis

An adversary can break HNE-CPPMA through one of the following three cases.

1) *NTRU-Based Credential Forgery:* In this case, the adversary may first forge a valid NTRU-based credential for a forged chameleon hash value, then bind a forged pseudonym and public key to this chameleon hash value, and finally forge message signatures by using the forged pseudonym and public key.
2) *Chameleon Hash Binding Forgery:* In this case, the adversary may forge a valid binding tuple for an existing valid chameleon hash value without the trapdoor, and then use the forged pseudonym and public key in the tuple to forge message signatures.
3) *ECC-Based Message Signature Forgery:* In this case, the adversary may directly forge message signatures under a valid pseudonym and public key.

According to the above possible attack cases, under the random oracle model (ROM), Lemma 1 reduces the security of the NTRU-based credential to the NTRU-SIS problem, while Lemmas 2 and 3 reduce the security of the chameleon hash function and the ECC-based message authentication to the ECDLP over $E_2(a_2, b_2)$ and $E_3(a_3, b_3)$, respectively. Then, based on Lemmas 1-3, Theorem 1 further demonstrates that HNE-CPPMA achieves conditional time-bounded quantum resilience against adaptive chosen-message/binding attacks.

Lemma 1. Assume that the NTRU-SIS$_{\lambda_1,q_1,\beta}$ problem is hard over the NTRU lattice $\Lambda_h^{q_1}$. Then, under the ROM, the NTRU-based shared PQ credential of HNE-CPPMA is EUF-ACMA secure against the adversary $\mathcal{A}_1$.

Proof. Assume that there exists a $\mathcal{PPT}$ adversary $\mathcal{A}_1$ that can forge a valid NTRU-Based credential with non-negligible probability ϵ_1. We can construct a challenger $\mathcal{C}_1$ that uses $\mathcal{A}_1$'s ability to solve the NTRU-SIS problem.

Given an instance of the NTRU-SIS$_{\lambda_1,q_1,\beta}$ problem defined by the public polynomial $h \in R_{q_1}^{\times}$, the goal of $\mathcal{C}_1$ is to find $d = (d_1, d_2) \in R^2$ satisfying $d_1 + h d_2 = 0$ and $\|(d_1, d_2)\| \le \beta$, where $\beta = 2s\sqrt{2\lambda_1}$.

Initialization Phase: $\mathcal{C}_1$ initializes the system and generates $params = \{\mathbb{G}_2, \mathbb{G}_3, P_2, P_3, q_1, q_2, q_3, h, H_2, H_3, H_4, H_5\}$. Here, h is directly set as the public NTRU parameter in the given NTRU-SIS instance. Note that $\mathcal{C}_1$ does not know the master private key B.

In order to simulate the random oracle and credential generation oracle, $\mathcal{C}_1$ initializes and maintains a list $L_{H_3} = (ch_v, T_v, d_{v,1}, d_{v,2}, R_v)$. Finally, $\mathcal{C}_1$ returns $params$ to $\mathcal{A}_1$.

Query Phase: In this phase, $\mathcal{A}_1$ can adaptively send the following queries to $\mathcal{C}_1$.

1) H_3 *Oracle Query:* $\mathcal{A}_1$ submits (ch_v, T_v) as input to this query. If (ch_v, T_v) already exists in L_{H_3}, then $\mathcal{C}_1$ returns the corresponding value R_v to $\mathcal{A}_1$. Otherwise, $\mathcal{C}_1$ randomly samples two short polynomials $d_{v,1}, d_{v,2}$ according to the GPV discrete Gaussian distribution over the corresponding preimage domain, and programs the random oracle as $R_v = d_{v,1} + hd_{v,2}$. Then $\mathcal{C}_1$ inserts $(ch_v, T_v, d_{v,1}, d_{v,2}, R_v)$ into L_{H_3} and returns R_v to $\mathcal{A}_1$.
2) *NTRU-Based Signature Query:* $\mathcal{A}_1$ issues this query with (ch_v, T_v) as input. Then, $\mathcal{C}_1$ checks whether there exists an entry associated with (ch_v, T_v) in L_{H_3}. If there already exists an entry associated with (ch_v, T_v) in L_{H_3}, $\mathcal{C}_1$ directly returns the corresponding $(d_{v,1}, d_{v,2})$ to $\mathcal{A}_1$. If no such entry exists in L_{H_3}, $\mathcal{C}_1$ first performs the above H_3-*Oracle Query* to generate the corresponding tuple $(ch_v, T_v, d_{v,1}, d_{v,2}, R_v)$. Then, $\mathcal{C}_1$ returns $(d_{v,1}, d_{v,2})$ to $\mathcal{A}_1$. Moreover, by the preimage-sampling property of the GPV framework, the simulated signature is statistically close to a real signature generated with the NTRU trapdoor.

