https://doi.org/10.31891/csit-2025-4-7
UDC 004.75

BOKHONKO Oleksandr, ATAMANIUK Olha
Khmelnytskyi National University

METHOD FOR SYNTHESIS OF A SCALABLE ARCHITECTURE OF A DISTRIBUTED COMPUTER SYSTEMS, RESISTANT TO SOCIAL ENGINEERING ATTACKS

Social engineering continues to be one of the most dangerous classes of threats for modern distributed IT systems, where event processing, resource access, and protection mechanisms are performed on a large number of heterogeneous nodes. The growth of the scale of architectures, the emergence of multi-channel interaction scenarios, remote users, and a high level of dynamism create challenges for the synthesis of systems that are able to maintain resistance to social engineering attacks. The study proposes methods and tools for the synthesis of distributed systems focused on ensuring structural, behavioral, and functional resistance to such attacks.

The basis of the approach is the use of a population multi-agent mean-field model, which allows considering a large number of nodes as a coordinated system of local detectors interacting through an aggregated state space. This makes it possible to describe the impact of attacks not on individual components, but on the entire distributed system as a whole, and to evaluate its response through integrated risk and resilience indicators. The study forms a generalized model of a distributed system, defines the roles of different types of nodes, protections and interaction channels, and also describes the methodology for architecture synthesis, which includes the classification of local actions, coordination mechanisms and evaluation criteria.

Special attention is paid to the integration of protective measures - deception components, multifactor authentication, filtering and segmentation mechanisms - into the structure of a distributed system. Methods for optimizing the distribution of these measures at different levels of the architecture are proposed in accordance with the dynamics of the mean field and target requirements for stability. An iterative approach to architecture synthesis is developed, which combines the adaptation of local node strategies with the tuning of global system parameters.

The results demonstrate that the use of the mean field concept allows to ensure scalability of solutions, consistency of node behavior, and also to increase the ability of a distributed system to counteract social engineering attacks in conditions of uncertainty and high variability of scenarios. The methodology can be used for the design, improvement and engineering synthesis of real distributed IT architectures operating in critical environments.

Keywords: distributed systems; social engineering attacks; Distributed computer system resilience; architecture synthesis; multi-agent models; middle ground.

БОХОНЬКО Олександр, АТАМАНЮК Ольга
Хмельницький національний університет

МЕТОД СИНТЕЗУ МАСШТАБОВАНОЇ АРХІТЕКТУРИ РОЗПОДІЛЕНИХ КОМПЬЮТЕРНИХ СИСТЕМ, СТІЙКИХ ДО АТАК СОЦІАЛЬНОЇ ІНЖЕНЕРІЇ

Соціальна інженерія залишається одним із найнебезпечніших класів загроз для сучасних розподілених комп'ютерних систем, у яких оброблення подій, доступ до ресурсів та механізми захисту виконуються на великій кількості гетерогенних вузлів. Зростання масштабів архітектур, поява багатоканальних сценаріїв взаємодії, віддалених користувачів і високий рівень динамічності створюють додаткові виклики для синтезу систем, здатних підтримувати стійкість до атак соціальної інженерії. У дослідженні запропоновано методи та засоби синтезу розподілених систем, орієнтованих на забезпечення структурної, поведінкової та функціональної стійкості до таких атак.

Основою підходу є використання популяційної багатоагентної моделі середнього поля, яка дає змогу розглядати велику кількість вузлів як узгоджену систему локальних детекторів, що взаємодіють через агрегований простір станів. Це дозволяє описувати вплив атак не на окремі компоненти, а на всю розподілену систему загалом, а також оцінювати її реакцію через інтегральні індикатори ризику та резильєнтності. У роботі формуються узагальнена модель розподіленої системи, визначаються ролі різних типів вузлів, засобів захисту та каналів взаємодії, а також описується методологія синтезу архітектури, що охоплює класифікацію локальних дій, механізми координації та критерії оцінювання.

Особливу увагу приділено інтеграції захисних механізмів компонентів обману, багатофакторної автентифікації, механізмів фільтрації та сегментації у структуру розподіленої системи. Запропоновано методи оптимізації розподілу цих засобів на різних рівнях архітектури відповідно до динаміки середнього поля та цільових вимог до стійкості. Розроблено ітеративний підхід до синтезу архітектури, який поєднує адаптацію локальних стратегій вузлів із налаштуванням глобальних параметрів системи.

Отримані результати демонструють, що використання концепції середнього поля забезпечує масштабованість рішень, узгодженість поведінки вузлів, а також підвищує здатність розподіленої системи протидіяти атакам соціальної інженерії в умовах невизначеності та високої варіативності сценаріїв.

Запропонований метод може бути використаний для проектування, модернізації та інженерного синтезу реальних розподілених комп'ютерних систем, що функціонують у критичних середовищах.

Ключові слова: розподілені системи; атаки соціальної інженерії; стійкість комп'ютерних систем; синтез архітектури; багатоагентні моделі; середнє поле.

Received / Стаття надійшла до редакції 14.11.2025
Accepted / Прийнята до друку 14.12.2025

Introduction

Distributed computer system are becoming the basis of modern digital infrastructures, but their vulnerability to social engineering attacks increases with scale and heterogeneity. The multi-channel nature of such attacks, the variability of user behavior, and the large volume of interactions between nodes complicate the application of traditional protection methods focused on local or centralized mechanisms. To ensure resilience, new engineering approaches are needed that take into account the collective behavior of the system and allow for a coordinated response to complex social engineering influences. The proposed method for synthesizing a scalable architecture is based on a population multi-agent model using a mean-field description that ensures the coordination of local node decisions with global requirements for resilience and risk.

Related works

In modern distributed computer systems, the key challenge remains the creation of scalable engineering solutions to counter social engineering (SI) attacks. The majority of scientific works are focused on methods for local detection or analysis of individual interaction channels, which limits their application in multi-node infrastructures. In [1], [2], AI-oriented methods for detecting SI attacks using behavioral, textual, and contextual features are considered, but they operate at the level of individual services and do not involve coordination between multiple nodes. In [3], [19], engineering implementations of email detectors based on hybrid and deep models (BERT+LSTM, adaptive optimizers) are presented, which confirms the effectiveness of deep networks in the email channel, but does not solve the problem of large-scale deployment in a distributed CS. Additionally, in [17], [22], ML and DL models were developed to detect smishing and phishing text messages, which remain isolated services without a unified integration architecture.

SI attack models developed in [4] demonstrate the possibility of engineering signatures for spam, spear-phishing and trojan-mails, which can be used as the basis for local algorithms of detector agents. Methods for analyzing URLs, web traffic and pages using convolutional, ensemble, hybrid and fuzzy deep neural networks are proposed in [5], [6], [16], [18], [20], [21]. They provide high accuracy in their domains, but remain autonomous components without coordination mechanisms between numerous nodes. Separate solutions for mobile and text channels, in particular hybrid DL smishing detection models [17] and multi-class phishing message classifiers [22], are also integrated into the infrastructure as single services, without providing scalable interaction of detectors at the distributed CS level.

In [7], a cloud-based method for processing sequential user interactions is presented, which allows detecting phishing scenarios in services with distributed event queues. Works [8] and [9] propose multimodal systems for detecting voice phishing and phishing websites, where features of audio, text, visual components and HTML structures are combined into a single decision-making algorithm. Although such systems show high engineering efficiency, they are implemented as separate modules with fixed logic. In [10], [11], [16], [18], optimization and adaptive ML models (hybrid DL–optimization structures, stacked autoencoders, dynamic reconfiguration of deep networks) are considered to improve the quality of detection and resistance to new attacks. Such methods demonstrate the importance of adapting to changing phishing patterns, but do not define an architecture in which multiple detectors could interact with each other in a large-scale environment. The work [12] proposes a system framework in which phishing detection is integrated with quantitative risk assessment, which is an important property for building the upper level of control in scalable distributed CSs.

The most relevant to the problem of synthesizing a scalable architecture are the results in the field of multi-agent, distributed and hierarchical cyber defense systems. In [13], [23], [24], the analysis and specific implementations of multi-agent RL and DRL models for network security, IoT surveillance and security key generation are presented, with an emphasis on decentralized execution, cooperation, policy scaling and coordination of agent actions in heterogeneous environments. In [14], a hierarchical MARL architecture is implemented, where the master agent distributes the task among specialized subagents, which proves the effectiveness of decomposition of complex threats in large infrastructures. In [15], [25], the organization of intelligent agents and CYBERSHIELD simulation environments for training AI in cyber defense is considered, with a focus on communication, routing, local decision-making and competitive learning in multi-node information systems.

Thus, the existing works [1] – [25] demonstrate significant progress in modeling and detecting SI attacks within individual channels or individual ML modules and multi-agent systems. However, the identified limitations - the lack of coordination between detectors, the lack of population organization of agents, the lack of a global control function and the impossibility of adaptive scaling in a distributed CS - form a scientific gap that should be filled by a method for synthesizing a scalable architecture of a distributed CS with multi-agent mechanisms of resistance to social engineering.

Distributed systems synthesis method resistant to social engineering attacks, based on population mean field modeling

The method of synthesizing distributed systems resistant to social engineering attacks based on population mean-field modeling is based on the idea of the Distributed computer system as a large set of detector agents whose behavior is coordinated through a generalized population distribution. This approach allows for the formation of

scalable interaction and decision-making strategies that ensure the stability of the system even in conditions of incomplete, noisy and dynamically changing information.

Fundamentals of the method for synthesizing a scalable architecture of a distributed CS resistant to social engineering attacks. The synthesis is based on considering distributed computer system as a set of interacting nodes operating in a multi-channel environment under the influence of social engineering threats. The need for distributed interaction of detectors capable of adaptively responding to changes in the context and coordinating their actions without a centralized control point is shown. This formulation allows you to avoid bottlenecks and increases the stability of the system as its scale increases.

Formalization of a population multi-agent system. A multi-agent system is represented by a set of agents, each of which corresponds to a certain type of node and has its own states, permissible actions and transition rules. The structure of populations, their interaction through communication channels and the mechanism of influence of social engineering attacks on the behavior of individual agents are determined. The formalization creates the basis for an aggregated description of the behavior of a large group of nodes.

Representation of a population of MAC agents. The population is described through distributions of agent states and characteristics, which provides a compact representation of the behavior of the system as a whole. This approach eliminates the need to analyze each node separately and avoids exponential growth in complexity. The use of aggregated population structures is key to the scalability of further architecture synthesis.

Modeling the extended state of a representative agent. The extended state includes local node parameters and generalized population characteristics formed based on the mean field. Such a state reflects both individual risk and collective context, allowing to build solutions consistent with the global dynamics of the system. The representative agent becomes an analytical model of any node in a large infrastructure.

