EXPLORING NOVEL CRYPTOGRAPHIC METHODS FOR SECURE KEY GENERATION

Integrating Emoji Semantics into Secure Key Generation

Bilge Demir

Department of Computer Engineering
Istanbul Kultur University
Istanbul, Turkey
bilgedemir1560@gmail.com

Hayal İldeniz İnanç
Department of Computer Engineering
Istanbul Kultur University
Istanbul, Turkey
ildenizinanc@gmail.com

Özge Küçükbayram
Department of Computer Engineering
Istanbul Kultur University
Istanbul, Turkey
ozgekucukbayram@hotmail.com

Abstract*— This project introduces a semantically emoji-assisted password generation system based on natural language processing (NLP) to facilitate the creation of secure and memorable passwords. The system performs grammatical analysis on user input to extract syntactic elements such as subject, verb, and object, and generates passwords by mapping these components to semantically relevant emojis. It also supports conventional password creation using standard character sets including lowercase letters, uppercase letters, digits, and symbols, with user-configurable password length and character types. If the chosen character categories do not fulfill the desired password length, the system automatically completes the password with either emojis or standard characters based on user preference, providing alternative suggestions for both emoji-enhanced and non-emoji passwords. The security of generated passwords is evaluated via statistical randomness tests—Monobit, Runs, Cumulative Sums, and Serial Tests—and includes an estimated brute-force attack cracking time to provide feedback. Accessible through a web-based interface, the system offers customizable password generation to meet diverse security requirements.

***Keywords*— password generation, natural language processing, emoji-based passwords, randomness testing, brute-force resilience**

I. Introduction

The exponential growth of online services has elevated digital identity protection into a critical concern. While users increasingly perform banking, education, healthcare, and communication tasks online, many continue to rely on weak password practices. Common behaviors such as reusing passwords, incorporating personal information, and choosing predictable patterns expose users to threats like credential stuffing, brute-force attacks, and widespread data breaches.

Creating passwords that are both secure and easy to remember remains a significant challenge. Although complex passwords with a mix of uppercase, lowercase, numbers, and symbols enhance security, their memorability is low. Consequently, users often resort to simplifying passwords, storing them insecurely, or neglecting regular updates. Overcoming this trade-off between usability and security necessitates innovative, user-focused password generation techniques.

This study introduces a password generation system utilizing Natural Language Processing (NLP) combined with semantic emoji matching to produce personalized, secure, and visually memorable passwords. The system analyzes user input to extract key grammatical elements—subject, verb, and object—and associates these with relevant emojis from a structured dataset. A weighted selection algorithm prioritizes emoji assignment in the order of object, verb, then subject to ensure semantic coherence and uniqueness.

Beyond emoji integration, the system includes traditional password components to meet complexity requirements. Generated passwords undergo rigorous security assessments, including statistical randomness tests (Monobit, Runs, Cumulative Sums, Serial tests) and brute-force resistance simulations. Delivered through a web-based interface, the system provides real-time password generation accessible to users with varying technical expertise.

The remainder of this paper is structured as follows: Section II reviews related literature; Section III details the methodology and system design; Section IV explains the implementation; Section V discusses evaluation results; and Section VI concludes with findings and future work suggestions.

II. Related Work

A. Password Generation Techniques and User Behavior

The study titled *Evaluating Passwords: User Behavior and the Psychology Behind Password Choice* highlights that users frequently favor memorability over entropy, often reusing weak and predictable passwords composed of personal or common terms. This behavioral pattern has informed the design of our system, which aims to preserve user familiarity while increasing security by transforming user input into structurally varied and semantically rich passwords. Specifically, the system introduces randomized layers such as character substitutions, casing variations, and emoji encoding to balance cognitive usability with security [1].

In comparison, the method presented in *Strong Password Generation Based on User Inputs* similarly takes user preferences as input but applies character-level transformations such as Leetspeak and random insertions. Our system expands upon this by incorporating weighted grammatical roles (subject, verb, object) and matching them with contextually appropriate emojis, thus extending password complexity through semantic personalization. In alignment with the findings in *Passwords and User Behaviors*, which emphasizes the importance of linguistic and cognitive alignment for user acceptance, our system enhances password strength without sacrificing user intuitiveness [2][3].

