Open Access Library Journal
2025, Volume 12, e13943
ISSN Online: 2333-9721
ISSN Print: 2333-9705

Multi-Agent Nash Q-Learning for Node Security in Personal Privacy

Yair Oppenheim

School of Philosophy, Linguistics and Science Studies, The Lester and Sally Antin Faculty of Humanities, Tel Aviv University, Tel Aviv-Yafo, Israel
Email: yairoppen@gmail.com

How to cite this paper: Oppenheim, Y. (2025) Multi-Agent Nash Q-Learning for Node Security in Personal Privacy. *Open Access Library Journal*, **12**: e13943.
https://doi.org/10.4236/oalib.1113943

Received: July 12, 2025
Accepted: August 23, 2025
Published: August 26, 2025

Abstract

The balance between individuals' interest in protecting their private information and the interests of other entities (other individuals, confidants, Internet companies, corporations, and government agencies) has been disrupted in the age of ICTs (Information and Communication Technologies). This paper presents a multi-agent learning model based on Nash Q-learning that simulates the interaction between two competing agents—a defender (a private person) and an attacker (a Large ICT company, such as Google or Facebook)—operating in a basic component of privacy nodes with dynamic states (Safe, Attacked, Isolated). Recent work has explored Nash Q-learning in adversarial cybersecurity-for-deception contexts, demonstrating convergence properties in attacker-defender scenarios. The model enables dynamic learning of optimal defense and attack strategies while accounting for the opponent's behavior. Additionally, the paper addresses the challenges of partial observability and limited inter-agent communication, aligning with recent advances that combine graph-attention with mean-field MARL to improve scalability and decision-making under partial information. We further integrate deep learning components, including attention weighting for critical privacy components—drawing on methods such as AERIAL, which applies attention-based recurrence to handle stochastic observability in multi-agent settings. A simulation involving ten nodes demonstrates the algorithm's functionality and highlights potential directions for future research.

Subject Areas

Development Economics

Keywords

Personal Privacy, Nash Equilibrium, Nash Q-Learning Players, Deep Personal Privacy, General Personal Privacy, Multi-Agent Reinforcement Learning (MARL), Honeypot, Partial Observability, Belief Vector

1. Introduction

The central argument is that replacing the discourse on personal privacy with one on personal privacy information is justifiable, supported by several points: personal privacy information can be digitized and detached from the individual; personal privacy is fundamentally viewed as personal privacy information; and any discussion about personal privacy is, in essence, a discussion about this information. Protecting them requires a dynamic approach that models interactions between adversarial agents, such as defenders and attackers.

The basic privacy components are [1]: private space, which encompasses both physical and virtual areas where one has privacy; the body (including genetic data); the mind (thoughts, emotions, and preferences inferred from behavior); and Actions. Property information, External entities (Other players in the system of balances between the protection and violation of one's privacy), Relationships with external entities. Autonomy (An individual's capacity for self-determination or self-governance, the ability to decide for oneself, without any external interference, regarding one's own body and mind). Identity (Our identity is the product of our unique individuality) and Anonymity (Different types of anonymity are discussed (e.g., donation, medical, communication, commercial, and expression)). In my model, each basic privacy component is a node with dynamic states (Safe, Attacked, Isolated). Some of the nodes hold critical privacy information. To recognize what critical privacy information is, I define the term "deep personal privacy" as the information about one's body, mind, etc. that nobody except oneself knows, *i.e.*, the knowledge one has about oneself minus the knowledge the world (or society) has about one. General personal privacy is the information about one that is shared by one and one's confidants, but not by the world in general (other individuals, databases, enterprises, or organizations). For my current model, I assume that the components—Mind, Anonymity, and Identity—are critical nodes.

ICTs have substantially increased the collection and aggregation of information related to the basic components of privacy.

Multi-Agent Reinforcement Learning (MARL) enables agents to learn strategies in such adversarial settings by considering the policies of their opponents [2]. Recent work has explored Nash Q-learning in adversarial cybersecurity-for-deception contexts, demonstrating convergence properties in attacker-defender scenarios [3].

Nash Q-learning combines reinforcement learning with game theory to identify equilibrium strategies in stochastic games [4]. Recent studies extend this approach to incorporate deep learning in MARL, transformer-inspired communication, and partial observability mechanisms [5], aligning with recent advances that combine graph-attention with mean-field MARL to improve scalability and decision-making under partial information [6].

2. Problem Formulation

2.1. System Structure

The system consists of a finite set of nodes

$$C = \{c_1, c_2, \cdots, c_{10}\}$$

Each Node c_i represents one of the basic privacy components. these could be broken down to subcomponents. For example, we can break down the Identity component [7] into the following categories: Passport identity, Numerical identity, Attribution identity, Social function identity, and Attachment identity. Breaking down each Personal Privacy component further will help us scale up our game model.

$$M = \{\text{Safe}, \text{Attacked}, \text{Isolated}\}$$

The overall state of the system is represented by a vector:

$$s = (m_{c1}, m_{c2}, \cdots, m_{c10}) \in M^{10}$$

where c_i denotes the i-th node in the system (e.g., one of the basic personal privacy components).

$m_{ci} \in M$ is the state of node c_i, (e.g., one of {Safe, Attacked, Isolated}).

The meaning of $s \in M^n$ is that the vector $\boldsymbol{s}$ is an element of the Cartesian product of the state set M with itself n times—that is, a list of n states, one for each node.

Then the system state is

$$s = (\text{Safe}, \text{Attacked}, \text{Isolated}) \in M^i$$

This state vector serves as the input to the Nash Q-learning algorithm. All actions, decisions, and updates of Q-values (e. g. $Q_i(s, a_D, a_A)$) are conditioned on the current global state s. Changes in the state of individual nodes (e.g., due to an attack or successful defense) are immediately reflected in an updated vector s', which is then used to guide future policy updates.