Definition of the reward function of the architecture synthesis task. The reward function combines local performance indicators (detection accuracy, operational costs) and global risk factors at the population level. Its structure sets priorities between detection accuracy, the number of false decisions, and the system's resistance to complex attacks. This allows shaping the behavior of agents in a way that reconciles local and global interests.

Synthesis of local policy of a representative agent. Local policy determines the optimal action of the agent in response to its own extended state. An approach is considered in which the policy is formed by minimizing risk and balancing between conflicting detector actions: warning, blocking, escalation, or further observation. Local policy is a key component of the adaptive behavior of the system.

Formation of optimal policy of a representative agent. The optimal policy is obtained by self-consistent behavior of the representative agent with population dynamics. The mutual influence of local strategies on the overall state of the system is analyzed, conditions of stability and equilibrium are formed. The result is a policy that minimizes the global risk of the system and ensures resistance to complex social engineering attacks. Architectural parameters in the context of the mean field. Architectural parameters density of deception nodes, level of multifactor authentication, network segmentation, topology of communications are integrated into the model of the mean field as elements that affect the population context. It is shown how changes in the engineering configuration change the behavior of agents and increase the resilience of the entire system. Synthesis of a scalable architecture. A holistic solution is formed regarding the set of architectural parameters, agent strategies, and population configuration of the system. The synthesis ensures risk minimization at the level of the entire infrastructure and creates a distributed architecture that can maintain stability under high load, a variety of attacks, and unpredictability of user behavior.

Scalability analysis of mean-field architectures and policies. The ε – Nash property means that an individual agent gains little from unilaterally deviating from the mean-field policy while all others follow it. If this error decreases with increasing number of agents, then the mean-field policy becomes close to the true equilibrium inherent in systems with large populations.

Fundamentals of the method for synthesizing a scalable architecture resistant to social engineering attacks

The scalable architecture synthesis method is based on representing the Distributed computer system as a population of detector agents attached to each node. Nodes are grouped by type (workstations, mail gateways, web proxies, servers, network segments, mobile devices). For each type, a local state and action space is defined that takes into account the level of risk, the interaction channel (email, web, social networks, voice), available features, and response modes (blocking, escalation, monitoring). The next model moves to a mean-field representation, in which instead of keeping track of the states of all nodes, a compact description in the form of class-wise state distributions is used. This description aggregates the fraction of high-risk nodes, attack intensity across channels, average loads, and integrated risk measures. The global state of the multi-node system is replaced by a low-dimensional vector, which eliminates the exponential growth of the state space. Next, a problem is formulated for a representative agent that sees its own local state and the context of the mean field, which reflects the aggregated behavior of the population. Its dynamics describes the change of the local state under the influence of attacks, the spread of compromises and the reaction of coordination modules. The evolution of the entire population is given by a master equation, in which the distributions are updated according to the policy of the detectors. Thus, local solutions are combined with the global behavior through the context of the mean field. The reward function evaluates the quality of the local solution: correct blocking is encouraged, missed attacks are penalized, unnecessary

escalations are punished for the load, and the cost of actions is explicitly taken into account. A systemic risk component is also added, which depends on the aggregated characteristics of the population. This allows penalizing globally dangerous configurations, even with acceptable local behavior. Based on the reward, a global optimality criterion (discounted or averaged over time) is formulated.

The policy of the representative agent is synthesized in a parameterized class independent of the number of nodes. The optimality equation determines the dependence of the optimal actions on the local state and the mean field vector. The policy is found analytically or by reinforcement learning. By working only with local states and a small vector of aggregates, the computational complexity does not depend on the population size.

Next, the self-consistency condition is introduced: the policy must be optimal for exactly the population dynamics that it itself creates when applied on a large scale. This is formulated as a fixed point search problem: the policy optimizes a criterion for given distributions, and these distributions arise precisely from this policy. In practice, iterative evaluation and improvement of the policy with parallel updating of the distributions is used.

Architectural parameters are integrated into the mean field context as a vector containing deception node density, MFA coverage, escalation chain depth, and network topological characteristics. This allows for the formulation of a combined optimization problem in which policy and architectural configuration are simultaneously determined, taking into account resource and SLA constraints. At this stage, quantitative scalability criteria are determined.

The algorithm is implemented as a two-scale procedure: inner loop - for a fixed architecture, an approximate mean-field policy is synthesized and the steady-state dynamics are estimated; outer loop - architectural parameters are adjusted based on a global criterion (by gradient, evolutionary, or stochastic methods).

In the final part, the scalability and quality of approximation are analyzed: it is shown that the found policy is an ε-Nash approximation for a system with N agents, and the error decreases with increasing N. It is justified that the complexity of the method is determined by the dimensionality of the local state and the mean field vector, but does not depend on the number of nodes.

Formalization of a population multi-agent system

The formalization of a population multi-agent system to counter social engineering attacks begins with the construction of a mathematical model of the Distributed computer system as a set of interacting detector agents. At the first stage, a set of nodes is given and its division into types (population classes) corresponding to different roles in the network is performed. For each type, local state and action spaces are defined, which specify possible risk states, interaction channels, and response modes. Such a definition of the structure allows us to move on to the representation of the mean field, where the analysis is carried out not at the level of individual nodes, but in terms of population distributions by states and actions. The Distributed computer system is considered as a finite set of nodes, each of which has its own detector agent. Formally, this set is denoted by $\mathcal{I} = \{1,2, \dots, N\}$ which creates the basis for a further aggregated model and simplifies the study of population dynamics.

In this entry $\mathcal{I}$ denotes the set of indices of all nodes that are part of the infrastructure under consideration. Each number$i$ appears in parentheses as an individual identifier for a specific computer network node.N denotes the total number of nodes, i.e. the population size. It is important that in the context of a scalable architecture$N$ is considered as a parameter that can be large and potentially growing, which creates the problem of exponential growth of the configuration space in classical MAS formulations. In the following, each element $i \in \mathcal{I}$ is matched with a detector agent that performs local processing of incoming events related to social engineering. Such an agent is modeled as an autonomous decision-making entity that observes the local state of its system, assesses risks, and selects actions from a certain allowable set. Since nodes in the Distributed computer system have different nature and functional purpose, it is necessary to introduce classification by types, which is a critical prerequisite for the approach based on the mean field model.

A finite set of types is introduced $\mathcal{K} = \{1,2, \dots, K\}$, where $\mathcal{K}$ denotes the set of classification indices, and K denotes the number of distinct node types. Each number $k \in \mathcal{K}$ corresponds to a certain class of population. For each type k a subset of nodes is introduced $\mathcal{I}_k \subseteq \mathcal{I}$, where $\mathcal{I}_k$ denotes the set of indices of those nodes that belong to the type k. If for some k mark through $N_k = \mid \mathcal{I}_k \mid$ number of nodes of this type, then the relation holds $\sum_{k=1}^{K} N_k = N$, where $\sum_{k=1}^{K}$ means summation over all types, and vertical bars in the expression $\mid \mathcal{I}_k \mid$ denote the cardinality of the set, i.e. the number of its elements. Such a division into population classes not only reflects the realistic heterogeneity of the architecture, but also lays the foundation for the further application of the multipopulation mean-field representation, in which each type of node is considered as a separate but interacting population of statistically similar agents.

After specifying the set of nodes and dividing it into types, it is necessary to formally define the local state and action spaces for agents of each type. For this, the sets are introduced $\mathcal{S}^{(k)}$ and $\mathcal{A}^{(k)}, k \in \mathcal{K}$, where $\mathcal{S}^{(k)}$ denotes the local state space of the agent acting on a node of type k, and $\mathcal{A}^{(k)}$ denotes the local space of available actions for such an agent. The index in parentheses (k) is used to fix membership in a particular population type, emphasizing that a user workstation and, for example, a mail gateway have different state semantics and can implement different sets of actions. Formal description of the local state of an agent of type k at a discrete time instant$t$ is given by a

variable $s_{i,t}^{(k)} \in \mathcal{S}^{(k)}, i \in \mathcal{I}_k$. In this expression $s_{i,t}^{(k)}$ denotes the state vector of the agent that is assigned to the node with index i, belonging to type k, at time t. The subscript i captures a specific node within a population of type k, subscript$t$ denotes a discrete time or decision step, and the superscript in parentheses (k) emphasizes the type of population. $s_{i,t}^{(k)}$ It is advisable to consider the local state as a composition of several components that reflect relevant aspects of the task of countering social engineering.

The typical representation is

$$s_{i,t}^{(k)} = (r_{i,t}^{(k)}, c_{i,t}^{(k)}, f_{i,t}^{(k)}, e_{i,t}^{(k)}), \quad (1)$$

where $r_{i,t}^{(k)}$ indicates the current level of risk or suspicion regarding activity observed on the node; $c_{i,t}^{(k)}$ denotes the context of the interaction channel that encodes; $f_{i,t}^{(k)}$ denotes a vector of local signature or behavioral features extracted from event logs, message content, metadata, and user action history; $e_{i,t}^{(k)}$ denotes the current escalation or containment mode for a given node, reflecting whether it is in normal mode, enhanced inspection mode, partial containment mode, or full isolation. This decomposition is not the only possible one, but it demonstrates that the local state contains both a risk assessment and a description of the environment and control policy.

The action space is described similarly. For an agent of type k, operating on node $i \in \mathcal{I}_k$, at time t the selected action is indicated by a variable $a_{i,t}^{(k)} \in \mathcal{A}^{(k)}$. In this expression $a_{i,t}^{(k)}$ denotes the agent's local decision to respond to a current event or set of events. Space $\mathcal{A}^{(k)}$ contains permissible actions for nodes of type k, which may include ignoring the event, simply logging it, displaying a warning to the user, requiring additional authentication, escalating the incident to a centralized security system, or immediately blocking suspicious activity. The choice of a specific set of permissible actions is determined by both security requirements and user convenience and performance constraints, but regardless of implementation, all possible actions are formalized as elements $\mathcal{A}^{(k)}$.

Formal introduction of sets $\mathcal{S}^{(k)}$and $\mathcal{A}^{(k)}$for all $k \in \mathcal{K}$ creates the basis for constructing the dynamics of the multi-agent system in the subsequent steps of the method. It is on these sets that the probabilistic rules of state transitions, reward functions and decision-making policies will be further defined. The key point is that when switching to the mean-field model, the analysis will not be carried out on the space of all vectors $(s_{i,t}^{(k)})_{i\in\mathcal{I}_k, k\in\mathcal{K}}$, whose dimension grows exponentially with increasing N, and on the space of state distributions within each type, which is much less demanding in terms of scalability. The formalization of the population MAC in terms of the set of nodes $\mathcal{I}$, the set of types $\mathcal{K}$and type-dependent state spaces $\mathcal{S}^{(k)}$ and actions $\mathcal{A}^{(k)}$ is a fundamental step that provides the possibility of further application of the mean field model to avoid exponential growth of the space of actions and states as the number of agents in the system grows.