B. NLP and Emoji Matching

Although natural language processing (NLP) has been widely applied in sentiment analysis and digital communication, its application in secure password generation is relatively unexplored. Our project introduces an NLP-based system that analyzes input sentences to identify grammatical roles—subject, verb, and object—and assigns semantically appropriate emojis based on these roles to increase entropy and memorability.

In contrast to EmojiAuth, where users manually select emoji passwords from a fixed grid, leading to personal biases and predictable choices , our system uses automatic, role-based emoji matching to reduce repetition and enhance unpredictability. This automated method avoids the non-uniform emoji distributions reported in prior studies [4].

Furthermore, the SSRN study describes emojis as “an evolution of older visual language systems” and “non-verbal surrogates” for meaning, supporting our view that emojis should be treated as semantic units. Unlike previous approaches that treat emojis as arbitrary icons or rely on user selection, our method applies emoji logic to password generation in a structured, semantically coherent, and security-focused way [5].

C. Randomness Testing and Cryptographic Quality

This study adopts the NIST SP 800-22 Rev1a framework, as introduced by Rukhin et al. (2010), to evaluate the statistical randomness of generated passwords. Four core tests—Monobit, Runs, Additive Sums, and Series—are reimplemented in Python and applied to SHA-256 hashes of password outputs, including those with emojis. Hashing ensures encoding consistency, particularly for multibyte Unicode characters, and supports entropy-focused evaluation [6].

To further reduce pattern bias, post-generation operations follow entropy-preserving algorithms. Inspired by Bishoi and Maharana (2017), we adopt the Fisher-Yates shuffle for randomizing all password components—letters, symbols, and emojis. This aligns with the approach used by Kirana et al. (2021) in educational systems. Together, these methods enhance output unpredictability and validate statistical integrity, even for semantically generated structures [7], [8].

D. Brute Force Attacks and Resistance Estimation

Vaithyasubramanian et al. proposed a Markov chain-based password generation model that improves brute-force resistance by using probabilistic transitions based on linguistic frequency patterns. This method reduces predictability while preserving usability, demonstrating that transition probabilities significantly lower brute-force success rates compared to random character generation [9].

Our system follows a comparable principle by assigning emojis to subject, verb, and object roles through semantic parsing. Although not probabilistic, this role-based structure introduces controlled variability, minimizing fixed patterns and enhancing resistance to brute-force attacks.

Additionally, the EmojiStory system by Kjellevand and Rauhut explores entropy-based modeling of emoji sequence spaces and calculates the combinatorial size of possible passwords. Similarly, our system estimates brute-force resistance by accounting for Unicode-aware character diversity and simulating attack scenarios where emojis are unknown to the attacker. This approach highlights how attacker assumptions critically impact resistance calculations. By incorporating SHA-256 hashing and dynamic simulation, our system offers a more comprehensive and realistic security evaluation [10].

E. UI Design and Application Integration in Emoji-Based Security Systems

The SEntiMoji study by Chen et al. emphasizes backend modularity between NLP components and emoji representations but lacks a graphical user interface, limiting its applicability for end users. In contrast, our system features an

interactive web-based UI that not only displays emoji-enhanced passwords but also provides dynamic feedback, input-based emoji previews, and real-time password strength metrics. These interactive elements enhance usability and transparency in ways not addressed by SEntiMoji [11].

Similarly, the BITREST system developed at Strathmore University focuses on GUI-based password storage using emojis as selectable input elements. While both systems integrate emojis into user workflows, BITREST relies on manual selection from a static emoji pool. Our system, by contrast, automates emoji assignment through NLP-based semantic analysis. Moreover, we incorporate real-time entropy evaluation and randomness testing, offering a more security-aware and personalized experience. Despite methodological differences, both approaches underscore the potential of emoji integration to improve memorability and user engagement [12].

