1 FORMAL SECURITY ANALYSIS USING ROR MODEL

We employ the widely-adopted game-based method for ROR model to prove the security of the proposed mutual authentication and key exchange protocol. Based on the ROR model predefined, the game-based safety proof simulates various competitive scenarios (games) to identify and exclude insecure conditions of the protocol, and analyzes the adversary's advantage to demonstrate the protocol's resilience against a range of potential threats. The ROR model consists of the following elements:

- *Participants*: We denote the instance of the t_1-th device and t_2-th fog node by $I_d^{t_1}$ and $I_f^{t_2}$, respectively. Furthermore, we define I_x^t as a generic instance that may represent either $I_d^{t_1}$ or $I_f^{t_2}$.
- *Partnering*: We say that two entities $I_x^{t_1}$ and $I_x^{t_2}$ are partnered if all of the following conditions are met: 1) Both $I_x^{t_1}$ and $I_x^{t_2}$ are in the accepted state. 2) $I_x^{t_1}$ and $I_x^{t_2}$ have mutually authenticated each other. 3) $I_x^{t_1}$ and $I_x^{t_2}$ are partners to each other.
- *Freshness*: The instance I_x^t is fresh if the session key held by the instance I_x^t has not been known by $\mathcal{A}$ through the *Reveal* query defined below.
- *Adversary*: Based on the threat model discussed in Section **??**, we assume that $\mathcal{A}$ can fully control the communication over a public channel and launch types of attacks on it, including eavesdropping, modifying, fabricating and injecting messages. Furthermore, $\mathcal{A}$ can also extract the parameters stored in participants by performing kinds of physical attacks.The abilities of the adversary are simulated by the query oracles defined in Table 1.
- *Random Oracles*: In addition, the cryptographic hash function, modeled as the random oracle RO_{Hash} i.e. its output is truly random, is accessible to both $\mathcal{A}$ and honest participants in proposed scheme. Also, the PUF is modeled as RO_{PUF} similarly. The only difference is that RO_{PUF} cannot be accessed by $\mathcal{A}$.

In the ROR model, $\mathcal{A}$ is allowed to perform numbers of the queries defined above, and tries to distinguish between the real generated sk and a same-sized random number at the end of game, i.e. to guess the value of hidden bit b invovlved in $Test(I^t)$ query. Let $Succ$ denote the event that $\mathcal{A}$ wins this game, that is, the guessed bit $b' = b$. By definition, the advantage of $\mathcal{A}$ to break the semantic security(SS) of our scheme $\mathcal{P}$ is:

$$Adv_{\mathcal{P}}^{SS}(\mathcal{A}) = |Pr[Succ] - \frac{1}{2}| \tag{1}$$

For a PPT adversary, if there exists a negligible function ϵ such that $Adv_{\mathcal{P}}^{SS}(\mathcal{A}) < \epsilon$, then this protocol $\mathcal{P}$ is considered semantically secure in the ROR model.

Theorem 1.1. *Let $\mathcal{P}$ be the proposed DF-MAKE protocol , and let $\mathcal{A}$ be a PPT adversary in the ROR model that compromises $\mathcal{P}$, with up to q_h Hash queries, q_s Send queries, and q_e Execute queries allowed. Let l_n, l_r and l_h represent the lengths of the nonces, PUF responses, and hash digests, respectively. Then, the advantage $Adv_{\mathcal{P}}^{SS}(\mathcal{A})$ can be bounded as:*

$$\begin{aligned} Adv_{\mathcal{P}}^{SS}(\mathcal{A}) \leq\ & 12\epsilon_{Hash} + 4\epsilon_{PUF} + 2\epsilon_{AES} \\ & + \frac{(q_s+q_e)^2 + 5q_s}{2^{l_n+1}} + \frac{q_h^2 + 8q_s}{2^{l_h+1}} + \frac{q_s}{2^{l_r-1}} \end{aligned} \tag{2}$$

Proof. Similar to [**?**], we introduce seven games $G_i (i = 0, 1, 2, 3, 4, 5, 6)$ to prove our scheme's security, with events $Succ_i$ representing $\mathcal{A}$'s success in guessing the hidden bit b in G_i.