Representation of the population of MAS agents

The transition to the mean-field representation begins with the abandonment of detailed tracking of the states of individual agents and the transition to an aggregated population description. The model is transferred from the microlevel, where each node is analyzed separately, to the macrolevel, where the distribution of states for each type of nodes becomes the main object. The approach eliminates the exponential growth of the state space and actions with an increase in the number of agents. Initially, for each type of node, denoted by the index k, an empirical distribution of states is introduced, which describes how the states of agents of this type are distributed in the population at a fixed point in time. Such a distribution at a point in time $t$with a finite, but possibly large, number of nodes is

$$\mu_t^{(k),N} = \frac{1}{N_k} \sum_{i\in\mathcal{I}_k} \delta_{s_{i,t}^{(k)}}, \quad (2)$$

$\mu_t^{(k),N}$ – the empirical distribution of the states of agents of the type k at a discrete time point t with a total number of nodes of this type equal to N_k, k fixes the belonging to a specific type of population, N indicates the dependence of the empirical distribution on the population dimension, i.e. on the total number of nodes in the system, t denotes the discrete time instant or the step number of the decision-making process at which the distribution of states is fixed, N_k denotes the number of nodes belonging to type k, i.e. the cardinality of the set $\mathcal{I}_k$. $\mathcal{I}_k$ denotes the subset of node indices from the total set $\mathcal{I}$, which are classified as nodes of type k, the notation $\sum_{i\in\mathcal{I}_k}$ - summing over all indices $i$belonging to this subset.

Thus, the sum in the numerator covers the contribution of each agent of type k to the empirical distribution, $\delta_{s_{i,t}^{(k)}}$ denotes the delta distribution (Dirac distribution) centered at the point $s_{i,t}^{(k)}$, the value $s_{i,t}^{(k)}$ denotes the local state of the agent assigned to the node with index i, which belongs to type k, at time t, k fixes the population type, the subscript i identifies a specific node, and the subscript t specifies the time. The operator $\delta_{s_{i,t}^{(k)}}$ can be interpreted

as a single "impulse" at a point in the state space that corresponds to the actual state of a given agent. When all such "impulses" are averaged with a coefficient $1/N_k$, an empirical distribution is formed that reflects the fractions of agents of type k, which are in different states at time t. Thus, $\mu_t^{(k),N}$ is a random measure on the state space $\mathcal{S}^{(k)}$, which depends both on the implementation of the system dynamics and on the specific value of N_k. To describe the configuration of the entire multipopulation system, the empirical distributions over all types are combined into a single vector object $\mu_t = (\mu_t^{(1)}, \dots, \mu_t^{(K)})$. In this notation μ_t denotes the generalized population state of the entire system at time t. Each component $\mu_t^{(k)}$ of such a vector corresponds to the empirical distribution of states in a population of nodes of type k. The superscript $(1), \dots, (K)$ numbers the population types from the first to K the -th, and the subscript t preserves the time dependence. Thus, μ_t can be considered as an element of the Cartesian product of measure spaces, that is, as a set of distributions $\mathcal{P}(\mathcal{S}^{(1)}) \times \cdots \times \mathcal{P}(\mathcal{S}^{(K)})$, where $\mathcal{P}(\mathcal{S}^{(k)})$ denotes the set of all probability measures in the state space $\mathcal{S}^{(k)}$.

The next conceptual step is to introduce a mean-field boundary variable that provides a deterministic description of the population in the regime of infinite node growth. This provides a transition from a stochastic multi-agent model to a compact population representation. The idea is that when $N \to \infty$, under certain regular conditions, the effect of fluctuations of individual agents averages out, and the empirical distributions converge to deterministic measures that satisfy certain evolutionary equations. This transition is formalized by the notation $\mu_t^{(k),N} \implies \mu_t^{(k)}$ while $N \to \infty$, where $\mu_t^{(k)}$ without index N denotes the boundary, the distribution of agent states of the type k at the moment of time t, and the arrow $\Longrightarrow$ is interpreted as the convergence of the distributions in the corresponding sense. In this notation, the superscript (k) again fixes the population type, and the lower index t means discrete time. Similarly, the limit vector description is formed:

$$\mu_t = (\mu_t^{(1)}, \dots, \mu_t^{(K)}), \tag{3}$$

where each component $\mu_t^{(k)}$ are treated as a deterministic measure that evolves according to a system of equations describing the dynamics of the mean field.

This transition from $\mu_t^{(k),N}$to $\mu_t^{(k)}$ is key from the point of view of scalability. It allows us to consider the behavior of the entire system without explicitly tracking the states of each of N agents, and through the analysis of the evolution of a limited number of distributions $\mu_t^{(k)}$, the number of which is equal to the number of types K and does not depend on N. Thus, the complexity of describing the global state of the system is transferred from the level of individual agents to the level of population characteristics. However, even a description in terms of complete distributions μ_t can be overly complex, since the space of probability measures on the state space remains infinite-dimensional.

Therefore, the next step is to introduce a compact parameterization of the mean field variable, i.e., to go from distributions $\mu_t$to a finite-dimensional feature vector. For this, the mapping is defined:

$$\phi(\mu_t) = z_t \in \mathbb{R}^d. \tag{4}$$

where ϕ denotes a mapping or aggregation operator that maps to each population state μ_t a vector of numerical characteristics in the space $\mathbb{R}^d$, μ_t is the argument of this mapping and is the vector of distributions $(\mu_t^{(1)}, \dots, \mu_t^{(K)})$. The result of applying the operator ϕ is denoted by z_t; the subscript t in and z_t means that this vector changes in time following the evolution of population distributions; the superscript in the notation $\mathbb{R}^d$ specifies the dimension of the space in which the vector lives z_t; d is a fixed natural number and defines the number of aggregated features used to describe the global state of the system.

Structurally display ϕ is given by a system of functionals of the distributions μ_t. Each component of the vector z_t can be interpreted as a certain aggregate characteristic calculated for the corresponding distribution. Formally, for each component j can be written

$$z_t^{(j)} = \psi_j(\mu_t), \tag{5}$$

where $z_t^{(j)}$denotes the j-th component of the vector z_t,ψ_j denotes a functional that reflects the input population state μ_t into a scalar value, and the subscript t preserves time dependence; dimension d is determined by the requirements for the accuracy and informativeness of the aggregated description, but does not depend on the number of nodes N. Even if the number of agents in the system increases by several orders of magnitude, the length of the vector z_t remains constant, and therefore the global state that will be used in the control or reinforcement learning problem does not undergo an exponential increase in dimensionality. It is at this stage that the fundamental elimination of exponential growth occurs: instead of considering the full vector of states of all agents, the dimension of which is proportional to N, the system is represented in terms of the mean field variable μ_t and, even more compactly, through its parameterization z_t, which has a fixed dimension d.

Setting the extended state of the representative agent

The formulation of the representative agent problem in the mean-field paradigm involves a transition from describing the entire set of agents to analyzing a single typical system interacting with an aggregated environment. At this stage, the extended state of the agent is given, its stochastic dynamics is determined taking into account the mean-field variables, and the evolution of the population distribution is formalized through the master equation. This approach reduces the multidimensional interaction of a large number of detectors to a controllable problem for a single agent with a parameterized environment.

The extended state of a representative agent at a discrete time instant is denoted by the vector $x_t = (s_t, z_t)$, where x_t is the complete (or extended) state of one conditional agent at a time instant, denoted by the subscript t, s_t is the local state of the agent representing a node of a certain type in the distributed computer system. Formally, we will assume that $s_t \in \mathcal{S}^{(k)}$, where $\mathcal{S}^{(k)}$ is the state space for nodes of type k. The index k fixes the population class, for example, user workstations, mail gateways or web proxies. We define the second component as z_t. Here, z_t denotes the vector of aggregates of the mean field obtained as a result of applying some aggregation operator to the population distribution – μ_t. It is determined that $z_t \in \mathbb{R}^d$, where $\mathbb{R}^d$ is d-dimensional Euclidean space, and the real number d is a fixed dimension that does not depend on the number of nodes in the system. Each component of the vector z_t can be interpreted as a numerical characteristic of the population: the proportion of nodes in a high-risk state, an estimate of the intensity of attacks on each channel, the average level of escalations, or the value of the tail risk characteristic, which reflects the average level of losses in the riskiest situations. Thus, the vector x_t simultaneously captures the local configuration of a specific system and the aggregated state of the entire distributed computer system, which allows the representative agent to take into account the global context when making local decisions.

The dynamics of a representative agent is formalized as a stochastic process in the space of extended states, in which the transition from one discrete time step to the next is described by a conditional state distribution. The local component of this dynamics is given by the transition probability - $\mathbb{P}(s_{t+1} \mid s_t, a_t, z_t)$.

Defining the reward function of the architecture synthesis problem

The reward function in the problem of synthesizing an architecture resistant to social engineering attacks in the mean-field paradigm formalizes the trade-off between security, operational efficiency, and global risk. At the representative agent level, it is given by a local instantaneous evaluation of the solution and is supplemented by a mean-field component reflecting the systemic risk of the node population. This instantaneous reward is included in the global optimization criterion that determines the goal of the synthesis of policy and architectural parameters. The local reward function of the representative agent is given by the mapping:

$$r(s,a,z) = w_{TP}\,\mathbf{1}\{\text{correct blocking}\} - w_{FN}\,\mathbf{1}\{\text{missed attack}\} - w_{FP}\,\mathbf{1}\{\text{false escalation}\} - c(a), \quad (6)$$

where $r(s,a,z)$ – the instantaneous numerical reward that the agent receives for making a one-time decision. Arguments – local state of the representative node at the time point under consideration; it is assumed that $s \in \mathcal{S}^{(k)}$, where $\mathcal{S}^{(k)}$ is the state space of nodes of type k. The argument a denotes the action chosen by the agent in this state, and $a \in \mathcal{A}^{(k)}$, where $\mathcal{A}^{(k)}$ – the set of permissible actions for a node of a given type. The argument z is a vector of mean field aggregates, i.e. $z \in \mathbb{R}^d$, where each component of this vector encodes a certain aggregated characteristic of the population (the fraction of nodes in a high-risk state, the average intensity of attacks, the level of loading, etc.).

Coefficient w_{TP} in the first term is a positive number that determines the "value" of correctly blocking the attack. Index TP is an abbreviation for the "true positive" and emphasizes that this coefficient scales the contribution of correctly detected and blocked attacks.