2.2. Players and Actions

- The defender D selects a single node to defend:

$$a_D \in C$$

- The attacker A selects a single node to attack:

$$a_A \in C$$

2.3. Dynamics and Rewards

- If $a_D = a_A$, the defense is successful and the node remains in the Safe state.
- Otherwise, according to a defined transition policy, the attacked node A_A transitions to either the Attacked or Isolated state.

The rewards are defined as follows:

- **Defender's Reward:**

$$r_D = \begin{cases} +1 & \text{if the defense succeeds (i.e., defender} \wedge \text{attacker chose the same node)} \\ -1 & \text{otherwise} \end{cases}$$

That is:

- If the defender successfully "catches" the attacker on the same node, they re-

ceive a reward of +1.

- If the attacker selects a different node from the defender, the defender fails and receives a reward of −1.
- For the Honeypot nodes, we use:

$$r_D = \begin{cases} +1 & \text{if the defense succeeds} \\ r_A - 2 & \end{cases}$$

The defender's reward is the attacker's reward −1 (e.g. $r_A - 1$). This is because the honeypot is like a trap designed to look valuable, so attackers will target it—but it's isolated and monitored. On the other hand, if the attacker attacks the honeypot and the defender doesn't defend it, the defender's negative reward is low.

Attacker's Reward:

$$r_A = -r_D$$

The attacker's reward is the negative of the defender's reward:

- If the defense succeeds, the attacker receives −1.
- If the attack succeeds (defender missed), the attacker receives +1.

- **Interpretation**

This is a zero-sum game, meaning that one player's gain is the other player's loss. Such a setup is common in adversarial environments and is especially suitable for two-player reinforcement learning.

3. Nash Q-Learning—Multi-Agent Dynamic Learning

Nash Q-learning [6] is an algorithm that enables each player to learn Q-values while accounting for the strategy of the opposing agent, based on the concept of Nash equilibrium in stochastic games[1].

For each player I, the values table is:

$$Q_i(s, a_D, a_A) \in \{D, A\}$$

In summary, each player (defender or attacker) maintains their Q-table. This table estimates how beneficial a given pair of actions is in a particular system state. These values are updated over time through experience and learning.

The update rule is given by:

$$Q_i(s, a_D, a_A) \to \alpha * \left[r_i + \gamma * V_i(s')\right] + (1-\alpha) * Q_i(s, a_D, a_A)$$

where $V_i(s')$ is computed according to the Nash equilibrium over the stage

[1]A stochastic game (also known as a Markov game) is a type of multi-agent game that unfolds over time in a sequence of stages. At each stage:

1. The game is in a particular state.
2. Each player chooses an action.
3. Based on the current state and all players' actions.
4. Every player receives a reward.
5. The game transitions to a new state according to a probability distribution.
6. Now, a Nash equilibrium in this context is a set of strategies—one for each player—such that.
7. No player can gain a higher expected total reward by unilaterally changing their strategy, assuming the other players keep theirs fixed.

game defined by $Q_i(s')$.

Elaborate Explanation:

$V(s')$ represents the future value of the next state—that is, how "valuable" it is to be in state s' after the players have acted. It is a prediction of the expected cumulative reward from the new state, assuming that both players continue to act rationally.

In standard Q-learning (single-agent):

$$V(s') = \max_a Q(s', a)$$

This means: the agent chooses the action in s' that *maximizes* the Q-value. But in Nash Q-learning, the situation is more complex:

- There are two players, each with their own Q-table.
- The value of $V(s')$ depends on the interaction between the players' strategies.
- Each player's reward depends not only on their action, but also on the action of the other player.

How is $V(s')$ computed in Nash Q-learning?

In the next state s', each player has a Q-matrix $Q_i(s')$, which defines the rewards for all combinations of actions. The system computes a Nash equilibrium for the matrix game defined by $Q_i(s')$.

α—alphaα—is the Learning Rate

Range: $0 < \alpha \le 1$

Meaning: This parameter determines how much weight is given to the new observation (*i.e.*, the current reward) compared to the previous Q-value.

- If $\alpha = 1$

The agent learns only from the latest experience, completely ignoring past values.

- If $\alpha = 0.1$

The agent combines the previous Q-value with a small influence from the current experience. This helps maintain stability in long-term learning.

A common practical choice: $\alpha \in [0.05, 0.3]$

γ—is the Discount Factor. Range: $0 \le \gamma \le 1$

Meaning: This parameter controls how much the agent "cares about the future". It defines the trade-off between immediate rewards and future rewards.

- If $\gamma = 0$

The agent focuses only on immediate rewards—a short-sighted (greedy) behavior.

- If $\gamma = 0.9$

The agent considers long-term rewards seriously but still discounts them compared to immediate ones.

- A common practical choice: $\gamma \in [0.8, 0.99]$. This helps balance between short-term and long-term outcomes.

3.1. Partial Observability and Communication

In real-world environments, such cyber-physical security systems, players often

operate under partial information. Each player, whether defender or attacker, typically observes only a subset of the system's global state. For example, an attacker may not know whether a node is protected or isolated, while a defender may be uncertain about the attacker's intended target. This scenario is modeled as a Partially Observable Stochastic Game (POSG).

Within the framework of Partially Observable Stochastic Games (POSG), methods such as GAMFQ employ graph-attention mechanisms to improve the estimation of peer agents' actions under partial observability [8]. Similarly, approaches like MA^{2}E demonstrate that masked auto-encoders can enhance inference of the global state based on local observations.

Multi-Player Learning with Partial Observations

To address such uncertainty, the learning process incorporates the following mechanisms:

- Local Observations: Each player receives a partial observation, rather than full knowledge of the global state.
- Communication Protocols: Players can share selected information, such as action intentions or inferred states, to improve coordination and decision-making.
- History-Based Policies: Players utilize not only the current observation but also historical data (*i.e.*, sequences of observations) to infer hidden aspects of the environment.