Forgery Phase: Finally, $\mathcal{A}_1$ outputs a forged NTRU-Based shared PQ credential $CYT_v^* = \{d_{v,1}^{*'}, d_{v,2}^{*'}, ch_v^*, T_v^*\}$. $\mathcal{A}_1$ wins the game if the signature $(d_{v,1}^{*'}, d_{v,2}^{*'})$ is valid, and (ch_v^*, T_v^*) has not been queried to the *NTRU-Based Signature Query*. For ease of notation, all parameters associated with (ch_v^*, T_v^*) are marked with $*$. Then, $\mathcal{C}_1$ can derive

$$\|(d_{v,1}^{*'}, d_{v,2}^{*'})\| \leq s\sqrt{2\lambda_1} \tag{3}$$

$$d_{v,1}^{*'} + hd_{v,2}^{*'} = R_v^* \tag{4}$$

Since $\mathcal{A}_1$ needs to output a valid signature for (ch_v^*, T_v^*), which is equivalent to finding the corresponding preimage of R_v^*, the probability that $\mathcal{A}_1$ has not queried (ch_v^*, T_v^*) to the H_3 *Oracle Query* before forgery is negligible, denoted by ξ. Therefore, $\mathcal{C}_1$ can recover $(ch_v^*, T_v^*, d_{v,1}^*, d_{v,2}^*, R_v^*)$ from the list L_{H_3}. Then, $\mathcal{C}_1$ can derive

$$\|(d_{v,1}^*, d_{v,2}^*)\| \leq s\sqrt{2\lambda_1} \tag{5}$$

$$d_{v,1}^* + hd_{v,2}^* = R_v^* \tag{6}$$

Then, by combining the equations 4 and 6, $\mathcal{C}_1$ can derive the equation $d_{v,1}^* + hd_{v,2}^* = d_{v,1}^{*'} + hd_{v,2}^{*'} = R_v^*$, which implies that $(d_{v,1}^* - d_{v,1}^{*'}) + h(d_{v,2}^* - d_{v,2}^{*'}) = 0$.

Since $(d_{v,1}^{*'}, d_{v,2}^{*'})$ is a forged signature, $\mathcal{A}_1$ has not submitted (ch_v^*, T_v^*) as an input to the *NTRU-Based Signature Query*. Hence, $(d_{v,1}^*, d_{v,2}^*)$ has never been revealed to $\mathcal{A}_1$. Meanwhile, by the preimage min-entropy property [1], [2], the probability that the preimage output by $\mathcal{A}_1$ satisfies $(d_{v,1}^*, d_{v,2}^*) = (d_{v,1}^{*'}, d_{v,2}^{*'})$ is at most $2^{-\omega(log\lambda_1)}$, which is negligible. Therefore, this implies that $(d_{v,1}^*, d_{v,2}^*) \neq (d_{v,1}^{*'}, d_{v,2}^{*'})$. By setting $d_1 = d_{v,1}^* - d_{v,1}^{*'}$, $d_2 = d_{v,2}^* - d_{v,2}^{*'}$, $\mathcal{C}_1$ can derive $d_1 + hd_2 = 0$, where $(d_1, d_2) \neq (0, 0)$. Furthermore, by combining the inequalities 3 and 5, $\mathcal{C}_1$ can derive $\|(d_1, d_2)\| = \|(d_{v,1}^*, d_{v,2}^*) - (d_{v,1}^{*'}, d_{v,2}^{*'})\| \leq \|(d_{v,1}^*, d_{v,2}^*)\| + \|(d_{v,1}^{*'}, d_{v,2}^{*'})\| \leq \beta$. Therefore, $\mathcal{C}_1$ can output (d_1, d_2) as the solution to the NTRU-SIS problem.

Finally, it follows that $\mathcal{C}_1$ can solve the NTRU-SIS problem with non-negligible probability $\epsilon_1 - \xi - 2^{-\omega(\log \lambda_1)}$. However, by Definition 2, this contradicts the assumed hardness of the NTRU-SIS problem. Therefore, we can conclude that the NTRU-based shared PQ credential of HNE-CPPMA is EUF-ACMA secure against $\mathcal{A}_1$. □

Lemma 2. Assume that the ECDLP is hard in the additive cyclic group $\mathbb{G}_2$ with prime order q_2 and generator P_2. Then, under the ROM, the chameleon hash function used in HNE-CPPMA is ETCRBU-ACBA secure against the adversary $\mathcal{A}_2$.

