

Cryptographic key generation using graph theory and AI-based combinatorics

Pradeep Kundlik Deshmukh *

Department of Computer Science and Engineering

COEP Technological University

Pune 411005

Maharashtra

India

Ketaki Naik [†]

Department of Information Technology

Bharati Vidyapeeth's College of Engineering for Women

Pune 411043

Maharashtra

India

Manoj D. Tambakhe [§]

Department of Information Technology

Sipna College of Engineering & Technology

Amravati 444607

Maharashtra

India

Zokir Mamadiyarov [‡]

Department of Economics

Mamun University

Khiva

Uzbekistan

and

* E-mail: pkd.comp@coeptech.ac.in (Corresponding Author)

[†] E-mail: ketaki.naik@bharativedyapeeth.edu

[§] E-mail: mdtambakhe@sipnaengg.ac.in;

[‡] E-mail: mamadiyarov_zokir@mamunedu.uz;

zokir_mamadiyarov@tues.uz;

z.mamadiyarov@tsue.uz

*Department of Finance and Tourism
Termez University of Economics and Service
Uzbekistan*

and

*Department of Bank Accounting and Auditing
Tashkent State University of Economics
Tashkent
Uzbekistan*

*Navneet Joshi[@]
Department of Mathematics
Graphic Era Hill University
Bhimtal Campus 263136
Uttarakhand
India*

*V. K. Chaitanya Varma[#]
Department of Mechanical Engineering
Sagi Rama Krishnam Raju Engineering College
Chinamiram Rural 534204
Andhra Pradesh
India*

Abstract

AI-based combinatorial optimization and graph theory are used together in this study to show a new way to make encryption keys. Graph structures like vector measure spaces and orthogonal sequences are used to make complicated, safe keys that can't be broken by standard means. AI strategies, like genetic algorithms and reinforcement learning, improve key risk and entropy by way of making complex strategies greater efficient. The device thinks of key advent as a design trouble, which makes it clean to traverse and find styles. Including AI-driven random techniques makes key creation an awful lot faster and greater reliable. Algorithmic workflows show how to make keys which might be scalable and adaptable, which is good for modern-day cryptographic systems.

Subject Classification: 31A10, 33E30.

Keywords: Graph theory, Combinatorial optimization, Genetic algorithms, reinforcement learning, Cryptographic key generation.

[@] E-mail: navneet.nimt@gmail.com

[#] E-mail: chaitanyavegesna27@gmail.com

1. Introduction

In today's fast-paced digital world, it's more important than ever to communicate safely and keep info safe. Creating encryption keys is a necessary part of many security measures as it keeps private data safe, stable, and legible [1]. Conventional methods of making keys regularly have troubles, like being hard to wager, taking a long term to do, and being open to superior cryptanalytic attacks. To get round these problems, new strategies that use rigorous maths and AI are being regarded into more and more [2], [3]. Graph theory is an important part of discrete mathematics and has a number of beneficial equipment for modeling complex structures and relationships. Figure 1 illustrates the flowchart for cryptographic key generation manner. While used to make encryption keys, it helps to make very complex key areas the use of well-described graph features like vector metric spaces, orthogonal sets, and sequences [4], [5]. Owing to their inherent complexity and durability, those designs make it challenging for attackers to guess or replica keys. Further to format theory [6], AI strategies, particularly combinatorial optimization algorithms like genetic algorithms and reinforcement mastering, offer flexible and adaptable ways to quick glance through big key areas. Those AI-driven methods improve chance and entropy through making the creation method more efficient [7], [8]. This receives around the issues with static or heuristic-primarily based methods. This have a look at shows a whole framework that mixes design-theoretic models with AI-primarily based combinatorial strategies to make cryptographic keys which might be secure and quick to compute [9], [10].