III. METHODOLOGY

A.Requirement Analysis and Use Case Modeling

The system requirements were defined through functional analysis of user interactions. The password generator is designed to allow users to input a word or sentence, specify the desired password length, and choose among character types including lowercase letters, uppercase letters, digits, symbols, and emojis. The system processes this input to generate a password that meets the criteria and, optionally, semantically maps words to relevant emojis. Users can regenerate passwords, view the output, and copy it to the clipboard.

To enhance security, the system includes a Recommended Password feature that strengthens passwords when users select limited or unbalanced character types, especially emojis. This feature increases the number of emojis and balances all selected character types to create a stronger, less predictable password. It maintains semantic relevance while filling any remaining spaces with random characters, then shuffles the entire password to ensure both security and usability.

To specify system behavior, detailed use case scenarios were created. These include operations such as text input, password generation, brute-force simulation, and randomness testing. When the emoji option is selected, the system displays a text field for input and uses fuzzy logic to match semantically relevant emojis—even for misspelled words. If no character types are selected, the system prevents password generation and prompts the user to correct the input. The use case diagram also includes administrator actions such as setting brute-force limits and adjusting test parameters for enhanced control.

B. Design

The proposed system was designed using a modular, web-based architecture consisting of a Python-Flask backend, a JavaScript/HTML/CSS frontend, and a structured JSON-based emoji dataset. These components interact through RESTful API calls to handle tasks such as natural language processing (NLP), semantic emoji matching, password generation, statistical testing, and brute-force simulation.

System functionalities were mapped through a use case diagram that includes two primary actors: the User and the Admin. Users can enter a base sentence, set a password length, choose character types (letters, digits, emojis, etc.), and generate a password. The system evaluates the generated password using Monobit, Serial, Cumulative Sums, and Runs Tests. If the password fails these tests, the system suggests a stronger version automatically. Users can also simulate brute-force attacks and copy generated passwords to the clipboard. Admins are responsible for managing system configurations, including test thresholds and emoji dataset updates.

.

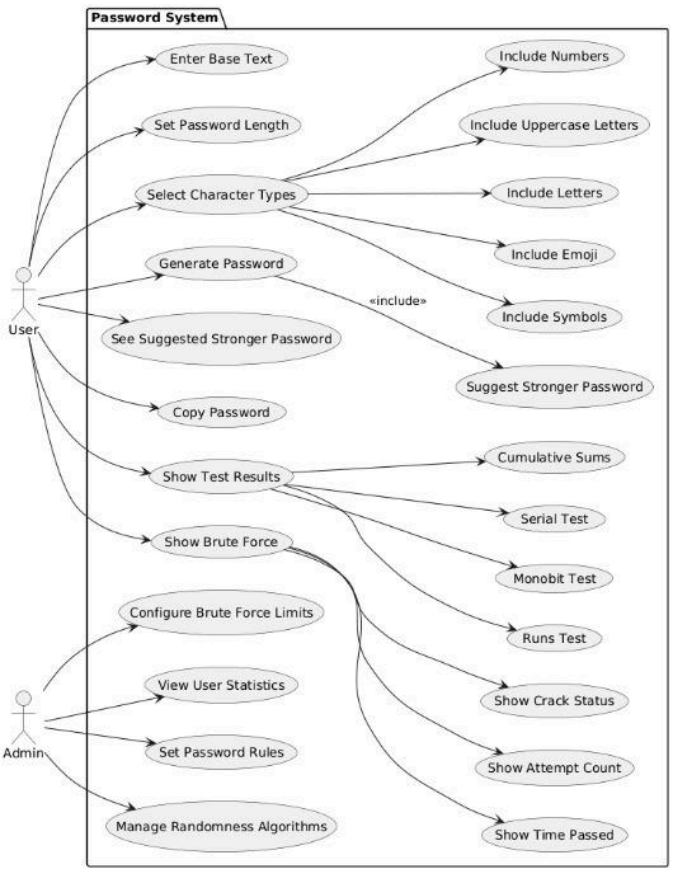

Fig.3.1: Use Case diagram illustrates these functionalities.