Proof. Assume that there exists a $\mathcal{PPT}$ adversary $\mathcal{A}_2$ that can forge a valid new binding tuple for the chameleon hash value with non-negligible probability ϵ_2. We can construct a challenger $\mathcal{C}_2$ that uses $\mathcal{A}_2$'s ability to solve the ECDLP over $\mathbb{G}_2$.

Let the ECDLP instance be $(\mathbb{G}_2, P_2, Q = tkP_2)$, where $tk \in Z_{q_2}^*$ is unknown to $\mathcal{C}_2$. The goal of $\mathcal{C}_2$ is to compute tk.

Initialization Phase: $\mathcal{C}_2$ initializes the system and generates $params = \{\mathbb{G}_2, \mathbb{G}_3, P_2, P_3, q_1, q_2, q_3, h, H_2, H_3, H_4, H_5\}$. In addition, to simulate the practical environment of HNE-CPPMA, $\mathcal{C}_2$ generates d chameleon hash values $\mathcal{GP}_{ch} = (ch_1, ch_2, \ldots, ch_d)$ and the corresponding trapdoors $(tpd_1, tpd_2, \ldots, tpd_d)$, where $tpd_v = (tk_v, y_v)$ and $v \in \{1, 2, \ldots, d\}$. Then, $\mathcal{C}_2$ randomly selects one chameleon hash value from $\mathcal{GP}_{ch}$ as the challenge chameleon hash value, denoted by ch_v^*, and replaces it with $ch_v^* = Q$. Therefore, the corresponding trapdoor tpd_v^* of ch_v^* is unknown to $\mathcal{C}_2$. Meanwhile, $\mathcal{C}_2$ initializes and maintains the lists L_{ch} and L_{H_4} to record all relevant queries from $\mathcal{A}_2$. Finally, $\mathcal{C}_2$ returns $params, \mathcal{GP}_{ch}$ to $\mathcal{A}_2$.

Query Phase: In this phase, $\mathcal{A}_2$ can adaptively issue the following queries to $\mathcal{C}_2$.

1) *Trapdoor Queries:* $\mathcal{A}_2$ submits ch_v as input to this query. If $ch_v = ch_v^*$, $\mathcal{C}_2$ aborts the simulation. Otherwise, $\mathcal{C}_2$ returns the corresponding tpd_v to $\mathcal{A}_2$.
2) H_4 *Oracle Queries:* $\mathcal{A}_2$ submits $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau})$ as input to this query. If there already exists an entry associated with $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau})$ in L_{H_4}, $\mathcal{C}_2$ directly returns the corresponding value $r_{i,\tau}$ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}_2$ randomly selects $r_{i,\tau} \in \mathbb{Z}_{q_2}^*$ and returns $r_{i,\tau}$ to $\mathcal{A}_2$. Then, $\mathcal{C}_2$ inserts $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau}, r_{i,\tau})$ into the list L_{H_4}.
3) *Chameleon Hash Binding Queries:* $\mathcal{A}_2$ submits $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau})$ as input to this query. $\mathcal{C}_2$ randomly selects $m_{i,\tau}, r_{i,\tau} \in \mathbb{Z}_{q_2}^*$ and computes $Z_{i,\tau} = (ch_v^* - m_{i,\tau}P_2)r_{i,\tau}^{-1}$. Then, $\mathcal{C}_2$ inserts $(Z_{i,\tau}, m_{i,\tau}, r_{i,\tau})$ and $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau}, r_{i,\tau})$ into the list L_{ch} and L_{H_4}, respectively. Finally, $\mathcal{C}_2$ returns $(Z_{i,\tau}, m_{i,\tau}, r_{i,\tau})$ to $\mathcal{A}_2$.

Forgery Phase: Finally, $\mathcal{A}_2$ outputs a forged binding tuple $(PID_{i,\tau}^*, PK_{i,\tau}^*, TTL_{i,\tau}^*, Z_{i,\tau}^*, m_{i,\tau}^*)$ for ch_v^*. $\mathcal{A}_2$ wins the game if the output tuple forms a valid collision, and $\mathcal{A}_2$ has not queried *Trapdoor Queries* for ch_v^* and *Chameleon Hash Binding Queries* for $(PID_{i,\tau}^*, PK_{i,\tau}^*, TTL_{i,\tau}^*)$. Then, $\mathcal{C}_2$ recovers $(PID_{i,\tau}^*, PK_{i,\tau}^*, TTL_{i,\tau}^*, Z_{i,\tau}^*, m_{i,\tau}^*, r_{i,\tau}^*)$ from L_{ch} and L_{H_4}. For ease of notation, all parameters associated

with $PID^*_{i,\tau}$ are marked with $*$. Subsequently, $\mathcal{C}_2$ can derive equation 7.