Numerical Illustration

Consider a system with three nodes: $C = \{c_1, c_2, c_3\}$

- Let the true state of the nodes be: c_1: Safe, c_2: Attacked, c_3: Isolated.

Thus, the global state vector is: s = (Safe, Attacked, Isolated).

- Let's assume:

The defender's observation: O_D = {c_1: Safe, c_2: Unknown, c_3: Unknown}

The attacker's observation: O_A = {c_1: Unknown, c_2: Attacked, c_3: Unknown}

Each player, therefore, acts based on partial knowledge: decisions are made from, not the true O_i state.

Communication and Belief Sharing

Limited communication may be allowed. For instance:

- The defender may transmit: "I am the defending node."
- Players may share belief vectors, representing probabilistic estimates of node states.

Example:

- Belief$_D$ (c_2) = 0.7 → Defender estimates a 70% chance that c_2 is under attack.
- Belief$_A$ (c_1) = 0.9 → Attacker believes c_1 is protected with 90% certainty.

Decision-Making Under Uncertainty

The learning algorithms adapt to these conditions as follows:

- **Belief-State Q-Learning**: Q-values are updated based on belief states rather than true states.
- **History-Augmented Policies**: Policies are functions of the observation history or outputs of attention-based encoders that weigh temporal relevance.

These methods improve robustness in partially observable environments and enable agents to operate effectively even when critical information is missing or uncertain.

3.2. Attention Mechanism for Critical Nodes

In order to enhance the focus of the learning algorithm on strategically significant parts of the system, we incorporate an attention mechanism that prioritizes learning around critical nodes. These critical nodes may represent data servers, control centers, or other high-value infrastructure components within the network, where successful attacks would lead to cascading failures or significant disruptions.

In consideration of threat exposure, this is similar to the AERIAL approach, which utilizes a recurrence-attention mechanism to address uncertainty and partially observable environments. Additionally, SACHA demonstrates the use of heuristic-based attention to support multi-agent tasks operating under partial observability [9].

Motivation

In high-dimensional and partially observable environments, agents must efficiently allocate learning resources. Without guidance, agents may waste learning capacity on low-impact nodes. The attention mechanism acts as a filter, highlighting important regions of the state space based on their influence, vulnerability, or previous history.

Attention Weighting

Formally, each node $c_i \in C$ is assigned a learnable attention weight $\alpha_i \in [0,1]$, where $\sum_{i=0}^{n} \alpha_i = 1$.

These weights are updated iteratively based on feedback from Q-value gradients, threat levels, or observed system impact. The attention vector $\alpha = [\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n]$ is used to scale the contribution of each node's features during Q-value or policy updates.

For instance, the Q-update for the player may be modified as:

$$Q_i(s, a_D, a_A) \to \alpha * \left[r_i + \gamma * \left(\sum_{j=0}^{n} \alpha_j * V_i(s') \right) \right] + (1-\alpha) * Q_i(s, a_D, a_A)$$

4. Model Description: Partial Observability Zero-Sum Game with Critical Nodes

Game Setup

- The system consists of 10 nodes:

$$C = \{c_0, c_1, c_2, \cdots, c_9\}$$

c_0: Private space, c_1: Body, c_2: Mind, c_3: Actions, c_4: Property, c_5: External entities, c_6: Relationships with external entities, c_7: Autonomy, c_8: Identity, c_9: Anonymity.

Among these nodes, three are critical due to their strategic or operational importance: Critical Nodes {c_2, c_7, c_8}.

Among these nodes, two are honeypots due they are part of the General Privacy, and in the ICTs age, it is impossible to defend them: Honeypot Nodes $\{c_5, c_6\}$.

- The game is played between two players:
  - Defender: attempts to protect one node per round.
  - Attacker: attempts to attack one node per round.
- Each player can only choose one action per turn, *i.e.*, a node to protect or attack.
- The player is zero-sum: the gain of one agent is the loss of the other.

Partial Observability

- **Defender's Observation**:
  - It has reliable information about the state of critical nodes only.
  - All other nodes appear as "Unknown".
- **Attacker's Observation**:
  - Knows the status of the currently attacked node (e.g., if it succeeded previously).
  - Has uncertain beliefs about the Defender's strategy.

Beliefs

Each agent maintains belief vectors about the likelihood of the other's actions:

- Defender's belief vector $\text{Belief}^D(i)$: probability that node c_i is being attacked.
- Attacker's belief vector $\text{Belief}^A(i)$: probability that node c_i is being defended.

These belief vectors are initialized with a bias towards critical nodes.

Payoff Matrices Initialization

Let:

$D(i,I)$ be the payoff to the Defender when defending node i Ind the attacker attacks node j (see Table 1).

$A(I,I)$ be the payoff to the Attacker when attacking node i Ind the defender defends node j (see Table 2).

Table 1. Defender's initial payoff matrix Q_def (10 × 10).

Case	Value
$i == j$	+1
$i \neq j$ and $j \in$ Critical	**−2**
$i \neq j$ and $j \notin$ Critical	−1

Table 2. Attacker's initial payoff matrix Q_att (10 × 10).

Case	Value
$i == j$	**−1**
$i \neq j$ and $i \in$ Critical	**+2**
$i \neq j$ and $i \notin$ Critical	+1

The learning parameters are in Table 3.

Table 3. Learning parameters.

Parameter	Value
Learning Rate (α)	0.2
Discount Factor (γ)	0.85

Each Q-table is updated during the game using the Q-learning rule with partial observability, incorporating beliefs into the future reward estimation:

$$Q(s,a_D,a_A) \to (1-\alpha)\cdot Q(s,a_D,a_A) + \alpha\cdot(r+\gamma\cdot \text{Belief})$$

Defender's Belief Vector Belief^D.

This is a vector of size 10.

The Defender Belief Vector $\text{Belief}^D = [b^{0D}, b^{1D}, \ldots, b^{9D}]$.