Although not visually included, the internal logic of the system also follows clear dynamic and structural models. The activity flow handles emoji selection logic, conditional UI rendering, and test initiation based on user preferences. The backend is organized with Flask controllers, semantic parsing modules (spaCy and PorterStemmer), a JSON emoji loader with fuzzy matching (RapidFuzz), and cryptographic hashing using SHA-256. Passwords are constructed dynamically based on user constraints, and if incomplete, are auto-filled with contextually relevant emojis or fallback characters.

The frontend complements this architecture with real-time input validation, password preview, randomness test visualization, and result comparison for standard versus emoji-enriched outputs. The password generation is also driven by entropy optimization, semantic richness, and user experience. This design ensures a robust balance between usability and cryptographic security while enabling meaningful personalization.

C. Implementation

The system was implemented as a modular web-based application composed of a Flask-powered backend, a JavaScript/HTML/CSS frontend, and a structured emoji dataset in JSON format. The backend is responsible for linguistic analysis, password generation, emoji matching, hashing, and security testing. Upon receiving text input from the user, the backend utilizes the spaCy NLP library to parse the sentence and extract grammatical components such as subject, verb, and object. Lemmatization is applied to standardize word forms, and the identified tokens are then semantically matched with emojis from the dataset. To improve matching accuracy, the system incorporates PorterStemmer for stemming and RapidFuzz for fuzzy similarity scoring. This ensures that even misspelled or inflected words can be reliably associated with semantically appropriate emojis. Emojis are selected according to a weighted strategy prioritizing objects first, followed by verbs, and then subjects. If the number of selected characters (e.g., emojis, symbols, digits) is less than the desired password length, the remaining portion is automatically filled with contextually relevant emojis or classical characters. This dynamic logic ensures each password is semantically expressive, personalized, and meets the required length.

To further enhance password strength, the system includes a Recommended Password feature that activates when users choose limited or unbalanced character sets, especially with emojis. This feature generates a stronger password by increasing the number of emojis beyond the user's initial selection, while maintaining semantic relevance. It also balances the inclusion of all selected character types and fills any remaining characters with random choices from the combined set. The final password is shuffled to maximize unpredictability, providing a secure yet meaningful password recommendation.

Once the password is generated, it is hashed using the SHA-256 algorithm to convert it into a uniform 256-bit binary string. This hashed representation is then evaluated through four NIST-inspired statistical randomness tests: Monobit, Runs, Cumulative Sums, and Serial. These tests assess entropy and distribution balance in the binary output to verify cryptographic strength. Additionally, a brute-force simulation module implemented in Flask dynamically constructs a character pool based on the structure of the password and performs randomized guessing for up to 30 seconds. If the password is cracked, the system provides the number of attempts and elapsed time; if not, failure is reported.

On the frontend, interactive components guide the user through text input, character selection, and password generation. Features such as real-time length counters, error warnings, dynamic field visibility, and password preview enhance usability. When the selected character types are insufficient to meet the desired length, the system displays both an emoji-supported and a non-emoji alternative password for comparison. The frontend also presents visual feedback for randomness tests and brute-force resistance, making the password creation process transparent and informative for users.

D. Testing

The testing phase of this project involved multiple layers to ensure both the functional integrity and the security robustness of the system. Initially, manual test cases were applied to verify the frontend interactions and backend logic under standard and edge-case scenarios. These included verifying correct responses when no input was given, when character type selections exceeded the defined password length, or when actions were triggered without a valid password. The system was observed to consistently block invalid operations and provide appropriate feedback—for example, issuing warnings when selections were misaligned or preventing the use of functions like "Copy" and "Brute Force" unless a valid password was present.

In addition to functional testing, the system implements real-time randomness tests that are automatically executed after each password generation. These include the Monobit Test, Runs Test, Cumulative Sums Test, and Serial Test, all inspired by NIST SP 800-22 randomness test methodologies. To ensure compatibility across different character types and encoding schemes, passwords are first hashed using SHA-256 before the tests are performed on the resulting 256-bit binary sequence. These tests are implemented in JavaScript and provide immediate, visual feedback to the user.