G_0: The starting game is defined as real attack against the proposed scheme $\mathcal{P}$ by $\mathcal{A}$. From the definition of SS, the advantage of $\mathcal{A}$ at the beginning game G_0 is:

$$Adv_{\mathcal{P}}^{SS}(\mathcal{A}) = |Pr[Succ_0] - \frac{1}{2}| \tag{3}$$

G_1: This game is constructed from the mechanisms of RO_{Hash} and RO_{PUF}, which are instinguishable from real hash function and PUF instances. According to the entropy-smoothing assumption for outputs of hash functions and responses of PUFs, we have:

$$|Pr[Succ_1] - Pr[Succ_0]| \leq 12\epsilon_{Hash} + 4\epsilon_{PUF} \tag{4}$$

where ϵ_{Hash} and ϵ_{PUF} are negligible.

G_2: This game considers potential nonces collisions. Collision occurs when an honest session selects a nonce previously used, or the adversary has chosen in an intiator request. Given the previous assumptions, a total of $(q_s + q_e)$ pairs of nonces will be independently and randomly picked, and $\mathcal{A}$ can perform q_s *Send* queries. Based on birthday paradox and the classical probability model, the occurrence probabilities for these two types of collisions are $\frac{(q_s+q_e)^2}{2^{l_n+1}}$ and $\frac{q_s}{2^{l_n}}$, respectively. Besides, in this game we also abort collisions of hash output. In summary, we claim that:

$$|Pr[Succ_2] - Pr[Succ_1]| \leq \frac{(q_s+q_e)^2 + 2q_s}{2^{l_n+1}} + \frac{q_h^2}{2^{l_h+1}} \tag{5}$$

Table 1: Queries of Adversary

Query	Purpose
$Send(I_x^t, msg)$	This query simulates an active attack by $\mathcal{A}$. By executing this query, $\mathcal{A}$ can send an arbitrary message msg to a participant instance I_x^t, and get the response message according to the proposed protocol. This query also allows $\mathcal{A}$ to initiate a protocol by executing $Send(I_d^t, start)$.
$Execute(I_d^{t_1}, I_f^{t_2})$	This query is modeled as a passive eavesdropping attack by $\mathcal{A}$. By executing this query, $\mathcal{A}$ can intercept and get all the messages transmitted over a public channel during an honest execution of a protocol between $I_d^{t_1}$ and $I_f^{t_2}$.
$Reveal(I_x^t)$	This query allows $\mathcal{A}$ to reveal the present session key sk established by I_x^t.
$Corrupt(I_x^t)$	This query simulates physical attacks, such as capture/side-channel attacks, on IoT devices or fog nodes where $\mathcal{A}$ can obtain all the secret parameters stored in the corrupted entities.
$Test(I_x^t)$	When $\mathcal{A}$ executes this query, the response is based on the state of the instance I_x^t: • If I_x^t has not yet established a fresh session key, the $Test(I_x^t)$ query returns a null value ($\perp$). • If I_x^t has successfully generated a fresh session key, the response is determined by a randomly selected hidden bit b: this query returns the real session key when $b = 1$, or a same-sized random number when $b = 0$. This query means as the semantic security of the session key sk.

G_3: In this game, we simulate the event that the attacker luckily obtains the secret authentication keys HD_{1f} and HD_{2d}. If $\mathcal{A}$ successfully guesses a enc/decrytion key HD, it can forge a valid message msg to impersonate one participant in the protocol, or decrypt messages to get the nonces thereby caculating the session key sk. $\mathcal{A}$ can obtain HD_{1f} and HD_{2d}:

- $\mathcal{A}$ corrupts the device to obtain ID and then randomly guesses either R_d or R_f to caculate $HD_{1f} = h(R_f \| ID)$, $HD_{2d} = h(h(R_d \| ID))$. Additionally, $\mathcal{A}$ can also guess the R_d^{new} or R_f^{new} for future authentication. The total probability of successfully guessing PUF response R is at most $\frac{4q_s}{2^{l_r}}$.
- Without resorting to R_d and R_d, adversary $\mathcal{A}$ directly guesses HD_1f and HD_2d, including both new and old versions, with the probability of a correct guess being $\frac{4q_s}{2^{l_h}}$.
- $\mathcal{A}$ gets either MIX_1 or MIX_2 by the $Corrupt(I_x^t)$ query and subsequently guesses HD_{1d} or HD_{2f}. Using the XOR relationships $HD_{1f} = HD_{1d} \oplus MIX_1$ and $HD_{2d} = HD_{2f} \oplus MIX_2$, $\mathcal{A}$ is able to deduce HD_{1f} and HD_{2d}. The probability is not more than $\frac{4q_s}{2^{l_h}}$.