Function$\mathbf{1}$ {correct blocking} is an indicator function of the "correct blocking" event. It takes on the value$\mathbf{1}$ in those process implementations where the selected action a in the state s, in the current context z, leads to the correct detection and blocking of a real social engineering attack, and zero in all other cases.

So, the addition $w_{TP}\,\mathbf{1}\{\text{correct blocking}\}$ increases the instant reward by exactly w_{TP} in the case of a successful attack blocking and does not affect it if the blocking did not occur or was incorrect.

The second term contains the coefficientw_{FN} and indicator $\mathbf{1}\{\text{missed attack}\}$. Coefficient w_{FN} is a positive number associated with the "cost" of a missed attack; superscript FN Abbreviates "false negative" and reflects a case where the system did not recognize an existing attack as malicious.

Indicator function $\mathbf{1}\{\text{missed attack}\}$ becomes important $\mathbf{1}$ in implementations where, given a given state$s$ and the middle field z, the selected action$a$ results in the real attack not being blocked, and the consequences of the compromise are still observed. Otherwise, the indicator value is zero.

Addition $w_{FN}\,\mathbf{1}\{\text{missed attack}\}$ means that for each missed attack, the instantaneous reward is reduced by the value w_{FN}. In a security design, this weight is usually chosen as the largest among all, since a missed attack has the most severe consequences for the integrity and confidentiality of information assets.

The third term contains the coefficient w_{FP} and indicator $\mathbf{1}\{\text{false escalation}\}$. Coefficient w_{FP} is also a positive number and reflects the losses associated with false positives of the system. Superscript FP means "false positive". Indicator function $\mathbf{1}\{\text{false escalation}\}$ becomes important $\mathbf{1}$ in situations where the chosen action a actually triggers a blocking or escalation of the incident, but the event that caused this decision is not a real attack. As a result, the system generates unnecessary operational costs: unnecessary overload of security services, inconvenience to users, temporary blocking of legitimate actions. Appendix w_{FP} $\mathbf{1}\{\text{false escalation}\}$ imposes a penalty for such actions, balancing between aggressive blocking and the acceptable level of false alarms.

The last addition is $c(a)$ represents the total operational cost of the action a. Here is $c(a)$a deterministic non-negative function that assigns to each action a compares a numerical estimate of resource costs. The presence of this term in the reward function allows the model to discretely take into account the undesirability of too expensive reactions even when they are formally correct.

Local reward is thus introduced $r(s,a,z)$ describes the trade-off at the level of a single node, but does not guarantee proper accounting for systemic, or population, risk. To model this aspect, an extended reward function with a global risk component is introduced in the mean-field formulation: $r_{MF}(s,a,z) = r(s,a,z) - \lambda\,\mathrm{RiskPop}(z)$,where $r_{MF}(s,a,z)$ – a modified instant reward used in the control task in the mean field paradigm.

The first term on the right-hand side is equal to the previously defined local reward $r(s,a,z)$. The second term $\lambda\,\mathrm{RiskPop}(z)$ represents a penalty proportional to the global population risk. λ is a positive weighting factor that controls the intensity of systemic risk consideration in the overall value function; an increase λ means that the architecture more aggressively minimizes population risk even at the expense of local convenience. Functionality $\mathrm{RiskPop}(z)$ denotes a population-level risk measure that depends on the mean-field aggregates z. The argument z in this functionality, it is a vector of aggregated indicators, in particular, the share of nodes in high-risk states, the share of active attacks on channels, and the load estimates on escalation mechanisms. Functionality $\mathrm{RiskPop}(z)$ can be implemented as a CVaR (Conditional Value–at–Risk) measure or other convex risk measure. In the case of CVaR, the functional approximates the expected losses in the upper tail of the loss distribution, which corresponds to protection against rare but extremely dangerous mass attack scenarios. Convexity $\mathrm{RiskPop}(z)$ means that the functional satisfies the property that mixing two regimes with different risk levels does not lead to a reduction in risk compared to the average value, which corresponds to a conservative approach to assessing systemic vulnerability. Thus, the term $\lambda\,\mathrm{RiskPop}(z)$ forces the representative agent policy to consider that individual decisions in an aggressively attacked environment should be more stringent, even if the local configuration seems relatively secure.

Based on the instant reward of the middle field $r_{MF}(s,a,z)$ a global optimization criterion is formulated, which is used to synthesize policy and architectural parameters. In the classical discounted formulation, the criterion is given by the functional:

$$J_\gamma(\pi,\mu_0) = \mathbb{E}\Big[\ \sum_{t=0}^{\infty}\gamma^t\, r_{MF}(s_t,a_t,z_t)\ \ \Big|\ \ \mu_0\Big]. \tag{7}$$

Synthesis of local representative agent politics

The synthesis of the local policy of the representative agent in the mean-field setting is the central element of the method, since it is at this stage that the abstract description of the dynamics and reward function is transformed into a concrete decision rule, which is subsequently transferred to all nodes of the population.

The policy is built on a fixed-dimensional state space given by the pair (s,z) , and in no way depends on the number of agents N. This ensures that there is no exponential growth in complexity when expanding the distributed computer system.

In the first substep, a parameterized class of policies is introduced. In the formalized model, the mapping is considered: $\pi_\theta(a \mid s,z)$, where $\pi_\theta(a \mid s,z)$ is the conditional probability distribution of choosing an action a provided that the extended state of the representative agent is equal to (s,z). Arguments in this expression is the local state of the node, i.e. an element of the space $\mathcal{S}^{(k)}$ for some fixed type k. The argument z is a vector of mean field aggregates belonging to the space $\mathbb{R}^d$; it is a compact representation of the population state obtained by parameterizing the distribution μ_t. The argument a belongs to the action space $\mathcal{A}^{(k)}$, i.e. the set of all permissible agent reactions: blocking, escalation, logging, additional verification, etc. Subscript θ in and π_θ – a parameter vector that specifies a specific policy within a parameterized class. This vector θ belongs to some parametric space $\Theta \subseteq \mathbb{R}^p$, where p is the dimension of the parameterization. Each fixed value θ uniquely defines the rule by which a representative agent transforms the observed state (s,z) into a distribution over a set of actions.

It is fundamentally important that the parameter vector θ does not depend on the number of nodes in the population N. This means that the complexity of the policy does not change when going from a small to a large infrastructure: the same parameterized form $\pi_\theta(a \mid s,z)$ used for both a thousand nodes and a million. Independence θ from N captures the formal fact that the control problem is posed and solved at the level of a representative agent in a state space of constant dimension, where the role of the "environment" is played by only the aggregated characteristics of the mean field, and not the full vector of states of all agents.

The second substep is to define the Bellman equation for the representative agent, which describes the optimal long-term value of each extended state (s,z). This equation is written as:

$$V(s,z) = \max_{a\in\mathcal{A}}[r_{MF}(s,a,z) + \gamma\,\mathbb{E}[V(s',z') \mid s,a,z]], \tag{8}$$

where $V(s,z)$ – value function (or Bellman function), for the discounted formulation of the control problem; s is the local state of the representative node, and z is a vector of mean field aggregates that describe the population context; the value $V(s,z)$ is interpreted as the maximum possible (under all admissible policies) expected value of the total discounted reward that the agent will receive in the future, starting from the extended state (s,z).

Operator $\max_{a\in\mathcal{A}}$ specifies the maximization over the set of permissible actions $\mathcal{A}$. $\mathcal{A}$ is used here as a generalized notation of the action space for a given node type; each action $a \in \mathcal{A}$ is a candidate for choice in state (s,z). The parentheses after the maximum sign contain the expression of the immediate reward plus the discounted mathematical expectation of the future value. Therefore, $r_{MF}(s,a,z)$ in this expression denotes the instantaneous mean-field reward, previously defined as:

$$r_{MF}(s,a,z) = r(s,a,z) - \lambda\,\mathrm{RiskPop}(z). \tag{9}$$

At the same time $r(s,a,z)$ is the local reward of the representative agent, λ is a positive coefficient regulating the weight of the global risk, and $\mathrm{RiskPop}(z)$ is a convex function of the population risk, for example, the CVaR measure. Thus, $r_{MF}(s,a,z)$ reflects both the benefit of correctly blocking an attack and the penalties for missed attacks, false positives, and population risk.

The coefficient γ of the second term is a discounting parameter that belongs to the interval $(0,1)$. It determines the relative importance of future rewards compared to current ones. Values of γ close to unity mean that the system strongly weighs the long-term consequences of decisions, while smaller valuesγ make the model more "myopic".

Expectation operator $\mathbb{E}[V(s',z') \mid s,a,z]$ denotes the expected value of the value function in the next expanded state(s',z') provided that the current state is (s,z), and the selected action is a. Instance s' is a random variable that reflects the local state of the agent at the next time instant, and the instance z' is a random vector of mean field aggregates. These variables are formed according to the transient model of the system. Formally, the mathematical expectation is expressed as:

$$\mathbb{E}[\,V(s',z') \mid s,a,z\,] = \sum_{s'} \int V(s',z')\,\mathbb{P}(\,\mathrm{d}s',\mathrm{d}z' \mid s,a,z\,), \tag{10}$$

where $\mathbb{P}(\mathrm{d}s',\mathrm{d}z' \mid s,a,z)$ – a conditional transition probability measure defined in the space of local states of the agent and aggregates of the mean field, which specifies the probabilities of stochastic transitions in the extended space. The variabless' runs through all possible values of the local state, and the variable z' runs through all possible values of the mean field characteristic vector consistent with the master equation of population distribution evolution: $\mu_{t+1} = \mathcal{F}(\mu_t,\pi)$, and the parameterization

$$z_{t+1} = \phi(\mu_{t+1}). \tag{11}$$

Thus, the member$\gamma\,\mathbb{E}[V(s',z') \mid s,a,z]$ represents the expected discounted future value if in a state (s,z) choose an action a and then act optimally. Bellman's equation, written as,

$$V(s,z) = \max_{a\in\mathcal{A}}[r_{MF}(s,a,z) + \gamma\,\mathbb{E}[V(s',z') \mid s,a,z]], \tag{12}$$

specifies a stationary optimality condition. It states that the optimal value $V(s,z)$ in any state corresponds to the maximum value of the instantaneous reward of the mean field plus the discounted expected future value. It is important that the argument of the function V there is a couple (s,z) where s varies in space $\mathcal{S}^{(k)}$, and z belongs to $\mathbb{R}^d$ with fixed dimension d. The number of nodes does not appear anywhere in this equation N; it only affects indirectly through the accuracy of the approximation of the mean-field dynamics for a finite population.