Furthermore, a custom brute-force simulation module was developed using Flask to estimate resistance to offline attacks. This simulator dynamically adjusts its guessing pool based on the actual character types used in the password, and notably increases attack complexity when emojis are included—since the attacker cannot predict their presence. The simulation

either returns a cracked password within a 30-second limit or presents failure statistics, thereby educating users on the practical consequences of their password choices.

As the project evolved, integration and regression tests were conducted following the implementation of features such as emoji fallback, dual password generation, and real-time result updates. These tests ensured continued compatibility between modules and preserved system stability.

Lastly, user interface responsiveness and error handling were evaluated to guarantee that all interactive components behaved predictably, only revealed relevant results when necessary, and provided clear, informative feedback in response to both proper and improper use.

IV. Experimental Results

To assess the performance and reliability of the proposed password generation system, a series of experimental evaluations were conducted. These include brute-force resistance analysis, similarity score comparison based on emoji usage, functional test case execution, and statistical randomness validation. The results collectively demonstrate the effectiveness of emoji-enhanced passwords in increasing complexity, reducing predictability, and maintaining functional integrity across various usage scenarios.

A. Brute-Force Test Results

Brute-force tests were conducted to evaluate how quickly and effectively the system could identify a given password. As shown in Table I, the tests varied based on password length and character type, including comparisons between emoji-supported and non-emoji passwords. Passwords that included emojis took significantly more time or failed to be cracked due to the increased search space. For example, TN-3 and TN-5 were not solved within the time limit, while simpler passwords (e.g., TN-1, TN-4) were cracked quickly.The test pairs TN-2 vs. TN-3, TN-5 vs. TN-6, and TN-8 vs. TN-9 were specifically designed to compare similarity scores under identical time constraints for passwords with and without emoji usage.

TABLE I.

COMPARISON OF BRUTE_FORCE TEST RESULTS

Test No	Password	Character Set	Password Found	Number Of Trials	Duration(s)	Best Guess	Similarity Score
TN-1	&K5	Without emoji	YES	506,082	0,93	&K5	100%
TN-2	r<0	Without emoji	YES	132.665	1,12	r<0	100,00%
TN-3	🌩(5	With emoji	NO	636,056	1,14	a(5	66,07%
TN-4	5G{u	Without emoji	YES	36,498,421	61,58	5G{u	100%
TN-5	>6🚴p	With emoji	NO	54.700.816	72,75	a>6p	66,67%
TN-6	^Lx6	Without emoji	YES	42.891.441	72,75	^Lx6	100%
TN-7	🐂z0*	With emoji	NO	54,700,816	94,43	az0*	60%
TN-8	[jB0	Without emoji	YES	49.681.307	99.18	[jB0	100,00%
TN-9	1C 🤳<	With emoji	NO	54,700,816	99,18	a1C<	54,55%

B. Emoji Impact on Similarity Scores

Additionally, Figure 4.1 illustrates the relationship between password length and similarity score during the password recovery process. This figure demonstrates how longer passwords generally result in lower similarity scores, indicating increased complexity and reduced likelihood of successful recovery through guessing or brute-force methods. It highlights the importance of choosing sufficiently long passwords to enhance security. Meanwhile, Figure 4.2 compares the impact of using emojis versus not using them on the similarity rate. The comparison shows that passwords containing emojis tend to have lower similarity scores compared to those without emojis, suggesting that the inclusion of emojis adds an extra layer of unpredictability and complexity. These results emphasize the security benefits of incorporating emojis into passwords, as they make password guessing and recovery significantly more difficult. Together, these figures provide clear evidence of how password length and character composition influence the overall strength and resilience of passwords against attacks.