This is a vector of size 10: Each element $b^{iD} \in [0,1]$ represents the Defender's estimated probability that node c_i is under attack in the current round. However, the Defender's prior assumption is that critical nodes are most likely to be attacked, even without direct evidence. Thus $b^{2D} = 0.8$, $b^{7D} = 0.9$.

Attacker's Belief Vector $\text{Belief}^A = [b^{0A}, b^{1A}, \ldots, b^{9A}]$.

This is a vector of size 10: Each $b^{iA} \in [0,1]$ represents the Attacker's estimated probability that node c_i is being defended in this round. However, the Attacker's belief that critical nodes are highly likely to be defended, and that the Defender is likely to focus on those. Thus $b^{2A} = 0.7$, $b^{7A} = 0.9$, $b^{8D} = 0.8$.

Defender's Belief Update Rule

After observing the attacked node (true info revealed after each episode), the Defender updates their belief:

- b^{iD} be the belief that node i will be attacked.
- attacked_node is the node attacked.

Let $\alpha_b = 0.1$

For each $0 \le i \le 9$:

if i == attacked_node:

$\text{Belief}^{D[i]} = \text{Belief}^{D[i]} + \alpha_b^{*}$ (1 - −elief$^{D[i]}$) − Means move toward 1

else:

$\text{Belief}^{D[i]}$] = $\text{Belief}^{D[i]}$ i] + α_b^{*} (0 - −elief$^{D[i]}$) means move toward 0

This pushes the belief on the true attack node closer to 1 and reduces belief in others.

Attacker's Belief Update Rule

After learning which node was defended, the Attacker updates its belief:

Let:

- b^{iA} be the belief that node i will be defended.
- defended node is the node defended.

Update Rule:

Let $\alpha_b = 0.1$

For each $0 \leq i \leq 9$:

if i == defended_node:

Belief$^{A[i]}$ = Belief$^{A[i]}$ + α_b* (1 - −elief$^{A[i]}$) # increase toward 1

else:

Belief$^{A[i]}$ = Belief$^{A[i]}$ + α_b* (0 - −elief$^{A[i]}$) # decrease toward 0

This moves the belief toward the actual defense choice.

ε-greedy (epsilon-greedy) function

The ε-greedy (epsilon-greedy) function is a key part of the agent's decision-making in reinforcement learning. It controls the balance between:

- Exploration—trying new actions to discover potentially better outcomes.
- Exploitation—choosing the best-known action based on experience.

The ε-greedy idea:

- At every decision point, the agent does the following;
- With probability ε (epsilon);
- It explores—picks a random action, even if it's not currently the best one;
- With probability 1 - −;
- It exploits—picks the best-known action, *i.e.*, the one with the highest Q-value.

Why is this important?

- Without exploration ($\varepsilon > 0$), the agent might;
- Get stuck always choosing the same action (even if it's suboptimal);
- Miss better options that it hasn't tried yet;
- Without exploitation ($\varepsilon = 1$ all the time), the agent;
- Learns too slowly, acting randomly all the time.

In most implementations (including yours), ε decreases over time (called *epsilon decay*):

- epsilon = max(EPSILON_MIN, epsilon * EPSILON_DECAY);
- This means;
- In early episodes, the agent explores more (high ε, like 0.9);
- In later episodes, the agent exploits more (low ε, like 0.02);
- This encourages broad learning first, and focused optimization later.

5. Python Model Simulation: Nash Q-Learning—Multi-Agent Dynamic Learning

To demonstrate the feasibility of the model, I ran a simulation of the model implemented in a Python program.

Game parameters: Num nodes = 10, num episodes = 200,critical_nodes = {2, 7, 8}, honeypot nodes = {5, 6}, alpha = 0.2, gamma = 0.85, alpha_b = 0.2, epsilon_start = 0.9, epsilon decay = 0.95, epsilon_min = 0.02, honeypot_reward[2] = 3.

The Initial Defender matrices' payments are in Table 4.

The Initial Attacker matrices' payments are in Table 5.

[2]This is an arbitrary starting value that the Q-Learning algorithm updates in any episode according to the reward optimization.

Table 4. Initial defender matrices payments.

Defender	Private	Body	Mind	Actions	Property	ExtEntities	RelWithEx	Autonomy	Identity	Anonymity
Private	1	−1	−2	−1	−1	−1	−1	−2	−2	−1
Body	−1	1	−2	−1	−1	−1	−1	−2	−2	−1
Mind	−1	−1	1	−1	−1	−1	−1	−2	−2	−1
Actions	−1	−1	−2	1	−1	−1	−1	−2	−2	−1
Property	−1	−1	−2	−1	1	−1	−1	−2	−2	−1
ExtEntities	−1	−1	−2	−1	−1	1	−1	−2	−2	−1
RelWithEx	−1	−1	−2	−1	−1	−1	1	−2	−2	−1
Autonomy	−1	−1	−2	−1	−1	−1	−1	1	−2	−1
Identity	−1	−1	−2	−1	−1	−1	−1	−2	1	−1
Anonymity	−1	−1	−2	−1	−1	−1	−1	−2	−2	1

Table 5. Initial attacker matrices payments.

Attacker	Private	Body	Mind	Actions	Property	ExtEntities	RelWithEx	Autonomy	Identity	Anonymity
Private	1	1	1	1	1	1	1	1	−2	−1
Body	−1	−1	1	1	1	1	1	1	1	1
Mind	2	2	−1	2	2	2	2	2	2	2
Actions	1	1	1	−1	1	1	1	1	1	1
Property	1	1	1	1	−1	1	1	1	1	1
ExtEntities	3	3	3	3	3	−3	3	3	3	3
RelWithEx	3	3	3	3	3	3	−3	3	2	3
Autonomy	2	2	2	2	2	2	2	−1	−2	2
Identity	2	2	2	2	2	2	2	2	−1	2
Anonymity	1	1	1	1	1	1	1	−2	1	−1

Simulation Results

Game parameters changes

Episode 0: $\varepsilon = 0.9 \rightarrow$ 90% chance of random move

Episode 50: $\varepsilon \approx 0.066$

Episode 150+: $\varepsilon \approx 0.02$ (minimum)

The Belief_Defender_over_time (see Table 6).