The optimal policy within a parameterized class is denoted by π_{θ^*}. θ^*– the optimal parameter vector for which the maximum of the selected global criterion is achieved, for example, the discounted total reward $J_\gamma(\pi,\mu_0)$ or time-averaged reward $\bar{J}(\pi,\mu_0)$. Formally, the optimal parameters can be determined as: $\theta^* = \arg\max_{\theta\in\Theta} J(\pi_\theta,\mu_0)$,where $\arg\max_{\theta\in\Theta}$ – the set of parameter values θ that maximize the functional J, and $J(\pi_\theta,\mu_0)$ – a global criterion of effectiveness is selected when using the policy π_θ and the initial distribution μ_0. For each admissible θ function $\pi_\theta(a \mid s,z)$ specifies a specific policy and the optimization task is to find one θ^* for which the corresponding policy provides the best balance between local detection efficiency, minimizing false positives, and reducing global risk.

Receiving θ^* can be done in different ways. In rare cases, when the dynamics and reward have a special structure (for example, linear-quadratic), it is possible to analytically solve the Bellman equation and obtain a closed-form expression for$V(s,z)$ and the corresponding optimal policy π_{θ^*}. In a more general case, reinforcement learning methods are used, where the vector θ is updated by stochastic gradient or actor-critic procedures, i.e., by methods where the parameterized policy (actor) is updated based on estimates of the value or payoff function, which are calculated by the critic based on samples of trajectories $(s_t,z_t,a_t,r_{MF,t})$. An alternative is numerical methods of

dynamic programming with approximation $V(s,z)$ and π_θ using basis functions or neural network parameterizations.

It is critical that all of these methods operate in an extended state space.(s,z) of constant dimension and do not require operating on the full vector of states of all N agents. The dimensionality of the problem is determined by the dimensionality of the local state $|\ s\ |$ and the dimension of the vector of mean field aggregates that generalize the population state of the system d. Since and $|\ s\ |$, and d are fixed and independent of N, the complexity of learning or computing an approximate policy grows polynomially with respect to the dimension of these spaces, but not exponentially with respect to the number of nodes. It is the fact that the optimal policy $\pi_{\theta^*}(a \mid s,z)$ is defined on a compact space (s,z) and has a parameterization independent of N, formally fixes the absence of exponential growth of the space of states and actions in the solution formulation of the problem of synthesizing a scalable architecture resistant to social engineering attacks.

Formation of the optimal policy of a representative agent

The self-consistent mean-field fixed point is the stage where the population dynamics, mean-field parameterization, reward function, Bellman equation, and the class of representative agent policies are locked into a single system. Here, the requirement of consistency between how the agent's policy changes the population state and how this state determines the optimal policy is formulated. The result is the problem of finding a fixed point for the pair "policy - mean-field dynamics" that corresponds to mean-field equilibrium or self-consistent optimal control.

The self-consistency condition states that an optimal policy should be optimal for the dynamics of the environment that it generates. That is, the policy and the evolution of the aggregated characteristics of the mean field should form a stable fixed point where the agent's actions and the population's response are mutually consistent.

Formally, the optimal policy is denoted as – π^*. In this case –π represents the policy, i.e. the rule for choosing an action depending on the extended state (s,z), and the superscript in the form of an asterisk$*$ means that this policy is optimal in some sense (e.g., maximizing the discounted total reward). Population state – μ_t; subscript t indicates discrete time, and μ this is a vector of distributions for all types of nodes.

The consistency condition means, on the one hand, that the policy π^* is the solution of the Bellman equation for the dynamics corresponding to the evolution of μ_t. On the other hand, the same evolution μ_t is formed if the policy is used in the master equation of population evolution π^*. That is π^* optimal for "its" environment, and "its" environment is the result of the action π^* on a large population. It is advisable to represent this concept in the form of a system of two equations: one determines the optimal policy of the representative agent, and the other forms the dynamics of the mean field that this policy generates in the population, ensuring the condition of self-consistency. Thus, the system can be represented in the form of the following system of equations:

$$\begin{matrix} \pi^* = \ \arg\max_\pi J(\ \pi,\mu) \\ \mu = \mathcal{G}(\pi^*). \end{matrix}, \tag{13}$$

where π^* is the optimal policy that needs to be found, expression$\arg\max_\pi J(\ \pi,\mu)$ – the set (or point) of policy values π for which the functional $J(\pi,\mu)$ reaches its maximum value. π – under the operator$\arg\max$ is an arbitrary valid policy from a selected policy class, such as parameterized policies $\pi_\theta(a \mid s,z)$; functionality $J(\pi,\mu)$ – a global performance criterion, such as expected discounted total reward $J_\gamma(\pi,\mu_0)$ or time-averaged reward $\bar{J}(\pi,\mu_0)$. Argument μ is used as a generalized characteristic of the population dynamics induced by the choice of a certain trajectory of the mean field states. Thus, the first equation formally means that π^* maximizes the chosen criterion J for fixed population dynamics, which is described either by the initial distribution μ_0 , or some steady state μ/

In the second equation $\mu = \mathcal{G}(\pi^*)$ is a self-consistency condition for the population state. In this case –μ it is a description of the population in terms of a mean field, such as a stationary distribution or the entire trajectory $\{\mu_t\}_{t\geq 0}$, depending on the specific formulation of the problem. $\mathcal{G}$ – an operator that each policy π maps the corresponding mean-field state generated by integrating the master equation of population evolution using this policy. The argument $\mathcal{G}$ is a policy π^* that determines how agents in the population choose actions for each extended state (s,z). The value $\mathcal{G}(\pi^*)$ is a population description in terms of the mean field μ , which is determined by the equation $\mu_{t+1} = \mathcal{F}(\mu_t,\pi^*)$, where $\mathcal{F}$ is the evolution operator, and μ_0 – initial distribution. Thus, the second part of the equation fixes that μ is not an arbitrary parameter in the criteria $J(\pi,\mu)$, but rather acts as the population process that arises under the action of the optimal policy π^*.

The solution of this system of equations is a pair (π^*,μ) that simultaneously satisfies the condition of optimality of the policy and the evolution of the mean field; such a pair is interpreted as a mean field equilibrium or as optimal control in a self-consistent model.

For practical implementation of the synthesis method, it is necessary not only to conceptually introduce the fixed point condition, but also to present an algorithm for its approximate finding. At the computational level, this is most often implemented as an iterative procedure in which the policy and the state of the mean field are alternately refined.

To formalize such a procedure, it is advisable to introduce an iteration index, which is denoted by$k$ and runs through the value $0,1,2,\ldots$. At each iteration step, the current approximation to the policy, denoted by $\pi^{(k)}$, and the current approximation to the population state, denoted by, are stored $\mu^{(k)}$.

We believe that in step with the number k is given a certain population state $\mu^{(k)}$, which is interpreted as an approximate estimate of the true mean-field dynamics obtained as a result of previous policy updates or system evolution. For a fixed $\mu^{(k)}$ an optimal control problem is formulated for a representative agent, i.e. an abstract "typical" node, whose behavior models the local dynamics of any population element – in the state space (s,z), where the part z, corresponding to the mean field, is considered as a deterministic or time-parameterized process, built on the basis of $\mu^{(k)}$. At this stage, the Bellman equation for the corresponding dynamics is solved and the approximation to the optimal policy is calculated, which is denoted $\pi^{(k+1)}$. This step is interpreted as policy improvement, i.e. improving the policy in a fixed environment. Computationally, this can be implemented through dynamic programming, gradient methods of policy with a value estimator, or other reinforcement learning algorithms that operate in a state space of constant dimension (s,z).

Next, with a fixed updated policy $\pi^{(k+1)}$, the mean field description is updated $\mu^{(k)}$. To do this, the evolution of the population distribution is calculated by solving the master equation $\mu_{t+1} = \mathcal{F}(\mu_t, \pi^{(k+1)})$, starting from some initial distribution μ_0, and the trajectory is determined $\{\mu_t^{(k+1)}\}_{t\geq 0}$ or the corresponding stationary regime. The result of this step is written as $\mu^{(k+1)} = \mathcal{G}(\pi^{(k+1)})$, where the operator $\mathcal{G}$ has the same meaning as in the fixed-point system: for a given policy, it returns the mean-field process generated by it. This step is interpreted as a "policy evaluation" at the population level: an assessment is made of what global risk profile the new policy leads to, provided that it is widely adopted by all agents.

The complete iterative scheme has the form of a sequence

$$\mu^{(k)} \;\longrightarrow\; \pi^{(k+1)} \;\longrightarrow\; \mu^{(k+1)}\,, \tag{14}$$

where the first correspondence reflects the synthesis of the representative agent's policy for a fixed mean-field environment, and the second correspondence reflects the update of the mean-field environment for a fixed policy. Index k increases and the process continues to increase sequentially, and iterations continue until the pair $(\pi^{(k)}, \mu^{(k)})$ approaches a fixed point in the given metric. Formally, this means that in the limiting case at $k \to \infty$ convergence is realized $\pi^{(k)} \to \pi^*$and $\mu^{(k)} \to \mu$, where(π^*, μ) satisfy the system with $\pi^* = \arg\max_{\pi} J(\pi, \mu), \mu = \mathcal{G}(\pi^*)$. In this iterative design, it is critical that all calculations in the policy synthesis stage occur in the state space (s,z) constant dimension; transition to the space of distributions μ_t only needed to update the midfield environment via the appropriate operators $\mathcal{F}$ and $\mathcal{G}$. Number of agents N enters the scheme only indirectly – through the accuracy of the approximation of the mean-field dynamics by a real multi-particle system – but does not affect the dimensionality of the optimal control problem solved by the representative agent.

This is what turns the synthesis of a scalable architecture resistant to social engineering attacks into a procedure for finding a self-consistent mean-field fixed point, performed without exponential growth of the state and action space.

Integrating architectural parameters into the mean field description

The integration of architectural parameters into the description of the mean field is the stage at which the multicomponent stochastic model is transformed into a full-fledged IT architecture design tool. At this level, not only the representative agent policy, but also the architecture configuration itself – in particular, the density of deception nodes, the proportion of nodes with multifactor authentication, the depth of escalation chains and the topology of network segmentation – are directly included in the mean field variables, affecting the population risk and attack dynamics. The starting point is a description of the population in terms of the mean field through the distribution of states μ_t and its compact parameterization

$$z_t = \phi(\mu_t), \tag{15}$$

where μ_t – vector of population distributions at a discrete time instant t, i.e. $\mu_t = (\mu_t^{(1)}, \ldots, \mu_t^{(K)})$,$\mu_t^{(k)}$ is a probability measure on the space of local states of nodes of type k; z_t– a vector of aggregated features of the population state, which belongs to the Euclidean space $\mathbb{R}^d$ constant dimension; ϕ – denotes a functional mapping from the distribution space μ_t in the finite-dimensional space of characteristics z_t; – each component z_t is a functional of μ_t, (for example, the fraction of high-risk nodes) or the average intensity of attacks.