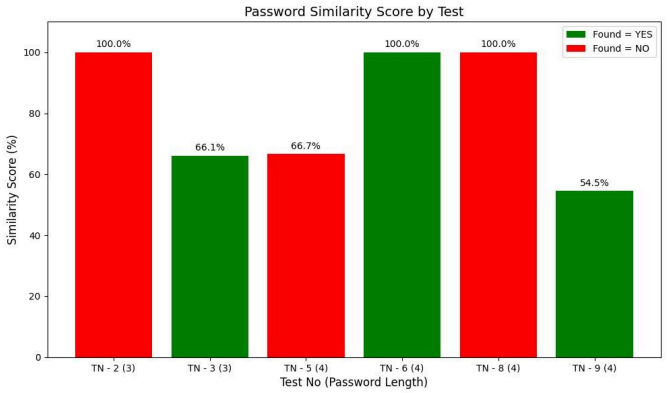

Figure 4.1. Similarity scores for selected test cases
(Comparing TN-2, TN-3, TN-5, TN-6, TN-8, TN-9 test cases)

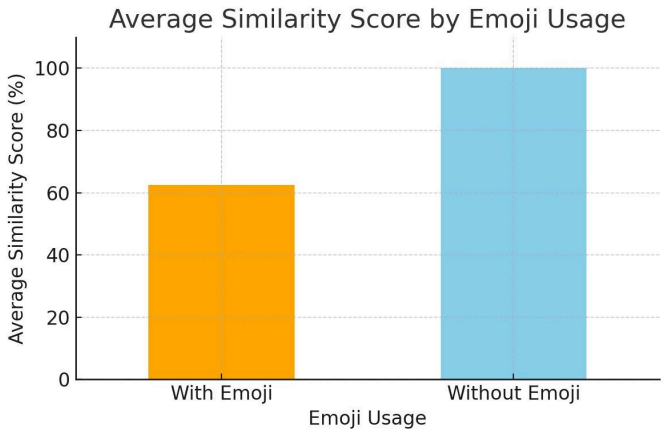

Figure 4.2. Average similarity by emoji usage
(Comparing With Emoji and Without Emoji test cases)

C. Test Scenarios and Success Evaluation

To validate the functional reliability of the proposed password generation system, various structured tests were conducted across common and edge-case usage scenarios. These included emoji-based password creation, input validation, brute-force resistance, randomness evaluation, and interface feedback.

The system successfully generated semantically relevant emojis when meaningful base text was provided, and issued appropriate warnings when required input was missing. It correctly handled invalid configurations, such as length-character mismatches or no character type selection, by auto-completing or preventing password generation with clear alerts.

Under brute-force simulations, the system resisted cracking within time constraints and accurately reported trial counts and duration. It also performed well under load, generating long passwords without performance degradation.

Repeated generations from the same input produced unique results, confirming internal randomness. The system reliably handled inputs with special characters, Turkish letters, and whitespace without failure.

Interface responsiveness was verified through dynamic field visibility, clipboard copying feedback, and consistent UI behavior. Additionally, semantic matching worked effectively even with misspelled inputs, successfully associating intended meanings with relevant emojis.

All test scenarios passed, demonstrating the system's robustness, usability, and preparedness for practical deployment.

D. Randomness Test Results

Passwords were also evaluated with statistical randomness tests (Monobit, Runs, Cumulative Sums, Serial). Emojis and longer lengths improved pass rates across all tests.

TABLE II.

COMPARISON OF RANDOMNESS TEST RESULTS

Test ID	Password	Character Set	Length	Monobit	Runs	Cumulative	Serial
TR - 1	]1📖T	With emoji	4	✅	✅	✅	✅
TR - 2	%1wC	Without emoji	4	❌	✅	✅	✅
TR - 3	)br]lM24	Without emoji	8	❌	✅	✅	✅
TR - 4	J_11a%🏎️d	With emoji	8	✅	✅	✅	✅
TR - 5	5Q!teY7💙[o8#	With emoji	12	✅	✅	✅	✅
TR - 6	_Yg5A4#X?zb2	Without emoji	12	✅	✅	✅	✅

V. CONCLUSIONS AND FUTURE WORK

This study presented a password generation system that combines natural language processing (NLP) with semantic emoji matching. The system analyzes the text entered by the user, identifies grammatical components, and generates passwords by matching them with appropriate emojis. It also supports classical character sets such as uppercase and lowercase letters, digits, and symbols, allowing the creation of complex passwords in compliance with modern security policies.

