

RESEARCH ARTICLE

Beyond Hardware Acceleration: A Heuristic Decision Framework for Optimizing WPA2 Dictionary Attacks

OLEKSII CHALYI^{ID}

Institute of Social Sciences and Applied Informatics, Kaunas Faculty, Vilnius University, 44280 Kaunas, Lithuania

e-mail: oleksii.chalyi@knf.vu.lt

ABSTRACT Offline dictionary attacks against WPA2 remain computationally demanding, particularly in resource-constrained digital forensics environments. While conventional approaches primarily emphasize hardware acceleration, this study investigates heuristic optimization and sorting of static wordlists. Five strategies, Alphabetical, Reverse Alphabetical, Length, N-gram, and a novel Structural method, are evaluated through a three-stage experimental pipeline comprising synthetic string-matching simulations, GPU-accelerated WPA2 hash recovery using Hashcat, and cross-dataset validation. The validation includes 1,000-target stress tests for Complex and Transliterated password categories. The study also examines regional password behaviors associated with Cyrillic-to-Latin transliteration and spatial QWERTY Keyboard Layout Patterns in Cyrillic-dependent regions. Under the evaluated conditions, Alphabetical sorting was the most effective strategy for high-entropy and security-compliant passwords, producing speedups exceeding $2.0\times$ with high statistical stability. Structural sorting performed best for low-entropy Numeric and Dictionary patterns, achieving speedups of up to $4.13\times$. For regionally influenced credentials, N-gram sorting accelerated Transliterated password recovery by $1.55\times$, although its practical value was limited by substantial preprocessing time. The results further demonstrate that dictionary size, password structure, hardware throughput, and preprocessing overhead jointly determine the operational value of each strategy. Based on these findings, an empirically grounded Heuristic Decision Framework is proposed to assist authorized digital forensics practitioners in selecting a wordlist strategy according to the expected target profile, dataset scale, available hardware, and the required balance between preparation cost and Time-to-Crack.

INDEX TERMS WPA2, dictionary attack, network security, password cracking, Hashcat, spatial heuristics.

I. INTRODUCTION

The evolution of wireless communication technologies has fundamentally transformed global connectivity. The advent of the IEEE 802.11 (Wi-Fi) standard, alongside the proliferation of smartphones, tablets, and laptops, has provided users with seamless remote access across the 2.4 GHz, 5 GHz, and 6 GHz frequency bands. However, the widespread adoption of this highly beneficial technology has introduced a broad spectrum of critical vulnerabilities, including data leakage, eavesdropping, phishing, and spoofing [1], [2], [3]. The sheer

variety and volume of these threats have necessitated the establishment of a dedicated cybersecurity domain focused specifically on wireless network security.

Password recovery attacks represent a critical subset of wireless security threats and are primarily aimed at extracting the Wi-Fi Pre-Shared Key (PSK) [4]. These attacks predominantly target WPA2-Personal configurations. According to recent statistics presented at the Wireless LAN Professionals Conference, WPA2 still dominates the landscape, accounting for approximately 50% of networks [5]. In contrast, the WPA3 encryption standard, which mitigates offline dictionary and brute-force attacks, currently has an adoption rate of only 1% among personal users.

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang^{ID}.

To execute a dictionary attack, an adversary captures a .cap file containing the network handshake or PMKID and utilizes specialized software, such as Hashcat, Aircrack-ng, or John the Ripper, to crack the cryptographic hash [6]. Unlike purely algorithmic brute-force attacks that generate character combinations using predefined masks, dictionary attacks rely on massive open-source wordlists to exploit human predictability. According to the 2025 Verizon Data Breach Investigations Report (DBIR), the exploitation of weak credentials remains a defining action in bypassing layered defenses, with brute-force techniques accounting for 56% of action varieties in basic access attacks [7]. In the context of wireless networks, the WPA2 password acts as this type of vulnerable perimeter key. Consequently, mitigating these unauthorized access vectors and enforcing robust password policies constitute fundamental components of a comprehensive Information Security Risk Management (ISRM) framework [8], [9].

However, wireless password recovery is not exclusively the domain of cybercriminals. It is also a critical, legally mandated operation for digital forensics specialists and law enforcement agencies seeking to thwart potential cyberattacks or investigate digital evidence [10], [11], [12]. In forensic scenarios, Wi-Fi password cracking is a computationally nondeterministic task because the complexity of the target password is fundamentally unknown. Although previous research [13] has extensively analyzed the influence of physical hardware architectures, including the CPU, RAM, and GPU, on hash-cracking speeds, this hardware-centric focus often overlooks the operational constraints of field forensics. In real-world investigative scenarios, rapid on-site password recovery frequently relies on the limited computing resources of operational laptops and mobile GPUs rather than high-end cloud clusters. While prior research focuses on maximizing hardware hash-rate capacity, it frequently overlooks the structural efficiency of the data being processed and primarily treats the attacking dictionary as a static, unoptimized input variable.

This research addresses this gap by shifting the focus from hardware acceleration to the heuristic optimization and spatial sorting of wordlists. This study analyzes and compares distinct wordlist sorting methodologies, namely Alphabetical, Length, N-gram, Structural, and Reverse Alphabetical, based on their execution time and recovery efficiency. Furthermore, the impact of strict protocol-based wordlist filtering, specifically the 8–63 character constraint dictated by WPA2 standards, is evaluated empirically. Crucially, this research addresses a substantial gap in the current literature by introducing regional linguistic specificity, with a focus on Cyrillic-dependent demographics. Users in these regions frequently exhibit distinct password-generation habits, such as Cyrillic-to-Latin transliteration and spatial Keyboard Layout Patterns. The latter phenomenon occurs when users type native Cyrillic words while the physical keyboard is set to the standard English QWERTY layout. This study systematically evaluates how specific sorting heuristics can

exploit these uniquely Cyrillic-dependent structural patterns to reduce the Time-to-Crack (TTC).