At the architecture integration stage, this parameterization is supplemented: the mean field aggregates include not only current population characteristics, but also static architectural parameters that determine the conditions for the spread of attacks and the effectiveness of protective mechanisms.

To formally include architectural parameters, we introduce a vector κ, which denotes the set of all IT architecture configuration parameters that are considered as design variables. Vector κ belongs to some parametric

space $\mathcal{K} \subseteq \mathbb{R}^q$, where the number q is the dimension of the architecture parameterization. Each component κ_j in vectorκ has a specific meaning. One group of components are indicators related to deception nodes; for example, the density of decoys in different network segments or the proportion of hosts that can behave as a honeypot when detecting anomalous activity.

Another group is the level of multifactor authentication. This can be the proportion of nodes where MFA is activated, or the level of MFA mandatory for critical operations. The architecture is further defined by the depth of the escalation chains, which can be formalized as the maximum or average length of the incident route from the local agent to the central SOC, and the degree of branching of these routes.

As a result, the network topology is characterized by a system of topological indicators, in particular the average node degree, the number of network segments, the average length of the shortest paths between critical nodes, the clustering coefficient, and other structural indicators.

All these quantities are represented as components of a vector κ, where each coordinate encodes one structural aspect of the architecture.

After introducing the vector κ, the aggregates of the mean field are determined not only by the distribution μ_t, but also by the architectural configuration. This is written as the relation: $z_t = \phi(\mu_t, \kappa)$. Where z_t is the vector of aggregated features at time t, μ_t – population distribution of states, reflecting the current "dynamic" configuration of the system, κ is a static vector of architectural parameters that is specified at the design level. Function $\phi(\mu_t, \kappa)$ is now constructed in such a way that part of the coordinates z_t reflected the dynamic state of the population (e.g., the proportion of nodes in certain risk classes), while the other part of the coordinates was directly equal to the componentsκ or their functionalities. To clearly fix the structure of the aggregates, we will use the following notation: $z_t = (z_t^{\text{dyn}}, z^{\text{arch}})$, where the first block$z_t^{\text{dyn}}$ depends on μ_t, and the second blockz^{arch} is equal to some function $h(\kappa)$. In this casez_t^{dyn} – dynamic aggregates that change over time, and z^{arch}– a set of effective architectural characteristics that remain constant for a given configuration. h can be either a trivial mapping that simply copies the componentsκ into the appropriate components z , or a more complex mapping that combines structural parameters into integrated risk or reliability indicators. It is important that in this definition $\phi(\mu_t, \kappa)$ vector dimension z_t remains constant: although it usually has to be increased from d to d', the value d' is still chosen to be fixed and independent of the population size N. Thus, the expansion of the mean-field aggregates due to architectural parameters does not violate the central property of scalability: the optimization of the representative agent's policy continues to be performed in a state space of fixed dimension, but this space now explicitly encodes the structural characteristics of the architecture.

The next step is to formulate a combined optimization problem in which the optimal control policy and the optimal architectural parameters are simultaneously determined. The global criterion is denoted by $J(\pi, \kappa)$.J – quality functional, which can be, for example, the expected discounted total reward or the time-averaged reward in the stationary mode; π – policies applied by representative agents; formally π is given as a mapping from the state space (s, z) into a distribution over a set of actions; κ – is used to denote an architectural configuration that determines both the dynamics of the mean field (in particular, by changing the network topology, the density of deception nodes, and the proportion of MFA nodes), and the resource intensity of its implementation in the IT architecture.

The joint optimization problem of policy and architectural parameters, subject to mean field dynamics and resource constraints, is formulated as: $\max_{\pi,\kappa} J(\pi, \kappa)$,where $\max_{\pi,\kappa}$– means that the pair (π^*, κ^*) that maximizes the functional is sought. J by all permissible policies π and permissible architectural parameters κ. Permissibility κ is set by a set of constraints, which include restrictions on the cost of hardware infrastructure, on available human resources, on the maximum allowable user delay, as well as the requirements of service level agreements (SLAs).

Such constraints can be formalized through auxiliary functions. For example, the function $C_{\text{res}}(\kappa)$ can be interpreted as the total resource costs associated with the architectural configuration κ, and the constant $C_{\max}$ is the maximum allowable budget. Then the constraint is written as $C_{\text{res}}(\kappa) \leq C_{\max}$. Similarly, the function $SLA(\pi, \kappa)$ can mean the worst or average response time for a legitimate user, and a constant $SLA_{\min}$ is the minimum acceptable level of service; then the condition is imposed $SLA(\pi, \kappa) \geq SLA_{\min}$. All these constraints together define the permissible region in the space (π, κ) within which the maximization is performed $J(\pi, \kappa)$.

In such a formulation, the dynamics of the mean field is taken into account through the dependence $J(\pi, \kappa)$ from the distribution μ_t, which evolves according to the equation: $\mu_{t+1} = \mathcal{F}(\mu_t, \pi, \kappa)$, and through the parameterization $z_t = \phi(\mu_t, \kappa)$. $\mathcal{F}$ acts as an evolutionary operator that reflects the current population state μ_t, policy π and architectural parameters κ into the next population state μ_{t+1}. Dependence $\mathcal{F}$ from κ means that the architecture affects the probabilities of transitions between states, for example by changing the probability of successful attack propagation between nodes or by changing the distribution of delays and intensities of escalation flows.

Within this single optimization problem, it is advisable to define scalability criteria that allow assessing the stability of the architecture. κ^* and politics π^* to increase the population size N. One such criterion is the behavior of the population risk functional RiskPop as N. We introduce the following notation $R(N; \pi, \kappa)$ for the expected

value RiskPop(z_t) in the steady state or in a characteristic time horizon for a population of size N, when all agents apply the policy π in the architecture given by κ. Thus, R is the risk function, N – the number of nodes, and π, κ specify the configuration of the control and architecture. A condition of scalability can be considered such a property when there is a limit $\limsup_{N\to\infty} R(N; \pi^*, \kappa^*)$ that is finite and does not exceed the acceptable risk threshold. This means that when the number of nodes increases, the population risk does not grow uncontrollably, but remains limited due to the chosen architecture and policy.

The second critically important aspect of scalability is the behavior of the computational and communication load on the agent and central modules as they grow N.

For this purpose, functions are introduced $L^{\text{loc}}_{\text{comp}}(\pi, \kappa)$ and $L^{\text{loc}}_{\text{comm}}(\pi, \kappa)$, which respectively denote the computational and communication load on one representative agent for given policies and architecture. In this case π in these functions means that the complexity of local decision-making and signal processing algorithms depends on the form of the policy (for example, on the size of the neural network that approximates π), and κ means that the architecture may require more or less messaging with central components.

Functions are introduced similarly $L^{\text{cent}}_{\text{comp}}(\pi, \kappa, N)$ and $L^{\text{cent}}_{\text{comm}}(\pi, \kappa, N)$, which describe the load on the central modules, with explicit consideration of N. The scalability requirement is such a dependence, under which $L^{\text{loc}}_{\text{comp}}(\pi^*, \kappa^*)$ and $L^{\text{loc}}_{\text{comm}}(\pi^*, \kappa^*)$ remain constant or grow slowly with increasing N, and the growth $L^{\text{cent}}_{\text{comp}}(\pi^*, \kappa^*, N)$ and $L^{\text{cent}}_{\text{comm}}(\pi^*, \kappa^*, N)$ has a non-exponential nature, such as polynomial or sublinear. This formally ensures that even with an increase in the number of nodes, the control policy and architectural mechanisms remain feasible in practice.

Thus, the integration of architectural parameters into the description of the mean field through the vector κ, the extended parameterization $z_t = \phi(\mu_t, \kappa)$ and joint optimization formulation $\max_{\pi,\kappa} J(\pi, \kappa)$ with resource and SLA constraints translates the mean-field model to the level of a full-fledged architecture synthesis tool.

In this formulation, the scalability of the architecture is quantified through the behavior of population risk. RiskPop and the load on the system components as the number of agents increases, and it is the mean field apparatus that allows us to formally prove that the optimal pair (π^*, κ^*) ensures the absence of exponential growth neither in the space of states and actions, nor in the resource requirements of the system.

Synthesis of a scalable architecture

The process of synthesizing a scalable architecture formalizes previous theoretical constructs as an iterative procedure that simultaneously approximates the optimal policy of the representative agent and the optimal configuration of architectural parameters. At this stage, two time scales are combined: the inner scale corresponds to the rapid adaptation of the policy under a fixed architecture, while the outer scale describes the slow evolution of the architecture itself based on aggregated performance and scalability indicators.

All this takes place in a mean-field formulation, where the global state of the system is given by distributions μ_t and their compact parameterization z_t, and increasing the number of agents N affects only the accuracy of such an approximation, but does not change the dimensionality of the control problem.

The initial stage consists in specifying the starting approximations for the architectural configuration and population state. The architecture is given by the vector $\kappa^{(0)}$. In this notation κ describes a vector of architectural parameters belonging to the space $\mathcal{K} \subseteq \mathbb{R}^q$, where q is the fixed dimension of the architecture parameterization, and the superscript (0) denotes the initial iteration of the outer loop. Each coordinate $\kappa_j^{(0)}$ This vector reflects a specific structural parameter: the density of deception nodes, the proportion of nodes with multifactor authentication, the depth of escalation chains, the topological characteristics of network segmentation. At the same time, an initial description of the population is given in terms of the mean field, which can be given either as a vector of distributions $\mu^{(0)}$, or through its compact parameterization $z^{(0)}$. In the first case $\mu^{(0)}$ this is the initial population state, i.e. a system of measures $\mu_0^{(k)}$ on the local state spaces for each type of node k. In the second case $z^{(0)}$ is a vector of aggregated indicators, which is obtained by mapping ϕ, i.e. $z^{(0)} = \phi(\mu^{(0)}, \kappa^{(0)})$, where ϕ is a functional mapping that maps the population distribution and architectural configuration to the mean field feature vector in the corresponding space $\mathbb{R}^{d'}$ constant dimension. Choosing a specific initialization method (through $\mu^{(0)}$ or through $z^{(0)}$) depends on whether the model is built from an abstract aggregated description or from real data of event logs and topology, but in both cases the initial state does not contain a dimensionality dependence on N.