Security evaluations conducted using the Monobit, Runs, Cumulative Sums, and Serial Tests demonstrated that the generated passwords exhibit statistically sufficient randomness. Additionally, the use of emojis and Unicode characters significantly increased resistance to brute-force attacks compared to traditional text-only passwords. When the

selected character types are insufficient, the system dynamically fills the remaining characters with either emojis or classical characters, depending on user preference. The web-based interface provides users with real-time customization, password testing, and security analysis features.

The findings indicate that grammatical and semantic structures can be effectively used in password generation. Natural language input improves memorability while enabling personalization without compromising security. This shows that NLP techniques are not limited to conventional tasks such as classification or sentiment analysis but can also serve as practical tools in cybersecurity applications.

Currently, the system supports only English language input, has limited mobile compatibility, and relies solely on brute-force modeling for attack simulation. Future work will focus on adding multilingual support, incorporating dictionary-based and AI-assisted attack scenarios, and evaluating usability through structured user testing. Furthermore, AI-powered recommendation mechanisms may be developed to provide personalized suggestions based on user behavior.

In conclusion, this study proposes a practical and user-oriented password generation approach that brings together security, usability, and semantic interaction. It lays a strong foundation for the development of future intelligent, customizable, and secure password management systems.

References

[1] Walid Ali Sulaiman Ali Alothman, Evaluating Passwords User Behavior and the Psychology of Password Management (2019)

[2] Farhana Zaman Glory, Atif U1 Aftab, Olivier Tremblay-Savard, Noman Mohammed, Strong Password Generation Based On User Inputs (2019)

[3] Tehreem Hussain*, Kiran Atta, N. Z. Bawany, Tehreem Qamar,Passwords and User Behavior (2018)

[4] Golla, M., & Dürmuth, M. (2017). EmojiAuth: Quantifying the Security of Emoji-based Authentication. USEC. (p. 4-6)

[5] SSRN (2020). Are Emojis Creating a New or Old Visual Language for New Generations?(p. 56-60)

[6] Rukhin, A. et al. (2010). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (SP 800-22 Rev1a). NIST.

[7] Bishoi, A., et al. (2020). Xorshift Random Number Generators from Primitive Polynomials. Journal of Telecommunications and Information Technology.

[8] Fadhillah, W. et al. (2021). Implementation of the Fisher-Yates Shuffle Algorithm in Exam-Problem Randomization. ResearchGate.

[9] Vaithyasubramanian, S., Prakash, R., & Udhayan, J. (2014). An Analysis of Markov Password Against Brute Force Attack for Effective Web Applications. Middle-East Journal of Scientific Research, 22(4), 5825–5827.

[10] Kjellevand, T., & Rauhut, R. (2018). Evaluating the Security and Usability of Emoji-Based Authentication. Master's Thesis, Norwegian University of Science and Technology (NTNU).

[11] Chen, X., Howard, D., Guggeri, F., & Bosu, A. (2019). SEntiMoji: An Emoji-Powered Learning Approach for Sentiment Analysis in Software Engineering. Proceedings of the 2019 IEEE/ACM 16th International Conference on Mining Software Repositories

[12] Strathmore University. (2021). BITREST Password Manager: Final Documentation Report for Information Systems Project 2. Nairobi, Kenya.

[13] T. Seitz, F. Mathis, and H. Hussmann, “The bird is the word: A usability evaluation of emojis inside text passwords,” in Proc. 29th Australian Conf. Human-Computer Interaction (OzCHI), Brisbane, Australia, (Nov. 2017), pp. 351–359.

[14] N. K. Jha, “An approach towards text to emoticon conversion and vice-versa using NLTK and WordNet,” in 2018 2nd Int. Conf. Data Science and Business Analytics (ICDSBA), Changsha, China, (Sept. 2018), pp. 161–165.

[15] X. Nan, *Password Security Reinforcement via Combining Unicode Character Set*, M.Eng. thesis, Dept. of Computer Sci. and Communications Eng., Waseda Univ., Tokyo, Japan, (Jul. 2024).