Table 6. Belief_Defender_over_time.

	Private	Body	Mind	Actions	Property	ExtEntities	RelWithEx	Autonomy	Identity	Anonymity
Episode 0	0.28	0.08	0.72	0.08	0.08	0.4	0.4	0.56	0.64	0.08
Episode 50	0.014068	0.000129	0.00579	2.86E−05	1.78E−06	0.897978	3.55E−05	0.081932	7.92E−05	8.75E−06
Episode 100	2.01E−07	1.83E−09	8.26E−08	4.09E−10	2.55E−11	0.871999	5.07E−10	1.17E−06	0.128	1.25E−10
Episode 150	2.87E−12	2.62E−14	1.18E−12	5.83E−15	3.63E−16	0.9999998	7.23E−15	1.67E−11	1.83E−06	1.78E−15

The Belief_Attacker_over_time (see Table 7).

Table 7. Belief_Attackerr_over_time.

	Private	Body	Mind	Actions	Property	ExtEntities	RelWithEx	Autonomy	Identity	Anonymity
episode 0	0.08	0.08	0.56	0.08	0.08	0.24	0.44	0.48	0.48	0.08
episode 50	0.996008	0.000762	4.12E−05	0.002307	5.60E−06	0.00013	0.000266	5.20E−05	5.88E−05	0.000392
episode 100	1	1.09E−08	5.88E−10	3.29E−08	8.00E−11	1.86E−09	3.80E−09	7.42E−10	8.39E−10	5.59E−09
episode 150	1	1.55E−13	8.40E−15	4.70E−13	1.14E−15	2.65E−14	5.42E−14	1.06E−14	1.20E−14	7.98E−14

The Max_Q_over_time (see Table 8).

Table 8. Max_Q_over_time.

		Defender max Q	Attacker max Q	Epsilon
Episode	0	1	3.00	0.855
Episode	50	7.7	12.97	0.066
Episode	100	12.07	16.42	0.02
Episode	150	13.06	19.66	0.02

Honeypot Trap Results

The Attacker attacked the Honeypot (c_5, c_6) 184 times out of 200 episodes, which means 92%.

The Defender defended the Honeypot (c_5, c_6) 4 times out of 200 episodes, which means 2%.

In only 2 episodes, Defender defended one of the Honeypots (c_5, c_6), while Attacker attacked one of the Honeypots (c_5, c_6).

Critical Nodes Results

The Attacker attacked the Critical nodes (c_2, c_7, c_8) 8 times out of 200 episodes, which means 4%.

The Defender defended the Critical nodes (c_2, c_7, c_8) 5 times out of 200 episodes, which means 2.5%.

In only 2 episodes, Defender defended one of the Critical nodes (c_2, c_7, c_8) while Attacker attacked one of the Critical nodes (c_2, c_7, c_8).

The Best Episode for Defender and the Attacker

Episode 199, DefenderNode c_0, AttackerNode c_5, Reward_Defender 2, Reward_Attacker 3, NashValue_Defender 13.2693, NashValue_Attacker 19.9201.

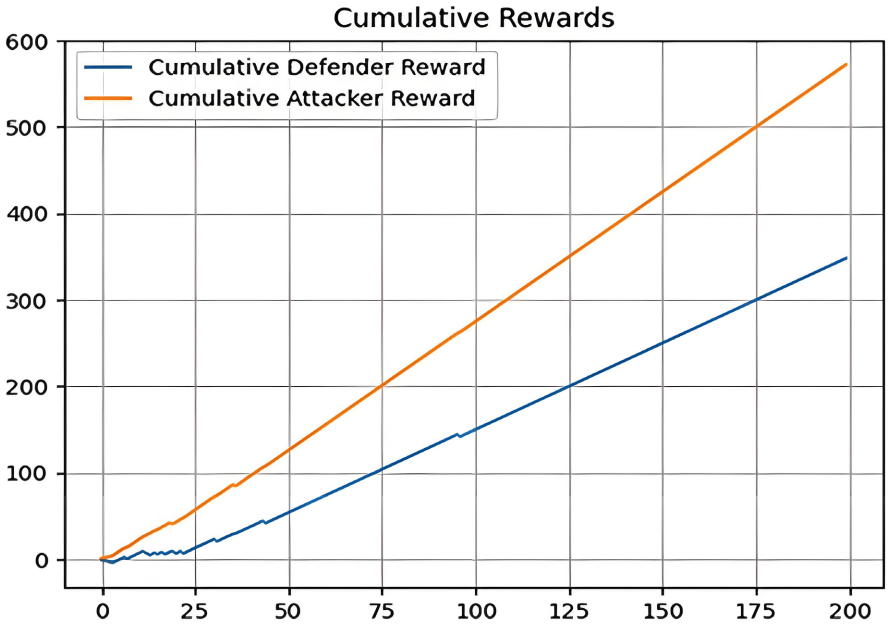
Here are the Results of **Graphs 1-5**.