$$ch^*_v = CH(m^*_{i,\tau}, r^*_{i,\tau}, Z^*_{i,\tau}) = m^*_{i,\tau}P_2 + r^*_{i,\tau}Z^*_{i,\tau}. \quad (7)$$

By the Forking Lemma [3], $\mathcal{C}_2$ can obtain another valid forgery $(PID^*_{i,\tau}, PK^*_{i,\tau}, TTL^*_{i,\tau}, Z^*_{i,\tau}, m^{*'}_{i,\tau})$ and derive the following equation 8, where $r^*_{i,\tau} \neq r^{*'}_{i,\tau}$.

$$ch^*_v = CH(m^{*'}_{i,\tau}, r^{*'}_{i,\tau}, Z^*_{i,\tau}) = m^{*'}_{i,\tau}P_2 + r^{*'}_{i,\tau}Z^*_{i,\tau}. \quad (8)$$

Let $Z^*_{i,\tau} = z^*P_2$. By combining the above two equations, $\mathcal{C}_2$ can derive

$$\begin{aligned} &CH(m^*_{i,\tau}, r^*_{i,\tau}, Z^*_{i,\tau}) - CH(m^{*'}_{i,\tau}, r^{*'}_{i,\tau}, Z^*_{i,\tau}) \\ &= m^*_{i,\tau}P_2 + r^*_{i,\tau}Z^*_{i,\tau} - (m^{*'}_{i,\tau}P_2 + r^{*'}_{i,\tau}Z^*_{i,\tau}) \\ &= (m^*_{i,\tau} - m^{*'}_{i,\tau})P_2 + (r^*_{i,\tau} - r^{*'}_{i,\tau})z^*P_2 \end{aligned}$$

Then, from $(m^{*'}_{i,\tau} - m^*_{i,\tau})P_2 = (r^*_{i,\tau} - r^{*'}_{i,\tau})z^*P_2$, $\mathcal{C}_2$ can compute $z^* = (m^{*'}_{i,\tau} - m^*_{i,\tau})(r^*_{i,\tau} - r^{*'}_{i,\tau})^{-1}$. Since $ch^*_v = Q = tkP_2$ and $ch^*_v = m^*_{i,\tau}P_2 + r^*_{i,\tau}Z^*_{i,\tau}$, $\mathcal{C}_2$ can further derive

$$\begin{aligned} Q &= m^*_{i,\tau}P_2 + r^*_{i,\tau}Z^* \\ &= m^*_{i,\tau}P_2 + r^*_{i,\tau}z^*P_2 \\ &= (m^*_{i,\tau} + r^*_{i,\tau}z^*)P_2. \end{aligned}$$

Therefore, $\mathcal{C}_2$ can output $tk = m^*_{i,\tau} + r^*_{i,\tau}z^*$ as the solution to the ECDLP instance $(G_2, P_2, Q = tkP_2)$ over E_2. Therefore, $\mathcal{C}_2$ does not abort during the *Query Phase* and the *Forgery Phase* with probabilities at least $(1-\frac{1}{d})^{q_{td}}$ and $\frac{\epsilon_2}{d}$, respectively. Furthermore, by the application of the Forking Lemma [3], [4], $\mathcal{A}_2$ can generate two valid forgeries that satisfy the required conditions with non-negligible probability $(1-\frac{1}{e})\frac{\epsilon_2}{q_{H_4}}$. Here, q_{td} and q_{H_4} denote the numbers of *Trapdoor Queries* and H_4 *Oracle Queries*, respectively. It follows that $\mathcal{C}_2$ can solve the ECDLP over E_2 with non-negligible probability $(1-\frac{1}{d})^{q_{td}}(1-\frac{1}{e})\frac{\epsilon_2^2}{dq_{H_4}}$. However, by Definition 3, the ECDLP over E_2 is computationally infeasible for any $\mathcal{PPT}$ adversary, which contradicts the above assumption. Therefore, we can conclude that the chameleon hash function used in HNE-CPPMA is ETCRBU-ACBA secure against $\mathcal{A}_2$. □

Lemma 3. Assume that the ECDLP is hard in the additive cyclic group $\mathbb{G}_3$ with prime order q_3 and generator P_3. Then, under the ROM, the ECC-Based signature of HNE-CPPMA is EUF-ACMA secure against the adversary $\mathcal{A}_3$.