The inner loop operates with a fixed vector of architectural parameters κ, which currently corresponds to some approximation $\kappa^{(\ell)}$, where ℓ is the iteration index of the outer loop. For a fixed $\kappa^{(\ell)}$ The problem is reduced to the synthesis of the policy of a representative agent in the state space (s, z). Formally, this policy is given by a parameterized family $\pi_\theta(a \mid s, z)$, where π_θ is a reflection of the policy parameterized by the parameter vector θ,a is a reflection of action, s – local state of the node, and z is the vector of aggregated characteristics of the mean field. For a fixed architecture $\kappa^{(\ell)}$ The Bellman equation is solved for the value function $V^{(\ell)}(s, z)$, which has the following form:

$$V^{(\ell)}(s, z) = \max_{a\in\mathcal{A}} \Big[r_{MF}\big(s, a, z; \kappa^{(\ell)}\big) + \gamma\, \mathbb{E}\big[V^{(\ell)}(s', z') \mid s, a, z, \kappa^{(\ell)} \big] \Big], \tag{16}$$

where $V^{(\ell)}(s,z)$ is the value function for the iteration of the architecture with index ℓ; arguments$s$ and z correspond to typical components: the local state of the agent and the vector of mean field aggregates, index (ℓ) marks the dependence on the current architectural configuration; $\mathcal{A}$ – the set of the space of permissible actions; $r_{MF}(s,a,z;\kappa^{(\ell)})$– the instantaneous reward function in the mean field model, which can explicitly depend on the architectural parameters through a change in the risk weights or through a modification of the structure of the population risk functional. Parameter γ– discount factor in the interval $(0,1)$ (the discount factor determines the relative weight of future rewards compared to current ones in optimal control and reinforcement learning problems); the expectation operator $\mathbb{E}[V^{(\ell)}(s',z') \mid s,a,z,\kappa^{(\ell)}]$ is calculated by the transition probability, which takes into account both the local dynamics of the agent and the evolution of the mean field variables under a fixed architecture $\kappa^{(\ell)}$.

The solution of this equation allows us to construct an approximately optimal policy $\pi^{(\ell)}$, which in practice is implemented through parameter settings. $\theta^{(\ell)}$ vector θ so that politics $\pi_{\theta^{(\ell)}}$ approximated the optimal policy for this architecture.

After receiving the policy $\pi^{(\ell)}$ The inner loop continues by calculating the updated mean field dynamics. The dynamic population state is described by a trajectory $\{\mu_t^{(\ell)}\}_{t\geq 0}$ that satisfies the master equation

$$\mu_{t+1}^{(\ell)} = \mathcal{F}\big(\mu_t^{(\ell)}, \pi^{(\ell)}, \kappa^{(\ell)}\big), \tag{17}$$

where $\mu_{t+1}^{(\ell)}$ – population distribution at a point in time$t+1$ in the configuration corresponding to the iteration ℓ, $\mathcal{F}$– population evolution, which is built on the basis of local agent dynamics and architecture topology, and the arguments $\pi^{(\ell)}$ and $\kappa^{(\ell)}$ set policy and architectural parameters accordingly. Trajectory $\mu_t^{(\ell)}$ are integrated either until a stationary distribution is reached $\mu_\infty^{(\ell)}$or until a quasi-stationary regime is reached over a long but finite horizon. The corresponding mean–field parameterization is calculated as $z_t^{(\ell)} = \phi(\mu_t^{(\ell)}, \kappa^{(\ell)})$, where $z_t^{(\ell)}$is the vector of mean–field aggregates at time t when configuring the architecture $\kappa^{(\ell)}$. In the stationary case, the characteristic is used $z_\infty^{(\ell)} = \phi(\mu_\infty^{(\ell)}, \kappa^{(\ell)})$:

Thus, the inner loop, starting from $\kappa^{(\ell)}$, consistently forms an approximately optimal policy $\pi^{(\ell)}$ in the space of constant dimensions (s,z) and the corresponding stationary configuration of the mean field $(\mu^{(\ell)}, z^{(\ell)})$, where neither the dimensionality of the state space nor the dimensionality of the parameters depend on the number of agents N.

The outer loop operates on the architectural configuration, using the results of the inner loop to evaluate the quality of the current pair $(\pi^{(\ell)}, \kappa^{(\ell)})$. The global quality criterion is denoted by the functional $J(\pi,\kappa)$. Where J – a scalar function that summarizes the behavior of the system over a long time horizon: it can be the expected discounted total reward, the asymptotic average reward, the negative population risk, or a combined indicator that incorporates both safety and resource intensity. π in functionality J corresponds to a policy that describes the behavior of a representative agent and, in the mean-field approximation, generalizes to the entire population.κ interpreted as an architectural configuration that affects both the dynamics of attacks and defenses, as well as resource consumption. At the point $(\pi^{(\ell)}, \kappa^{(\ell)})$ value is calculated $J(\pi^{(\ell)}, \kappa^{(\ell)})$ or, when using stochastic methods, its estimation based on a sample.

The update of architectural parameters can be specified by different types of iterations. In the gradient formulation it is written as $\kappa^{(\ell+1)} = \kappa^{(\ell)} + \eta_\ell \nabla_\kappa J\big(\pi^{(\ell)}, \kappa^{(\ell)}\big)$. Where $\kappa^{(\ell+1)}$ – a new approximation of the architectural configuration; η_ℓ is the gradient update step at iteration ℓ, and $\nabla_\kappa J(\pi^{(\ell)}, \kappa^{(\ell)})$ is the gradient of the quality criterion along the vector of architectural parameters at the point $(\pi^{(\ell)}, \kappa^{(\ell)})$. This gradient can be obtained analytically, if the model structure allows differentiation via master equations, or approximately, using methods such as REINFORCE for structural parameters, or numerically, via finite differences.

Alternatively, evolutionary or stochastic optimization methods can be used, in which the new approximation $\kappa^{(\ell+1)}$ is formed as a random mutation or combination of previous configurations with the selection of the best ones in terms of value J.

The process is repeated until convergence, which means that the sequence $\{\kappa^{(\ell)}\}_{\ell\geq 0}$ stabilizes in the vicinity of some vector κ^*, and the sequence of policies $\{\pi^{(\ell)}\}_{\ell\geq 0}$ obtained by the inner loop for each one $\kappa^{(\ell)}$leads to a certain policy π^*. Formally, this is expressed by the conditions $\lim\limits_{\ell\to\infty} \kappa^{(\ell)} = \kappa^*$, $\lim\limits_{\ell\to\infty} \pi^{(\ell)} = \pi^*$, where $\lim_{\ell\to\infty}$is the limit of the sequence as the index increases ℓ, and the vectors κ^* and π^* are respectively the limiting architectural configuration and policy. At this point the method fixes the pair (κ^*, π^*) as a result of the synthesis of a scalable architecture. Vector κ^* is interpreted as an optimized architecture configuration that specifies the density of deception nodes, MFA coverage, the depth of escalation chains, and topological segmentation, and the policy π^* is the optimal or approximately optimal response of the detectors to the local states and the mean field that arise in this architecture.

A key consequence of this construction is that both the inner and outer loops operate in spaces of constant dimensions. The inner loop solves the optimal control problem in the state space (s, z), where the dimension of the local state is$s$ and the vector of mean field characteristics$z$ does not depend on the number of agents N. The outer loop optimizes the vectorκ in space $\mathbb{R}^q$, where q is also fixed. All dependencies on N are included only indirectly - through the operator form $\mathcal{F}$ and the accuracy of the mean field approximation. This formally establishes the absence of exponential growth of the state and action space in the synthesis method and confirms the scalability of the constructed architecture, resistant to social engineering attacks.

Scalability analysis of architecture and mean field policies

The analysis of scalability and approximation properties completes the construction of the method, since it is at this stage that it is formally justified that the mean-field description is not just a convenient heuristic, but really approximates the behavior of a real N-agent system with a controlled error, and without an exponential increase in computational complexity with an increase in the number of agents. Within the framework of the problem of countering social engineering attacks, such justification means that the synthesized policy and architecture remain relevant for large corporate infrastructures, and the quality of protection does not degrade in an unpredictable way with an increase in the number of nodes.

To formalize the approximation properties, a finite N-agent system is considered, in which each node of the network is represented by a detector agent that makes decisions according to the policy obtained in the mean-field model. The states of the entire system are denoted by the vector:

$$s_t^N = (s_{1,t}, \dots, s_{N,t}), \tag{18}$$

where s_t^N – global state at the discrete time instant t, superscript N fixes the dependence on the number of agents, and each component $s_{i,t}$ – local state of the agent with index i at time t. Each agent applies a policy that is a copy of the mean–field policy π^{MF}constructed for the representative agent in the state space (s, z). Formally, this means that at any time for each agent$i$ action $a_{i,t}$ is chosen according to the distribution $a_{i,t} \sim \pi^{MF}(\cdot \mid s_{i,t}, z_t^N)$, where $a_{i,t}$ acts as an agent's action i at the moment of time t,~ means that this action is a random variable that is chosen according to a probability distribution, and the expression $\pi^{MF}(\cdot \mid s_{i,t}, z_t^N)$ is a representation of the conditional distribution on the set of actions for a fixed local state$s_{i,t}$ and the aggregated mean field z_t^N. Vector z_t^N – is a finite-dimensional approximation of the mean field variable, constructed on the basis of the empirical distribution of states in the N-agent system. Formally, the empirical distribution is denoted by:

$$\mu_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{s_{i,t}}, \tag{19}$$

where μ_t^N – a random measure on the space of local states, $\delta_{s_{i,t}}$ – point measure centered at the point $s_{i,t}$. The mean field aggregates are given by the mapping: $z_t^N = \phi(\mu_t^N, \kappa)$, where ϕ is the same parameterization operator used in the mean field model, andκ is a vector of architectural parameters. Thus, the policy π^{MF}defined in the mean field boundary is transferred to the N-agent system by replacing the true state of the mean fieldz_t on its empirical approximation z_t^N.

To assess the quality of such an approximation, the concept of the ε–Nash property is introduced. Let for i – the expected discounted quality functionality of the agent when the entire system operates according to the policy π^{MF} is indicated by:

$$J_i^N(\pi^{MF}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{MF}(s_{i,t}, a_{i,t}, z_t^N)], \tag{20}$$

where $J_i^N(\pi^{MF})$ – mathematical expectation of the total discounted reward of the mean field for an agent with index i in the N-agent system; operator $\mathbb{E}[\cdot]$ is interpreted as the mathematical expectation over all random trajectories of the system; the summation is carried out over discrete time t from zero to infinity; parameterγ belongs to the interval $(0,1)$ and is the discount rate; the function $r_{MF}(s_{i,t}, a_{i,t}, z_t^N)$ serves to define the agent's immediate reward i at a point in time t, which depends on its local state $s_{i,t}$, local action $a_{i,t}$ and the empirical mean field in z_t^N. Now let the agent i unilaterally deviates from politics π^{MF} and uses another valid policy $\tilde{\pi}_i$, while all other agents continue to apply policy π^{MF}. The corresponding quality functionality for agent i is determined by: $J_i^N(\tilde{\pi}_i, \pi_{-i}^{MF})$, where $\tilde{\pi}_i$ – a policy used only by the agent i, and π_{-i}^{MF} – the vector of policies of all agents, except i, which act according to policy π^{MF}. Policy π^{MF} is called an ε(N) – Nash approximation if for all agents$i$ and all permissible one-sided deviations $\tilde{\pi}_i$ the inequality holds $J_i^N(\pi^{MF}) \geq J_i^N(\tilde{\pi}_i, \pi_{-i}^{MF}) - \varepsilon(N)$.