Result Simulation Conclusions

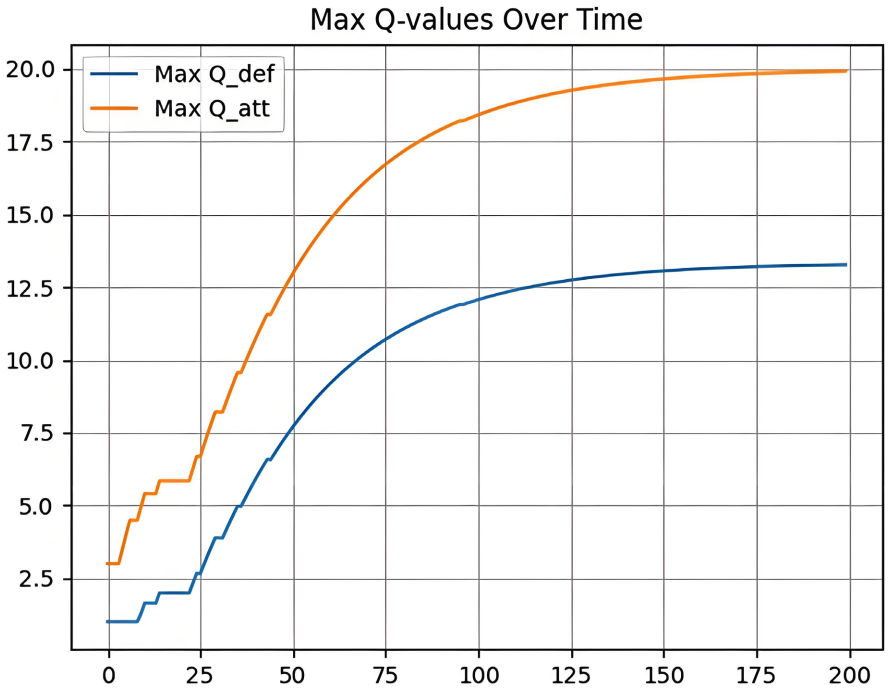
- The Attacker rewards are greater than the Defender rewards, but they are almost the same cumulative rate. This fact aligns with the reality that big ITC companies make enormous profits from personal privacy information.
- The Max Q-values over the time of the Attacker is greater than the Defender. However, both Max Q-values have almost the same rate. Both almost reached

the steady state from episode 150, and this point onward. The players have a similar learning curve that reaches a steady point on episode 150.

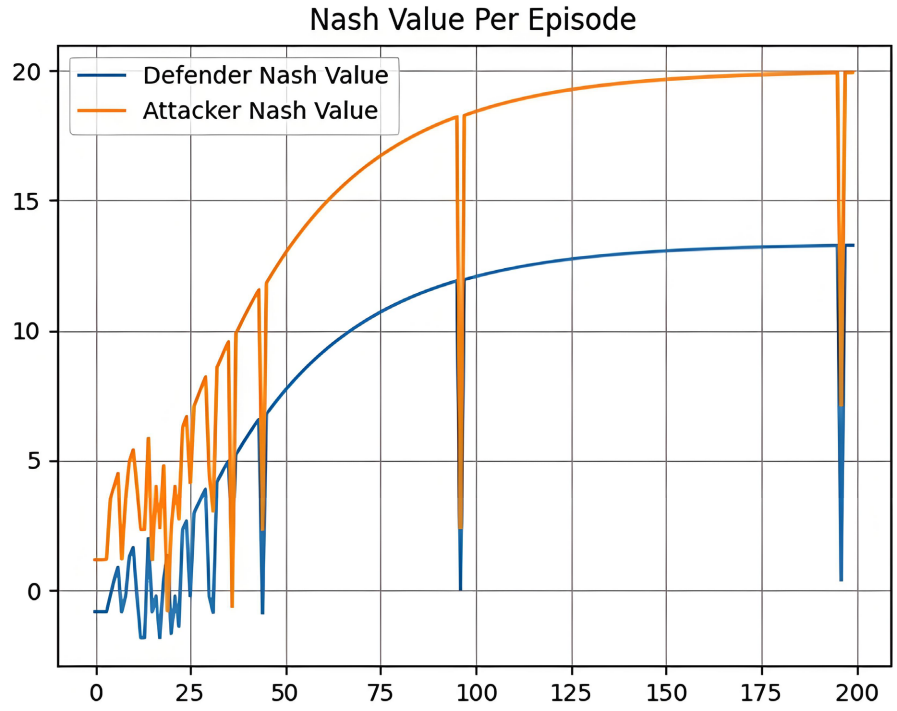
- The Nash value per episode is better for the Attacker than the Defender. This fact aligns with the reality that big ITC companies make enormous profits from personal privacy information. However, the best Nash value for both players occurred on the last episode (199).
- For graph Final Attacker Q-table.