Proof. Assume that there exists a $\mathcal{PPT}$ adversary $\mathcal{A}_3$ that can forge a valid ECC-based message with non-negligible probability ϵ_3. We can construct a challenger $\mathcal{C}_3$ that uses $\mathcal{A}_3$'s ability to solve the ECDLP over $\mathbb{G}_3$.

Let the ECDLP instance be $(\mathbb{G}_3, P_3, Q = xP_3)$, where x is unknown to $\mathcal{C}_3$. The goal of $\mathcal{C}_3$ is to compute x.

Initialization Phase: $\mathcal{C}_3$ initializes the system and generates $params = \{\mathbb{G}_2, \mathbb{G}_3, P_2, P_3, q_1, q_2, q_3, h, H_2, H_3, H_4, H_5\}$. In addition, to simulate the practical environment of HNE-CPPMA, $\mathcal{C}_3$ randomly generates credential groups $\mathcal{GP} = (\mathcal{GP}_1, \mathcal{GP}_2, \ldots, \mathcal{GP}_d)$ for binding with the pseudonyms queried by $\mathcal{A}_3$, where $\mathcal{GP}_v = \{ch_v, Y_v, tpd_v, CYT_v, T_v\}$ and $v \in \{1, 2, \ldots, d\}$. Meanwhile, $\mathcal{C}_3$ initializes and maintains four lists L_{bind}, L_{H_2}, L_{H_4} and L_{H_5} to record all relevant queries from $\mathcal{A}_3$. Finally, $\mathcal{C}_3$ returns the system parameters $params$ to $\mathcal{A}_3$.

Query Phase: In this phase, $\mathcal{A}_3$ can adaptively send the following queries to $\mathcal{C}_3$, where $\mathcal{C}_3$ uniformly selects one query from the H_5 *Oracle Queries*, indexed by $(1, \ldots, q_{H_5})$, and regards the corresponding $PID^*_{i,\tau}$ as the challenge identity.

1) *Public Key Queries:* $\mathcal{A}_3$ submits $PID_{i,\tau}$ as input to this query. If $(PK_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, TTL_{i,\tau})$ already exists in List L_{bind}, $\mathcal{C}_3$ directly returns the corresponding $PK_{i,\tau}$ to $\mathcal{A}_3$. Otherwise, $\mathcal{C}_3$ randomly selects $\mathcal{GP}_v \in \mathcal{GP}, z_{i,\tau}, r_{i,\tau} \in \mathbb{Z}^*_{q_2}$ and generates $TTL_{i,\tau}$. Then, $\mathcal{C}_3$ computes $Z_{i,\tau} = z_{i,\tau}Y_v, m_{i,\tau} = tk_v - r_{i,\tau}z_{i,\tau}y_v$ and performs the following operations.
   - If $PID_{i,\tau} \neq PID^*_{i,\tau}$, $\mathcal{C}_3$ randomly selects $a_{i,\tau}, x_{i,\tau}, h_{2i,\tau} \in \mathbb{Z}^*_{q_3}$, and computes $A_{i,\tau} = a_{i,\tau}P_3$, $X_{i,\tau} = x_{i,\tau}P_3$, $sk_{i,\tau} = a_{i,\tau} + h_{2i,\tau}x_{i,\tau}$, $PK_{i,\tau} = sk_{i,\tau}P_3$. Afterwards, $\mathcal{C}_3$ inserts $(PID_{i,\tau}, z_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, a_{i,\tau}, A_{i,\tau}, x_{i,\tau}, X_{i,\tau}, sk_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, \mathcal{GP}_v)$ into L_{bind}.
   - If $PID_{i,\tau} = PID^*_{i,\tau}$, $\mathcal{C}_3$ sets $X_{i,\tau} = Q = xP_3$ and randomly selects $a_{i,\tau}, h_{2i,\tau} \in \mathbb{Z}^*_{q_3}$. Then, $\mathcal{C}_3$ computes $A_{i,\tau} = a_{i,\tau}P_3$ and $PK_{i,\tau} = A_{i,\tau} + h_{2i,\tau}X_{i,\tau}$. Afterwards, $\mathcal{C}_3$ inserts $(PID_{i,\tau}, z_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, a_{i,\tau}, A_{i,\tau}, \perp, X_{i,\tau}, \perp, PK_{i,\tau}, TTL_{i,\tau}, \mathcal{GP}_v)$ into L_{bind}.