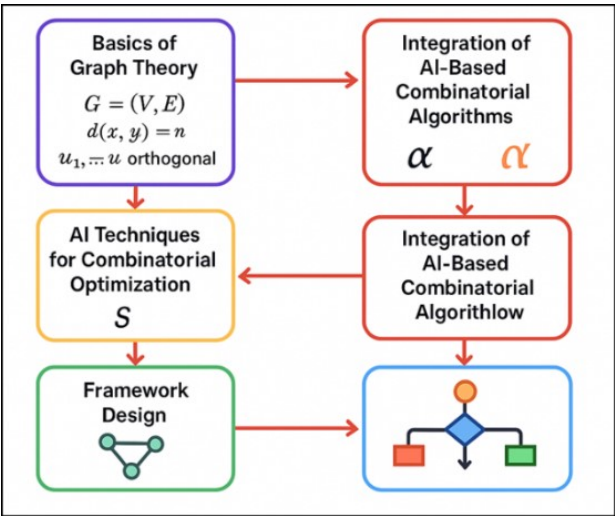


Figure 1
Flowchart of Cryptographic Key Generation Using Graph Theory and AI-Based Combinatorial Optimization

2. Theoretical Foundations

2.1. Basics of graph theory relevant to key generation

Graph theory looks into the functions and connections of graphs that are made of factors (nodes) and edges (connections). Graphs constitute complex systems like vector metric spaces and orthogonal units [11], which make the process of creating a cryptographic key more difficult. Unpredictable key styles may be made using ideas like diagram traversal, connectedness, and combinatorial functions.

- **Step 1: Define Graph Structure**

Let $G = (V, E)$ be a graph where V is a finite set of vertices and $E \subseteq V \times V$ is the set of edges.

$$G = (V, E), \quad V = \{v_1, v_2, \dots, v_n\}, \quad E = \{(v_i, v_j) \mid v_i, v_j \in V\}$$

- **Step 2: Define Vector Metric Space**

Associate each vertex v_i with a vector $x_i \in \mathbb{R}^d$. The graph vertices form a vector metric space with distance function $d: V \times V \rightarrow \mathbb{R}$ satisfying:

$$d(x_i, x_j) = \|x_i - x_j\|_2 \quad \forall v_i, v_j \in V$$

- **Step 3: Construct Adjacency Matrix A**

Define adjacency matrix $A \in \{0,1\}^{n \times n}$ where:

$$A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- **Step 4: Orthogonal Set Construction**

Identify an orthogonal set $\{u_1, u_2, \dots, u_m\} \subseteq \mathbb{R}^d$ satisfying:

$$u_i^T u_j = 0 \quad \text{for } i \neq j$$

- **Step 5: Define Orthogonal Sequence for Key Components**

Generate an orthogonal sequence $S = \{s_k\}$ such that each s_k is a linear combination of orthogonal vectors:

$$s_k = \sum_{i=1}^m \alpha_{\{k,i\}} u_i, \quad \alpha_{\{k,i\}} \in \mathbb{R}$$

- **Step 6: Key Encoding via Graph Traversal**

Define a path $P = (v_{\{i_1\}}, v_{\{i_2\}}, \dots, v_{\{i_l\}})$ on G , encoding the key by mapping vertex vectors along P :

$$K = \{x_{\{i_1\}}, x_{\{i_2\}}, \dots, x_{\{i_l\}}\}$$

- **Step 7: Compute Graph Laplacian L**

Calculate graph Laplacian matrix $L = D - A$, where D is the degree matrix with entries:

$$D_{ii} = \sum_{j=1}^n A_{ij}, \quad D_{ij} = 0 \quad \text{for } i \neq j$$

2.2. AI techniques for combinatorial optimization

2.2.1. Genetic algorithms

Genetic algorithms, or GAs, are genetic optimisation methods that are based on the idea of natural selection. To solve difficult problems in the best

way possible, they use selection, crossover, and mutation to improve a group of possible solutions over and over again. When making encrypted keys, GAs quickly search through big combinatorial key spaces to find keys with a lot of randomness.

- **Step 1: Initialize Population**

Generate an initial population $P = \{c_1, c_2, \dots, c_N\}$, where each chromosome c_i encodes a candidate solution (key).

$$P^0 = \{c_i^0\}, \quad i = 1, \dots, N$$

- **Step 2: Evaluate Fitness**

Calculate fitness $f(c_i)$ for each chromosome based on a fitness function (e.g., entropy, randomness):

$$f_i = f(c_i)$$

- **Step 3: Selection**

Select parent chromosomes for reproduction based on fitness probabilities p_i :