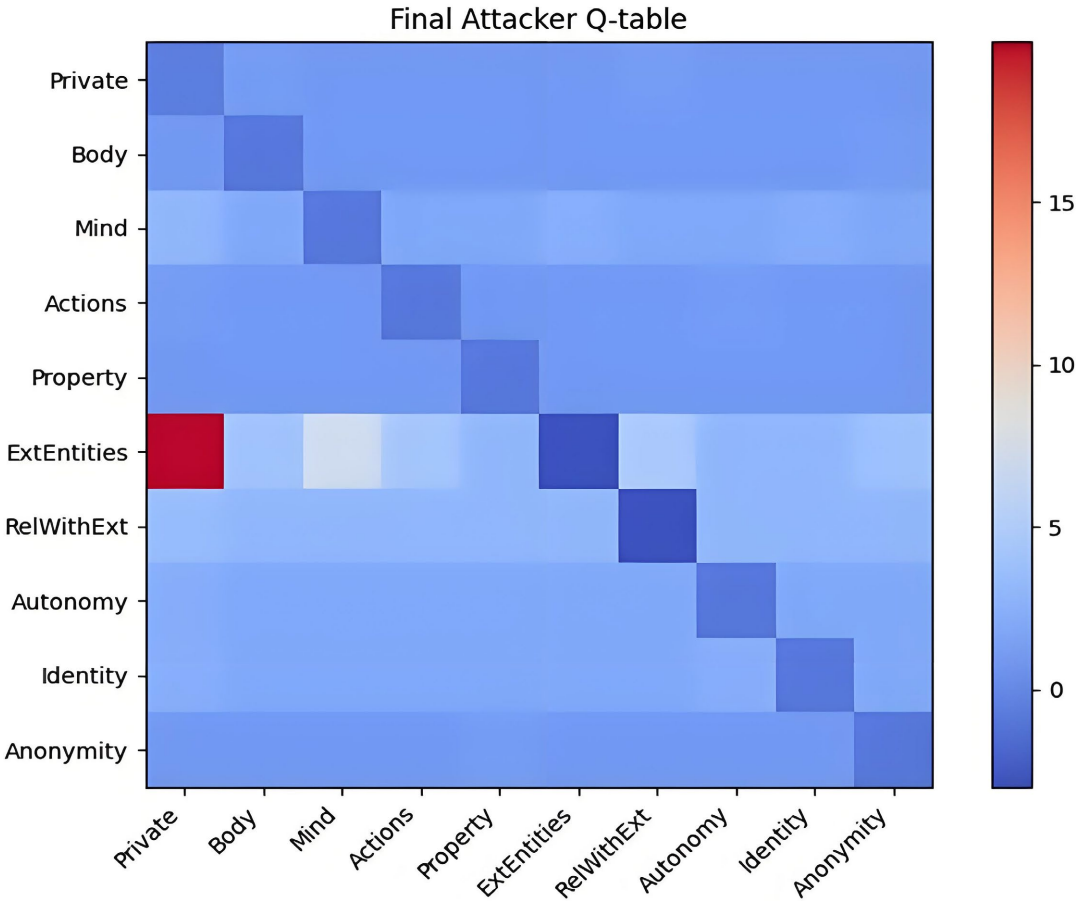
Graph 1. Cumulative rewards.



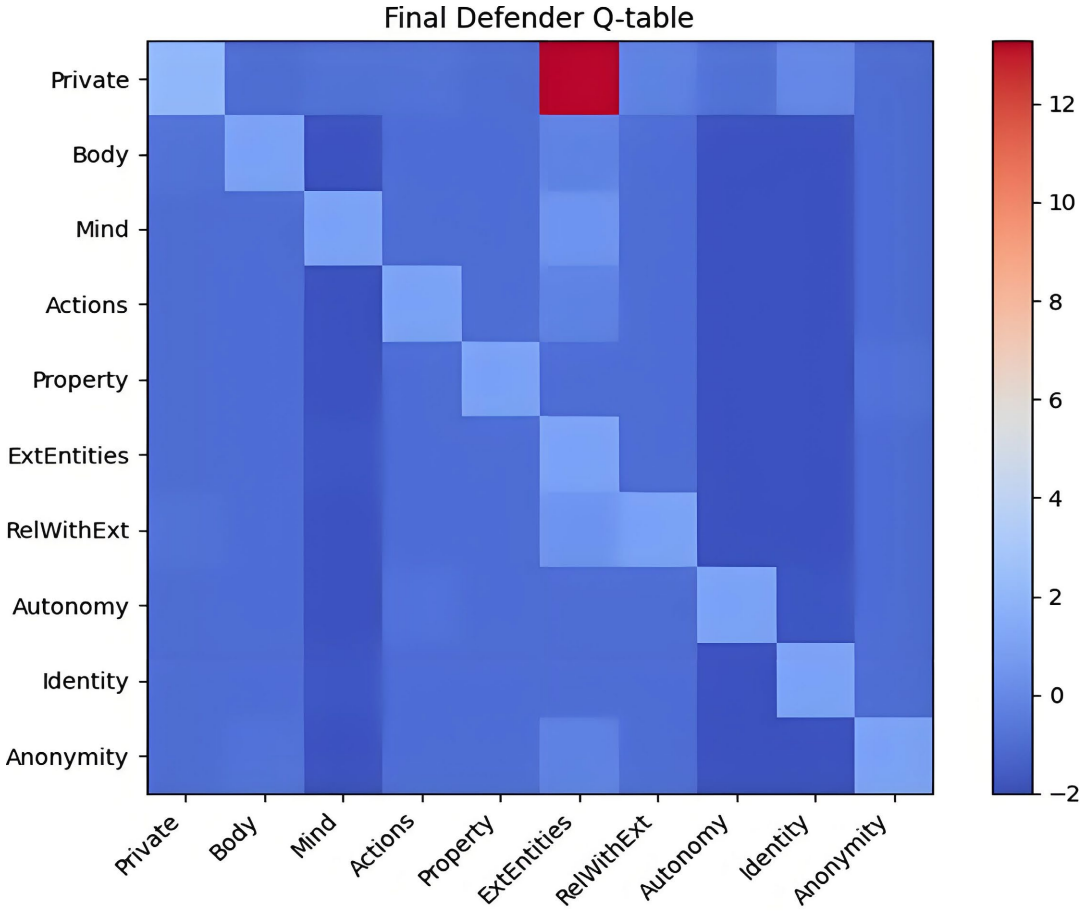
Graph 2. Max Q-values over time.



Graph 3. Nash value per episode.



Graph 4. Final attacker Q-table.



Graph 5. Final defender Q-table.

Axes:

Y-axis (rows): Represents the node the attacker is acting *from* (*i.e.*, the current state).

X-axis (columns): Represents the node the attacker is *targeting* (*i.e.*, the action).

Both axes are labeled with 10 conceptual nodes: Private, Body, Mind, ..., Anonymity.

Colors:

Redder cells = higher Q-values = high expected reward for attacking that node from that state.

Blue cells = lower or even negative Q-values = low or undesirable expected outcomes.

Key Observations:

The strongest red cell appears on the row for ExtEntities, targeting some node—meaning the attacker learned that targeting that node from ExtEntities gives them very high reward.

Diagonal cells (where action = state) are usually avoided in attacker policies—unless they represent critical or honeypot nodes.

- For graph Final Defender Q-table

Axes:

Y-axis: Represents the node the defender is choosing to monitor or defend.