   Finally, $\mathcal{C}_3$ inserts $(A_{i,\tau}, X_{i,\tau}, TTL_{i,\tau}, h_{2i,\tau})$ and $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau}, r_{i,\tau})$ into L_{H_2} and L_{H_4}, respectively. Afterwards, $\mathcal{C}_3$ returns the corresponding $(PK_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, TTL_{i,\tau})$ to $\mathcal{A}_3$.
2) *Private Key Queries:* $\mathcal{A}_3$ submits $PID_{i,\tau}$ as input to this query. If $PID_{i,\tau} = PID^*_{i,\tau}$, $\mathcal{C}_3$ terminates the simulation immediately. Otherwise, if there already exists an entry associated with $PID_{i,\tau}$ in L_{bind}, $\mathcal{C}_3$ directly returns the corresponding values $(x_{i,\tau}, a_{i,\tau}, sk_{i,\tau})$ to $\mathcal{A}_3$. If no such entry exists, $\mathcal{C}_3$ issues the *Public Key Queries* on input $PID_{i,\tau}$ and returns the corresponding values $(x_{i,\tau}, a_{i,\tau}, sk_{i,\tau})$ to $\mathcal{A}_3$.
3) H_2 *Oracle Queries:* $\mathcal{A}_3$ submits $(A_{i,\tau}, X_{i,\tau}, TTL_{i,\tau})$ as input to this query. If there already exists an entry associated with $(A_{i,\tau}, X_{i,\tau}, TTL_{i,\tau})$ in L_{H_2}, $\mathcal{C}_3$ directly returns the corresponding value $h_{2i,\tau}$ to $\mathcal{A}_3$. Otherwise, $\mathcal{C}_3$ randomly selects $h_{2i,\tau} \in \mathbb{Z}^*_{q_3}$ and returns $h_{2i,\tau}$ to $\mathcal{A}_3$. Then, $\mathcal{C}_3$ inserts $(A_{i,\tau}, X_{i,\tau}, TTL_{i,\tau}, h_{2i,\tau})$ into the list L_{H_2}.
4) H_4 *Oracle Queries:* $\mathcal{A}_3$ submits $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau})$ as input to this query. If there already exists an entry associated with $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau})$ in L_{H_4}, $\mathcal{C}_3$ directly returns the corresponding value $r_{i,\tau}$ to $\mathcal{A}_3$. Otherwise, $\mathcal{C}_3$ randomly selects $r_{i,\tau} \in \mathbb{Z}^*_{q_2}$ and returns $r_{i,\tau}$ to $\mathcal{A}_3$. Then, $\mathcal{C}_3$ inserts $(PID_{i,\tau}, PK_{i,\tau}, TTL_{i,\tau}, Z_{i,\tau}, r_{i,\tau})$ into the list L_{H_4}.
5) H_5 *Oracle Queries:* $\mathcal{A}_3$ submits $(PID_{i,\tau}, PK_{i,\tau}, U_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, t_{i,\tau}, TTL_{i,\tau}, Ms_{i,\tau})$ as input to this query. If the list L_{H_5} already contains the tuple

$(PID_{i,\tau}, PK_{i,\tau}, U_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, t_{i,\tau}, TTL_{i,\tau}, Ms_{i,\tau}, h_{5i,\tau})$, then $\mathcal{C}_3$ directly returns the corresponding value $h_{5i,\tau}$ to $\mathcal{A}_3$. Otherwise, $\mathcal{C}_3$ randomly selects $h_{5i,\tau} \in \mathbb{Z}_{q_3}^*$ and returns $h_{5i,\tau}$ to $\mathcal{A}_3$. Then, $\mathcal{C}_3$ inserts $(PID_{i,\tau}, PK_{i,\tau}, U_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, t_{i,\tau}, TTL_{i,\tau}, Ms_{i,\tau}, h_{5i,\tau})$ into the list L_{H_5}.

6) *Signature Queries:* $\mathcal{A}_3$ issues this query with $(PID_{i,\tau}, Ms_{i,\tau})$ as inputs. $\mathcal{C}_3$ first checks whether there exists an entry associated with $PID_{i,\tau}$ in L_{bind}. If no such entry exists, $\mathcal{C}_3$ issues the *Public Key Queries* with $PID_{i,\tau}$ as input. Then, $\mathcal{C}_3$ extracts $(PID_{i,\tau}, PK_{i,\tau}, Z_{i,\tau}, TTL_{i,\tau}, m_{i,\tau})$ from L_{bind} and selects $t_{i,\tau}$. Afterwards, $\mathcal{C}_3$ randomly selects $\sigma_{i,\tau}, h_{5i,\tau} \in \mathbb{Z}_{q_3}^*$, and computes $U_{i,\tau} = \sigma_{i,\tau}P_3 - h_{5i,\tau}PK_{i,\tau}$. Finally, $\mathcal{C}_3$ inserts $(PID_{i,\tau}, PK_{i,\tau}, U_{i,\tau}, Z_{i,\tau}, m_{i,\tau}, t_{i,\tau}, TTL_{i,\tau}, Ms_{i,\tau}, h_{5i,\tau})$ into the list L_{H_5} and returns $\sigma_{i,\tau}, U_{i,\tau}, t_{i,\tau}$ to $\mathcal{A}_3$.