In this inequality $\varepsilon(N)$ – is a non-negative function of the number of agents N, which characterizes the maximum gain of an agent from a unilateral deviation from the mean-field policy; if $\varepsilon(N)$ tends to zero at $N \to \infty$, then the mean-field policy asymptotically becomes a true Nash equilibrium. Typical estimates in mean-field theory give a dependence of the form: $\varepsilon(N) \leq \frac{C}{\sqrt{N}}$.

Experiments

Fixed local state space$\mathcal{S}$ and actions $\mathcal{A}$. Middle ground politic sπ^{MF} is trained once for a representative agent; this time does not depend on N. Further, for each N is being modeled $T = 1000$ steps of an N-agent system where all agents use the same π^{MF}.

Event flow and policy parameters: probability of attack: $p_{\text{attack}} = 0.05$, probability of correctly blocking an attack: $p_{\text{detect}} = 0.9$, probability of incorrectly blocking a legitimate event: $p_{\text{fp}} = 0.02$

For everyone N calculated: mean field policy training time (same for all N); total simulation/planning time for N-agent system; time per step for one agent; quality indicators: TPR, FPR, FN – rate, FP – rate, simple risk index risk = FN_rate + 0.1 · FP_rate.

Table 1

Results for different N (all times in seconds, agent time in microseconds)

N	training time via mean_field	evaluation total time	evaluation time per agent step (μs)	TPR	FPR	FN_rate	FP_rate	risk metric
5	0.00039	0.02426	0.31	0.9066	0.0158	0.0934	0.0158	0.0950
20	0.00039	0.02355	0.43	0.8969	0.0208	0.1031	0.0208	0.1052
50	0.00039	0.02812	0.56	0.8966	0.0200	0.1034	0.0200	0.1054
100	0.00039	0.02484	0.25	0.8927	0.0220	0.1073	0.0220	0.1095
200	0.00039	0.07616	0.38	0.9005	0.0200	0.0995	0.0200	0.1015
500	0.00039	0.03614	0.07	0.8978	0.0202	0.1022	0.0202	0.1042
1000	0.00039	0.04441	0.04	0.8985	0.0202	0.1015	0.0202	0.1035
5000	0.00039	0.12243	0.02	0.8996	0.0200	0.1005	0.0200	0.1025
10000	0.00039	0.18338	0.02	0.8996	0.0201	0.1004	0.0201	0.1024
100000	0.00039	1.71601	0.02	0.9001	0.0200	0.0999	0.0200	0.1019

Analysis of the experimental results showed that the learning time of the mean field policy is the value 0.00039c, and it is the same for all N, since learning occurs once in space (s, z) of constant dimension and does not depend on the number of agents.

Total planning time eval_time_total increases with N approximately linearly, but the time per step for one agent quickly stabilizes at the microsecond level $\approx 0.02, 0.06$demonstrating the absence of exponential growth in computational cost.

The quality of protection TPR fluctuates around the theoretical 0.9, FPR – around the theoretical 0.02; risk_metric stabilizes in the interval $\approx 0.10 – 0.11$. With the growth of$N$ fluctuations decrease, which corresponds to the ε–Nash approximation with an error of order $O(1/\sqrt{N})$.

Conclusion

The article proposes a method for synthesizing a scalable architecture of a distributed computer system that is resistant to social engineering attacks. The approach is based on population multi-agent modeling, which allows describing the behavior of a large number of nodes through aggregated population characteristics without increasing the dimensionality of the problem as the infrastructure grows.

The developed model combines local node states with a global risk background, which allows for the coordination of individual agent decisions with the needs of the entire system. The state structure, decision-making rules, interaction between agents, and the mechanism for achieving a self-consistent system state in which agent policies and population dynamics do not contradict each other are determined.

The architectural parameters of the system segmentation, access levels, multifactor authentication coverage, deception node density are integrated into a population model, which allows optimizing both agent behavior and the structure of the Distributed computer system simultaneously.

Experimental results demonstrate that the training and policy execution time are independent of the infrastructure scale, and the attack detection quality remains stable even with a large number of nodes. This confirms the scalability of the model and its practical applicability to real-world enterprise environments.

As a result, the proposed approach provides: synthesis of an architecture resistant to social engineering attacks; coordinated operation of a large number of detectors in different interaction channels; combination of local solutions with global risk assessment; possibility of optimizing both agent behavior and infrastructure structure. The proposed method forms the basis for creating scalable Distributed computer systems with formally justified properties of resistance to complex social engineering attacks.

Author Contributions

For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used “Conceptualization, O.B.; methodology, O.B.; software, O.B.; validation, O.B.; formal analysis, O.B.; investigation, O.B.; resources, O.A.; data curation, O.B.; writing - original draft preparation, O.A.; writing - review and editing, O.A.; visualization, O.B. and O.A.; supervision, O.B.; project administration, O.B.; funding acquisition, O.B. All authors have read and agreed to the published version of the manuscript.

Declaration on the use of generative artificial intelligence tools"

In preparing this work, the authors used Grammarly for: grammar and spelling checks. After using this service, the authors reviewed and edited the content and take full responsibility for the content of this publication.

References

1. Fakhouri H. N., Makhadmeh S. N., Alhadidi B., Hamad F., Omar K., Halalsheh N. AI-driven solutions for social engineering attacks: detection, prevention, and response. Preprint. 2024.
2. Ogunlade O. Human vulnerabilities in cybersecurity: analyzing social engineering attacks and AI-driven machine learning countermeasures. European Journal of Technology. 2025. Vol. 9(1), pp.186–197.
3. AI-driven social engineering attack detection in email communications. International Journal of Artificial Intelligence and Deep Learning Research and Development. 2025. Vol. 6(1), pp.15–23.
4. Bokhonko O., Lysenko S., Gaj P. Development of the social engineering attack models. AdvAIT 2024 – Advanced Applied Information Technologies, CEUR-WS. 2024. Vol. 3899, pp.288–303.
5. Haq Q. E. U., Faheem M. H., Ahmad I. Detecting phishing URLs based on a deep learning approach to prevent cyber-attacks. Applied Sciences. 2024. Vol. 14(22), 10086.
6. Innab N., Osman A. A. F., Mohammed Ataelfadiel M. A., Abu-Zanona M., Elzaghmouri B. M., Zawaideh F. H., Alawneh M. F. Phishing attacks detection using ensemble machine learning algorithms. Computers, Materials & Continua. 2024. Vol. 80(1), pp.1325–1345.
7. Senouci O., et al. Enhancing phishing detection in cloud environments using deep learning. Journal of Telecommunications and Information Technology. 2025.
8. Kim J., Gu S., Kim Y., Lee S., Kang C. A multimodal voice phishing detection system integrating text and audio analysis. Applied Sciences. 2025. Vol. 15(20), 11170.
9. Vulfin A., et al. A multimodal phishing website detection system using multimodal analysis and explainable AI. Preprints. 2025.
10. Ghadami R., et al. Explainable hybrid deep learning–optimization framework for phishing detection. Engineering Applications of Artificial Intelligence. 2025.
11. Ravula V., et al. Enhancing phishing detection with dynamic optimization of deep learning models. PeerJ Computer Science. 2025.
12. Tian Y., et al. PRIME: a phishing detection framework with quantitative risk modeling. IEEE Transactions on Knowledge and Data Engineering. 2025.
13. Finistrella S., Mariani S., Zambonelli F. Multi-agent reinforcement learning for cybersecurity: methods and challenges. WOA 2024 – From Objects to Agents, CEUR-WS. 2024. Vol. 3735, pp.103–118.
14. Singh A. V., Rathbun E., Graham E., Oakley L., Boboila S., Oprea A., Chin P. Hierarchical multi-agent reinforcement learning for cyber network defense. AAMAS 2025 (extended abstract). Full version: arXiv:2410.17351. 2024.
15. Yaroviy O. Intelligent agents in multi-agent information security systems. Scientific and Methodological Issues of Computer Sciences (SMiCS). 2024/2025.
16. Vidyasri P., Suresh S. FDN-SA: fuzzy deep neural stacked autoencoder-based phishing attack detection in social engineering. Computers & Security. 2025. Vol. 148, 104188.
17. Mahmud M., Alshamrani A., Rahman M., et al. Hybrid deep learning approaches to smishing attack detection. Systems. 2024. Vol. 12(11), 490.
18. Meda S., Srinivas V. S., Rao K. C. B., et al. Dual-phase deep learning framework for advanced phishing detection using OptSHQCNN. PeerJ Computer Science. 2025. Vol. 11, e3014.
19. Hosseinzadeh M., Ali U., Ali S., et al. Improving phishing email detection performance through deep learning with adaptive optimization. Scientific Reports. 2025.
20. Barik K., Misra S., Mohan R. Web-based phishing URL detection model using deep learning optimization techniques. International Journal of Data Science and Analytics. 2025.
21. Thiruvoth P., Laya D., Ahmad I., et al. Phishing detection system for enhanced cybersecurity. Proceedings of International Conference on Information Systems Security. 2025.
22. Muñoz J. B., Islam R. Deep learning approaches for multi-class classification of phishing text messages. Preprint. 2025.
23. Baccour E., Erbad A., Mohamed A., Hamdi M., Guizani M. Multi-agent reinforcement learning for privacy-aware distributed CNN in heterogeneous IoT surveillance systems. Journal of Network and Computer Applications. 2024. Vol. 230, 103933.
24. Wang L., Wei Z., Guo W. Multi-agent deep reinforcement learning-based key generation for graph layer security. ACM Transactions on Privacy and Security. 2025. Vol. 28(2), Article 18.
25. Fernández Carrasco J. A., Amonarriz Pagola I., Orduna Urrutia R., Román R. CYBERSHIELD: a competitive simulation environment for training AI in cybersecurity. IOTSMS 2024 – International Conference on Internet of Things: Systems, Management and Security. 2024. pp.11–18.

Oleksandr Bokhonko **Олександр Бохонько**	асистент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет https://orcid.org/0000-0002-7228-9195	Assistant of the Department of Computer Engineering and Information Systems, Khmelnytskyi National University
Olha Atamaniuk **Ольга Атаманюк**	асистент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет https://orcid.org/0000-0001-9802-864X	Assistant of the Department of Computer Engineering and Information Systems, Khmelnytskyi National University