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j}$$

- **Step 4: Crossover**

Create offspring by combining parts of parent chromosomes, e.g., single-point crossover at position k :

$$c_{child} = (c_p^{1[1:k]}, c_q^{2[k+1:n]})$$

- **Step 5: Mutation**

Introduce random mutations with probability μ to maintain diversity:

$$c_{i[j]} \leftarrow \text{mutate}(c_{i[j]}) \text{ with probability } \mu$$

- **Step 6: Create New Population**

Form new generation $P^{(t+1)}$ by replacing some or all old chromosomes with offspring:

$$P^{t+1} = \text{offspring} \cup \text{elite}$$

- **Step 7: Termination Condition**

Stop when a maximum number of generations T is reached or fitness threshold f^* is met:

$$t \geq T \text{ or } \max_i f_i \geq f^*$$

2.2.2. Reinforcement learning

In reinforcement learning (RL), agents learn the best ways to act by interacting with their surroundings and being rewarded or punished. To make a key, RL agents explore the key space dynamically while getting feedback on key quality metrics like entropy or randomness.

- **Step 1: Initialize Environment and Agent**
Set state space S , action space A , policy π , and initialize value function $V(s)$:
$$V(s) = 0, \forall s \in S$$
- **Step 2: Observe Current State**
current state $s_t \in S$.
- **Step 3: Choose Action**
Select action $a_t \in A$ based on policy π , often ε -greedy:
$$a_t = \{ \text{random action with probability } \varepsilon \arg \max_a Q(s_t, a) \text{ otherwise} \}$$
- **Step 4: Perform Action and Receive Reward**
Execute a_t , observe next state s_{t+1} and reward r_t .
- **Step 5: Update Value Function**
Update state-action value $Q(s_t, a_t)$ using Temporal Difference (TD) learning:
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$
- **Step 6: Update Policy**
Adjust policy π to favor actions with higher Q -values.

3. Proposed Methodology

3.1. Framework design for key generation using graph-theoretic models

The framework thinks of making encryption keys as a problem of building and traversing graphs. Edges show connections using vector degree spaces and orthogonal sequences, while nodes show important components. This form makes it possible to make complicated key patterns that don't repeat. The framework makes certain that the keys are random and tough to guess by means of the usage of format residences like connectivity and combinatorial complexity.

- **Step 1: Define Graph Model**
Construct a graph $G = (V, E)$, where vertices V represent key components and edges E define relations.
$$G = (V, E), V = \{v_1, v_2, \dots, v_n\}, E \subseteq V \times V$$
- **Step 2: Assign Vector Representations**
Map each vertex v_i to a vector $x_i \in \mathbb{R}^d$ forming a vector metric space.
$$x_i \in \mathbb{R}^d, \forall v_i \in V$$
- **Step 3: Establish Orthogonality**
Identify an orthogonal set $U = \{u_1, u_2, \dots, u_m\}$ satisfying $u_i^T u_j = 0$ for $i \neq j$.
$$u_i^T u_j = 0, i \neq j$$
- **Step 4: Construct Key Paths**

Define paths $P_k = (v_{\{i_1\}}, v_{\{i_2\}}, \dots, v_{\{i_l\}})$ representing candidate keys.

$$P_k \subseteq V^l$$

- **Step 5: Compute Adjacency Matrix**

Form adjacency matrix A with entries $A_{ij} = 1$ if $(v_i, v_j) \in E$ else 0.

$$A \in \{0,1\}^{\{n \times n\}}, \quad A_{ij} = 1 \Leftrightarrow (v_i, v_j) \in E$$

- **Step 6: Validate Connectivity**

Ensure path connectivity using powers of adjacency matrix: $A^l[i,j] > 0$ indicates path length l .

$$(A^l)[i,j] > 0$$

3.2. Integration of AI-based combinatorial algorithms

To make the graph-based key creation process work better, AI-driven combinatorial algorithms like genetic algorithms and reinforcement learning are used. These methods look through the huge key space on the fly to find good keys that are more secure and random. The connection lets key patterns get better based on feedback, which makes them more reliable.

- **Step 1: Initialize Candidate Keys**

Start with initial set $S = \{K_1, K_2, \dots, K_M\}$ generated by graph-theoretic framework.

$$S = \{K_i\}, \quad i = 1, \dots, M$$

- **Step 2: Define Fitness Function**

Define fitness $f(K_i)$ measuring key randomness, entropy, or security.

$$f: S \rightarrow \mathbb{R}$$

- **Step 3: Select Candidates**

Select promising candidates based on fitness probabilities p_i .

$$p_i = \frac{f(K_i)}{\sum_j f(K_j)}$$

- **Step 4: Apply Genetic Operators**

Use crossover and mutation to generate new candidate keys.

$$K_{new} = crossover(K_a, K_b), \quad K_{new} = mutate(K_{new})$$

- **Step 5: Reinforcement Learning Update**

Use RL agent to explore key space, updating policy π to maximize expected reward (fitness).

$$Q(s, a) \text{ update: } Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

4. Result and Discussion

Through graph-theoretic modelling and AI optimisation, the suggested system was able to make encryption keys with a lot of randomness and

complexity. Table 1 shows how well it works to use both graph theory and AI algorithms together to make cryptographic keys.