Forgery Phase: Finally, $\mathcal{A}_3$ forges an ECC-based signature $\sigma_{i,\tau}^*$ for the challenge identity $PID_{i,\tau}^*$ on message $Ms_{i,\tau}^*$. $\mathcal{A}_3$ wins the game if the signature $\sigma_{i,\tau}^*$ is valid, and $\mathcal{A}_3$ has not queried *Private Key Queries* for $PID_{i,\tau}^*$ and *Signature Queries* for $(PID_{i,\tau}^*, Ms_{i,\tau}^*)$. Then, $\mathcal{C}_3$ recovers $(PK_{i,\tau}^*, A_{i,\tau}^*, X_{i,\tau}^*, U_{i,\tau}^*, a_{i,\tau}^*, h_{2i,\tau}^*, h_{5i,\tau}^*)$ from $L_{bind}, L_{H_2}, L_{H_5}$. For ease of notation, all parameters associated with $PID_{i,\tau}^*$ are marked with $*$. Subsequently, $\mathcal{C}_3$ can derive equation 9.

$$\sigma_{i,\tau}^* P_3 = U_{i,\tau}^* + h_{5i,\tau}^* PK_{i,\tau}^*. \tag{9}$$

By the Forking Lemma [3], $\mathcal{C}_3$ can obtain another valid forgery $\sigma_{i,\tau}^{*'}$ and derive the following equation 10, where $h_{5i,\tau}^* \neq h_{5i,\tau}^{*'}$.

$$\sigma_{i,\tau}^{*'} P_3 = U_{i,\tau}^* + h_{5i,\tau}^{*'} PK_{i,\tau}^*. \tag{10}$$

Then, by combining the two equations above, $\mathcal{C}_3$ can derive

$$\begin{aligned}
&(\sigma_{i,\tau}^* - \sigma_{i,\tau}^{*'})P_3 \\
&= \sigma_{i,\tau}^* P_3 - \sigma_{i,\tau}^{*'} P_3 \\
&= U_{i,\tau}^* + h_{5i,\tau}^* PK_{i,\tau}^* - (U_{i,\tau}^* + h_{5i,\tau}^{*'} PK_{i,\tau}^*) \\
&= h_{5i,\tau}^* PK_{i,\tau}^* - h_{5i,\tau}^{*'} PK_{i,\tau}^* \\
&= (h_{5i,\tau}^* - h_{5i,\tau}^{*'}) PK_{i,\tau}^* \\
&= (h_{5i,\tau}^* - h_{5i,\tau}^{*'})(A_{i,\tau}^* + h_{2i,\tau}^* Q) \\
&= (h_{5i,\tau}^* - h_{5i,\tau}^{*'})(a_{i,\tau}^* + h_{2i,\tau}^* x)P_3
\end{aligned}$$

Finally, $\mathcal{C}_3$ can output $x = [(\sigma_{i,\tau}^* - \sigma_{i,\tau}^{*'})(h_{5i,\tau}^* - h_{5i,\tau}^{*'})^{-1} - a_{i,\tau}^*]h_{2i,\tau}^{*-1}$ as the solution to the ECDLP instance $(\mathbb{G}_3, P_3, Q = xP_3)$ over E_3. Therefore, $\mathcal{C}_3$ does not abort during the *Query Phase* and the *Forgery Phase* with probabilities at least $(1 - \frac{1}{q_{H_5}})^{q_{sk}}$ and $\frac{\epsilon_3}{q_{H_5}}$, respectively. Here, q_{H_5} and q_{sk} denote the numbers of H_5 *Oracle Queries* and *Private Key Queries*, respectively. Furthermore, by the application of the Forking Lemma [3], [4], $\mathcal{A}_3$ can generate two valid forged signatures satisfying the required conditions with non-negligible probability $(1 - \frac{1}{e})\frac{\epsilon^3}{q_{H_5}}$. It follows that $\mathcal{C}_3$ can solve the ECDLP over E_3 with non-negligible probability $(1-\frac{1}{q_{H_5}})^{q_{sk}}(1-\frac{1}{e})\frac{\epsilon^3}{q_{H_5}^2}$. However, by Definition 3, the ECDLP over E_3 is computationally infeasible for any PPT adversary, which contradicts the above assumption. Therefore, we can conclude that the ECC-Based signature of HNE-CPPMA is EUF-ACMA secure against $\mathcal{A}_3$. □

Theorem 1. Assume that the NTRU-SIS$_{\lambda_1,q_1,\beta}$ problem is intractable for quantum polynomial-time algorithms, and that the CTB-ECDLP assumption hold. If $T_v^{exp} - T_v^{start} < T_Q^{E_2}$ and $TTL_{i,\tau}^{exp} - TTL_{i,\tau}^{start} < T_Q^{E_3}$, then HNE-CPPMA achieves conditional time-bounded quantum resilience against adaptive chosen-message/binding attacks under the ROM.