X-axis: Represents where the attack occurred.

Colors:

Red cells = higher Q-values = the defender learned that defending this node (against that attack) yields high reward.

Dark blue cells = low or negative expected utility.

Key Observations:

A strong red value appears when defending ExtEntities against some specific node (attack).

The defender seems to have concentrated most learning value on specific nodes (e.g., ExtEntities, RelWithExt)—possibly due to being critical nodes or frequent attack targets.

- The conclusion regarding the Honeypot trap results is that they achieved the goal of being a trap for the attacker. We see in the simulation that the Attacker concentrates his attacks on the Honeypot instead of on the critical nodes.
- The similar optimization outcomes between attack and defense align with findings from recent surveys on the applications of MARL in cybersecurity.

6. Conclusions

This paper introduces a multi-agent reinforcement learning framework using Nash Q-learning to simulate the strategic interaction between two adversarial agents in the context of personal privacy protection. The model involves a Defender (representing the individual) and an Attacker (representing a powerful ICT corporation such as Google or Facebook) competing over a network of privacy-related nodes, each representing a fundamental component of personal privacy (e.g., mind, identity, anonymity). Each node can be in one of three states: *Safe, Attacked, or Isolated*.

The model:

- Formulates a zero-sum stochastic game, where each agent selects one node per turn to either attack or defend.
- Implements Nash Q-learning, which allows agents to learn optimal strategies while accounting for the opponent's behavior and adapting to changing system dynamics.
- Incorporates partial observability, with agents maintaining belief vectors about the opponent's likely actions.
- Enhances learning through attention weighting, prioritizing critical nodes like *Mind, Identity,* and *Autonomy*.
- Introduces honeypot nodes to divert attacks away from critical components.

A Python simulation with ten nodes over 200 episodes demonstrates the system's behavior. Results indicate:

- The attacker predominantly targeted honeypot nodes (92%), suggesting the traps were effective.

- Both agents converged to near-stable strategies after ~150 episodes.
- Nash values and Q-values showed the attacker generally gained higher rewards, aligning with real-world asymmetries in power between individuals and tech corporations.

Ultimately, the research highlights how game-theoretic multi-agent reinforcement learning can model real-world privacy dynamics, offering insights for developing adaptive and strategic privacy protection mechanisms.

Can the model presented here be implemented by individual human beings? The answer to this question depends on their motivation to protect their privacy and on whether, assuming such motivation exists, they possess the legal and technological tools necessary to safeguard their personal privacy. As of today, the answer to both questions is negative.

Regarding the motivation, not everyone agrees that privacy is such a great matter of concern; some scholars believe the current preoccupation with privacy to be largely an expression of older Westerners' paranoia, bringing as evidence the ease with which people share personal information on social media. According to Calvin Gottlieb, "most people, when other interests are at stake, do not care enough about privacy to value it" [10].

Regarding protecting personal privacy by legal regulation is under ongoing discourse. more and more individuals and authorities are beginning to realize the urgent need for regulation and supervision in this area. Some major examples include Mark Zuckerberg's testimony before the US Congress, as well as the European Union's new General Data Protection Regulation (GDPR) [11] designed to protect the personal information of individuals inside Europe and to prevent improper use of personal data. However, this approach hasn't succeeded yet in achieving real privacy protection.

Regarding protecting personal privacy with technology. In this race by definition the individuals can't compete with the big ICTs companies.

Changing this situation requires a shift in the prevailing paradigm regarding personal privacy.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Oppenheim, Y. (2024) Personal Privacy in the Age of the Internet. Spines, 140-154.

[2] Phan, T., Ritz, F., Altmann, P., *et al.* (2023) Attention Based Recurrence for Multi Agent Reinforcement Learning under Stochastic Partial Observability. arXiv:2301.01649 https://doi.org/10.48550/arXiv.2301.01649

[3] Kong, G., Chen, F., Yang, X., Cheng, G., Zhang, S. and He, W. (2023) Optimal Deception Asset Deployment in Cybersecurity: A Nash Q-Learning Approach in Multi-Agent Stochastic Games. *Applied Sciences*, **14**, 357. https://doi.org/10.3390/app14010357

[4] Lin, Q. and Ma, H. (2023) SACHA: Soft Actor-Critic with Heuristic-Based Attention

for Partially Observable Multi-Agent Path Finding. arXiv:2307.02691.

[5] Yang, M., Liu, G., Zhou, Z. and Wang, J. (2023) Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms. *Drones*, **7**, Article 476. https://doi.org/10.3390/drones7070476

[6] Kang, S., *et al.* (2025) MA^{2}E: Addressing Partial Observability in Multi-Agent Reinforcement Learning with Masked Auto-Encoder. *ICLR*.

[7] Oppenheim, Y. (2025) A Metric for Calculating the Extent of Non-Knowledge (Level) of Personal Privacy. *Open Access Library Journal*, **12**, e13554 https://doi.org/10.4236/oalib.1113554

[8] Finistrella, S., Mariani, S. and Zambonelli, F. (2025) Multi-Agent Reinforcement Learning for Cybersecurity: Classification and Survey. *Intelligent Systems with Applications*, **26**, 200495. https://doi.org/10.1016/j.iswa.2025.200495

[9] Busoniu, L., Babuska, R. and De Schutter, B. (2008) Multi-Player Reinforcement Learning: A Survey. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, **38**, 156-172. https://doi.org/10.1109/TSMCC.2007.913919

[10] Gottlieb, C.C. (1996) Privacy: A Concept Whose Time Has Come and Gone. In: Lyon, D. and Zuriek, E., Eds., *Computer*, *Surveillance*, *and Privacy*, University of Minnesota Press, 156.

[11] Massey, S.R. (2017) The Ultimate GDPR Practitioner Guide: Demystifying Privacy & Data Protection. Fox Red Risk, 1.