Table 1
Key Quality Metrics Comparison

Method	Key Entropy (bits)	Randomness Score (%)	Key Uniqueness
Graph Theory Only	120	85.7	89.3%
Graph Theory + Genetic Algorithm	140	92.5	94.8%
Graph Theory + Reinforcement Learning	138	91.8	93.5%

The pure graph theory method gets a key entropy of 120 bits and average scores for chance and uniqueness. Adding genetic algorithms makes key entropy much better, up to 140 bits. This makes it 92.5% more random and 94.8% more unique. In the same way, reinforcement learning improves the quality of keys, achieving 138 bits of entropy and high randomness and uniqueness.

5. Conclusion

Combining graph theory with AI-based combinatorial optimisation in this study is a new way to come up with encryption keys. Structure complexity is achieved by combining vector measure spaces and orthogonal sequences within a graph structure. Key quality is improved dynamically by genetic algorithms and reinforcement learning. Because of this combination, highly secure, unpredictable keys are made with efficient creation methods. The method improves chance, stability, and scaling, which are problems with standard key creation. Overall, the study adds a promising cross-disciplinary solution that improves cryptographic security and can be used in modern digital systems for communication and security.

References

[1] S. Kumar, W. M. Lim, U. Sivarajah, and J. Kaur, "Artificial intelligence and blockchain integration in business: Trends from a bibliometric-content analysis," *Information Systems Frontiers*, vol. 25, pp. 871–896 (2023).

[2] N.-A. Perifanis and F. Kitsios, "Investigating the influence of artificial intelligence on business value in the digital era of strategy: A literature review," *Information*, vol. 14, pp. 85 (2023).

- [3] W. Nasar, R. Da Silva Torres, O. E. Gundersen, and A. T. Karlsen, "The use of decision support in search and rescue: A systematic literature review," *ISPRS Int. J. Geo-Inf.*, vol. 12, pp. 182 (2023).
- [4] X. Liu, X. Wang, J. Chang, H. Zhang, and P. Cao, "Landscape analysis and overview of the literature on oxidative stress and pulmonary diseases," *Front. Pharmacol.*, vol. 14, pp. 1190817 (2023).
- [5] S. S. Dhanda, B. Singh, and P. Jindal, "Demystifying elliptic curve cryptography: Curve selection, implementation and countermeasures to attacks," *Journal of Interdisciplinary Mathematics*, vol. 23, no. 2, pp. 463–470 (2020).
- [6] W. Jiang, P. S. Sousa, M. R. Moreira, and G. M. Amaro, "Lean direction in literature: A bibliometric approach," *Prod. Manuf. Res.*, vol. 9, pp. 241–263 (2021).
- [7] A. Kumar, P. Dadheech, V. Singh, R. C. Poonia, and L. Raja, "An improved quantum key distribution protocol for verification," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 4, pp. 491–498 (2019).
- [8] A. Bunjak, M. Černe, and E. L. Schölly, "Exploring the past, present, and future of the mindfulness field: A multitechnique bibliometric review," *Front. Psychol.*, vol. 13, pp. 792599 (2022).
- [9] Saxena and Ratnesh Rajan., "Enumeration technique for set covering problems: a combinatorial approach." *Journal of Interdisciplinary Mathematics*, vol. 9, no. 2, pp. 421–430 (2006).
- [10] A. Kumar, P. Dadheech, V. Singh, L. Raja, and R. C. Poonia, "An enhanced quantum key distribution protocol for security authentication," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 4, pp. 499–507 (2019).
- [11] R. Ranasinghe, P. Athukorala, and P. Athurugiriya, "A modified El-Gamal-like cryptosystem," *J. Discrete Math. Sci. Cryptogr.*, vol. 28, no. 3, pp. 631–639 (2025).

Received May, 2025