Proof. Assume that there exists a quantum adversary $\mathcal{A}_Q$ that can break HNE-CPPMA with a non-negligible probability ϵ_Q. According to the above analysis, $\mathcal{A}_Q$ must satisfy at least one of the following three cases.

1) *NTRU-Based Credential Forgery:* If $\mathcal{A}_Q$ forges a valid NTRU-based credential, then by Lemma 1, we can construct an algorithm that solves the NTRU-SIS problem with non-negligible probability, which contradicts the assumed intractability of the NTRU-SIS problem.
2) *Chameleon Hash Binding Forgery:* If $\mathcal{A}_Q$ forges a new valid chameleon hash binding tuple without the corresponding trapdoor, then by Lemma 2, we can construct an algorithm that solves the ECDLP over $E_2(a_2, b_2)$ with non-negligible probability. Since the forgery must be completed before the corresponding credential expires, it contradicts the assumption of CTB-ECDLP over $E_2(a_2, b_2)$ within T_v.
3) *ECC-Based Message Signature Forgery:* If $\mathcal{A}_Q$ forges a valid ECC-based message signature, then by Lemma 3, we can construct an algorithm that solves the ECDLP over $E_3(a_3, b_3)$ with non-negligible probability. Since the forgery must be completed before the expiration of the corresponding pseudonym and public key, it contradicts the assumption of CTB-ECDLP over $E_3(a_3, b_3)$ within $TTL_{i,\tau}$.

Since any successful forgery would contradict one of the underlying hardness assumptions, no quantum adversary can break HNE-CPPMA with a non-negligible probability before the expiration of the corresponding credential or pseudonym. Therefore, HNE-CPPMA achieves conditional time-bounded quantum resilience against adaptive chosen-message/binding attacks. □

III. Correctness Proof

In this Section, we prove the correctness of the chameleon hash binding, the ECC-based individual message authentication, and the ECC-based batch message authentication.

The correctness proof of chameleon hash binding is as follows:

$$\begin{aligned}
CH(m_{i,\tau}, r_{i,\tau}, Z_{i,\tau}) &= m_{i,\tau}P_2 + r_{i,\tau}Z_{i,\tau} \\
&= (tk_v - r_{i,\tau}z_{i,\tau}y_v)P_2 + r_{i,\tau}z_{i,\tau}y_v P_2 \\
&= tk_v P_2 = (m_v + r_v y_v)P_2 \\
&= m_v P_2 + r_v Y_v = ch_v
\end{aligned}$$

The correctness of the individual signature verification is shown as follows:

$$\begin{aligned}\sigma_{i,\tau}P_3 &= (u_{i,\tau} + h_{5i,\tau}sk_{i,\tau})P_3 \\ &= u_{i,\tau}P_3 + h_{5i,\tau}sk_{i,\tau}P_3 \\ &= U_{i,\tau} + h_{5i,\tau}PK_{i,\tau}.\end{aligned}$$

The correctness of the batch verification is proved as follows:

$$\begin{aligned}\left(\sum_{i=1}^{n} v_i\sigma_{i,\tau}\right)P_3 &= \sum_{i=1}^{n} v_i\sigma_{i,\tau}P_3 \\ &= \sum_{i=1}^{n} v_i\left(U_{i,\tau} + h_{5i,\tau}PK_{i,\tau}\right) \\ &= \sum_{i=1}^{n} v_iU_{i,\tau} + \sum_{i=1}^{n} v_ih_{5i,\tau}PK_{i,\tau}.\end{aligned}$$

REFERENCES

[1] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," *Electron. Colloquium Comput. Complex.*, vol. TR07, 2007.

[2] ——, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, C. Dwork, Ed. ACM, 2008, pp. 197–206.

[3] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[4] W. Wu and F. Ye, "IPCAS: an improved conditional privacy-preserving certificateless aggregate signature scheme without bilinear pairing for vanets," *J. Syst. Archit.*, vol. 152, p. 103175, 2024.