Therefore, we have:

$$|Pr[Succ_3] - Pr[Succ_2]| \leq \frac{4q_s}{2^{l_r}} + \frac{8q_s}{2^{l_h}} \tag{6}$$

G_4: This game is defined by the event that the encryption is broken; specifically, $\mathcal{A}$ unexpectedly forges authentication tokens A_1 or A_2, which should have been encrypted with the keys HD_{1f} or HD_{2d}, in the absence of such keys. Let the ϵ_{AES} be the probability of this event. Hence, we claim that:

$$|Pr[Succ_4] - Pr[Succ_3]| \leq 2\epsilon_{AES} \tag{7}$$

G_5: This game simulates the event that $\mathcal{A}$ is lucky in obtaining nonces used in the protocol. The G_3 and G_4 have excluded the probability of decrypting A_1 and A_2 both with and without secret keys. In this game, attacker tries to randomly guessing the nonces, devoid of any additional knowledges. We have:

$$|Pr[Succ_5] - Pr[Succ_4]| \leq \frac{3q_s}{2^{l_n}} \tag{8}$$

G_6 We define this game by considering that $\mathcal{A}$ can caculate sk based on the sk' in previous or subsequent sessions between the same participants. According to our protocol, we know all of credentials stored will be updated after a session. Also, sk is computed from $Nonce_2$, $Nonce_3$, HD_{2d}, HD_{1f} which are randomly generated and dynamically updated. And all of these parameters have nothing to do with sk' in other sessions. So we can claim that this game does not increase $\mathcal{A}$'s advantage, i.e.

$$Pr[Succ_6] = Pr[Succ_5] \tag{9}$$

The aforementioned series of games have exclude the events that $\mathcal{A}$ could win through other types of queries; consequently, $\mathcal{A}$ can only attempt to guess the hidden bit b via a $Test$ query. It is clear that:

$$Pr[Succ_6] = \frac{1}{2} \tag{10}$$

According to triangular inequality and the differences in probabilities of success from (3) to (10) for Game 0 to Game 6, we have:

$$
\begin{aligned}
Adv_{\mathcal{P}}^{SS}(\mathcal{A}) &= |Pr[Succ_0] - \frac{1}{2}| = |Pr[Succ_0] - Pr[Succ_6]| \\
&\leq |Pr[Succ_1] - Pr[Succ_0]| + |Pr[Succ_2] - Pr[Succ_1]| \\
&\quad + \cdots + |Pr[Succ_6] - Pr[Succ_5]| \\
&\leq 12\epsilon_{Hash} + 4\epsilon_{PUF} + 2\epsilon_{AES} \\
&\quad + \frac{(q_s + q_e)^2 + 5q_s}{2^{l_n+1}} + \frac{q_h^2 + 8q_s}{2^{l_h+1}} + \frac{q_s}{2^{l_r-1}}
\end{aligned} \tag{11}
$$

□

According to Theorem 1.1, we can infer that the probability of the PPT adversary $\mathcal{A}$ to break proposed mutual authentication and key exchange protocol is negligible. Therefore, we claim that our scheme is semantically secure.

2 INFORMAL SECURITY ANALYSIS AND OTHER DISCUSSIONS

This subsection presents an informal, non-mathematical analysis to discuss the security and functionality features of the proposed scheme.

2.0.1 Device Capture Attack. Assume an adversary $\mathcal{A}$ captures an IoT device and extracts all the secret information stored within it. Considering such conditions, $\mathcal{A}$ can know $\{C_d, C_f, helper_d, MIX_1\}$ and creat a msg_1 which can successfully pass the freshness and C_d, C_f verifications. Nonetheless, without a PUF, the adversary, despite knowing C_d, cannot obtain the corresponding R_d and consequently cannot derive HD_{1f} from MIX_1, which is utilized to encrypt $Nonce_2$ for identity verification. Consequently, the adversary cannot forge a correct A_2, precluding the impersonation of the device. Hence, the proposed scheme is resilient against device capture attack.

2.0.2 Fog Node Capture Attack. We now turn our discussion to the circumstances pertaining to the fog node: the FN is physically attacked by $\mathcal{A}$ and all pre-stored information is fully exposed to $\mathcal{A}$. According to the threat model, the attacker can also fully control all transmitted messages. With this capability, $\mathcal{A}$ can eavesdrop or intercept the msg_1 from an IoT Device. However, in the absence of PUF_f, the adversary is incapable of generating R_f and computing HD_{2d} from MIX_2. Consequently, $\mathcal{A}$ cannot respond to the device's challenge of encrypting $Nonce_1$, thereby preventing $\mathcal{A}$ from impersonating the fog node and completing the authentication process with the device. Therefore, we claim that our scheme also resists fog node capture attack.

2.0.3 Replay Attack. We suppose that $\mathcal{A}$ can eavesdrop and intercept all the transmitted messages in our protocol. If $\mathcal{A}$ replays these messages later, the attached timestamps are expired and fail to pass through the freshness check, resulting in the termination of the ongoing session. Therefore, our proposed scheme resists the replay attack.

2.0.4 Man-in-the-Middle Attack. According to Section **??**, $\mathcal{A}$ can inject, resend, eavesdrop and intercept the transmitted messages $\{msg_1, msg_2, msg_3\}$ during the authentication phase. However all messages are protected by secrets generated from the PUFs, rendering $\mathcal{A}$ unable to fabricate messages and impersonate one of the participants invovlved.

2.0.5 Machine Learning-based modeling Attack. An machine learning-based modeling attack is a specialized assault on real-world PUFs based on the potential correlations in their challenge-response mappings, which may be functionally related due to physical structure (e.g. linearity with MUX delays of an arbiter PUF). After $\mathcal{A}$ accummalates a large set of CRPs, $\mathcal{A}$ employs machine learning to exploit this relationship, thereby compromising the security of the PUF. However, our protocol ensures that the PUF's responses are hashed using a cryptographic function before being stored in NVM or used in communication. The high unpredictability of the hash function makes it nearly impossible to derive a relationship from the hash digest back to the challenge ($C \rightarrow h(R)$). Consequently, our protocol is resistant to machine learning-based modeling attacks.

2.0.6 PFS and Resilience against Ephemeral Secret Leakage . When there is an unexpected leakage of the secrets of the protocol, we now investigate the following two cases:

Case 1. Assume that $\mathcal{A}$ knows the random nonces $Nonce_1$, $Nonce_2$ and $Nonce_3$ unexpectedly. Without the HD_{2d} and HD_{1f}, the session key sk, established via $sk = h(Nonce_2 \| Nonce_3 \| HD_{2d} \| HD_{1f})$, cannot be computed by $\mathcal{A}$ from the disclosed nonces and transmitted messages.

Case 2. If $\mathcal{A}$ knows the secret credentials HD_{2d} and HD_{1f}, $\mathcal{A}$ could obtain $Nonce_2$ and $Nonce_3$ by decryting the intercepted msg_2 and msg_3, inferring the session key and breaching confidentiality. However, our protocol updates secret parameters per session, making the current parameters and sk unrelated to prior sessions. This ensures that past session keys and contents remain secure, thus achieving PFS.

2.0.7 Desynchronization Attack. Desynchronization attacks target protocols that dynamically update credentials by intercepting update messages, causing parameter desynchronization and invalidating subsequent authentication. In our protocol, each update message msg_2 and msg_3 includes a message digest of the updated values to ensure that the parameter updates on both sides are synchronized. Additionally, We can back up old parameters $\{C^{old}, helper^{old}, MIX^{old}\}$ while updating, to counteract authentication failure from potential desynchronization.