The proposed methodologies are first evaluated through synthetic string-matching simulations to establish a theoretical baseline, followed by real-world GPU-accelerated empirical validation. The empirical findings are then cross-validated using an additional set of independent open-source wordlists, culminating in a robust 1,000-target stress test specifically for the Complex and Transliterated password categories. This procedure is designed to ensure the validity, reproducibility, and statistical reliability of the research. The ultimate objective of this methodological pipeline is to extract actionable execution metrics and construct an empirically grounded decision framework for forensic practitioners.

Consequently, this research addresses the following research questions (RQs):

- 1) **RQ1:** Does strict protocol-based wordlist filtering provide a measurable speed boost during GPU-accelerated WPA2 cracking, and can specific sorting algorithms actively decrease the TTC? Most importantly, is the preprocessing time required to generate these sorted lists (T_{prep}) effectively compensated by the actual cracking-time advantage (T_{crack}) under hardware constraints?
- 2) **RQ2:** Are the obtained performance results statistically stable across varying datasets and password complexities? Can these empirical execution metrics be synthesized to formulate an empirically grounded, profile-aware Heuristic Decision Framework that optimizes wordlist selection for forensic operations under specific hardware assumptions?
- 3) **RQ3:** Do targeted sorting methodologies, such as probabilistic N-gram or spatial heuristics, provide a quantifiable computational advantage when recovering passwords heavily influenced by regional semantics, specifically those utilizing Cyrillic transliteration and keyboard-layout behaviors characteristic of users in Cyrillic-dependent regions?

The findings of this research deliver actionable, data-driven intelligence for digital forensics specialists seeking to optimize limited computational time during field operations. They also provide evidence-based security recommendations for end users seeking to fortify their wireless networks against advanced offline attacks.

II. RELATED WORK

This section reviews the existing literature surrounding wireless network vulnerabilities and password recovery techniques. The analyzed studies approach the reduction of Time-to-Crack and security assessment through various complementary methodologies. To provide a structured and comprehensive overview, the related works are classified into three primary categories: 1) password guessing models; 2) hardware-accelerated cracking; and 3) regional password patterns.

A. PASSWORD GUESSING MODELS

The fundamental vulnerability exploited by dictionary attacks is human predictability. Taneski et al. demonstrated that careless security behaviors persist globally, including pervasive password reuse and misunderstanding of cryptographic strength [14]. Pilar et al. further showed that these behaviors are driven not primarily by cognitive decline but by “password fatigue,” namely the large number of systems and circumstances in which users must maintain credentials [15]. This reliance on memorability leads to highly predictable structural patterns. For example, Awad et al. identified a widespread tendency to capitalize only the first letter and append numbers to the end of a password, thereby substantially reducing its effective entropy [16].

To exploit these predictability flaws mathematically, researchers have developed advanced probabilistic models. Houshmand et al. advanced Probabilistic Context-Free Grammar (PCFG) password cracking by integrating keyboard spatial patterns and multiword combinations into the underlying grammars, significantly improving cracking efficiency during prolonged password-recovery operations [17].

Although these generative models are effective, they introduce a substantial operational bottleneck. Generating billions of guesses through complex grammars is inefficient for rapid, field-level WPA2 offline attacks, particularly because the protocol’s strict 8–63-character constraint already narrows the operational scope. Furthermore, existing probabilistic models focus primarily on generating new combinations rather than heuristically optimizing lightweight breach dictionaries that already exist. The present research bridges this gap by applying established structural insights directly to the pre-sorting of static wordlists, thereby optimizing the attack vector for immediate TTC reduction without the overhead associated with generating massive grammar-based datasets.

It is important to distinguish the present approach from several related password-guessing paradigms. Static wordlist sorting reorders a fixed set of existing candidates without creating new guesses; the Alphabetical, Reverse Alphabetical, Length, and Structural strategies evaluated in this study belong to this category. A frequency-ranked dictionary is also static, but its order reflects observed password prevalence, as in the original RockYou baseline. In contrast, rule-based approaches generate additional candidates by applying transformations to seed words, whereas probabilistic models generate or prioritize guesses according to learned structural probabilities. Markov and N-gram methods rank character sequences using conditional or substring probabilities and may be either generative or ranking-based. The N-gram strategy used in this study is non-generative: it assigns bigram-derived scores only to candidates already present in the filtered wordlist and then reorders that fixed candidate set. Consequently, the study evaluates candidate-order optimization rather than candidate-space expansion.

B. HARDWARE-ACCELERATED CRACKING

Moissinac et al. investigated WPA2 key-generation vulnerabilities and found that, although hybrid authentication mechanisms can increase computational overhead, offline dictionary attacks remain a persistent threat against hardened WPA2 configurations until WPA3 is widely adopted [18]. Building on this work, Prodani and Rista demonstrated a strong correlation between hardware specifications and cracking speed, emphasizing that dictionary attacks are substantially faster than algorithmic brute-force methods when the target password is present in the predefined wordlist [19].

The significance of parallel processing was further quantified by Alkhwaja et al., whose empirical benchmarks revealed substantial speedups when discrete GPUs and multicore CPU scheduling were used [20]. This hardware dependency was corroborated by Chalyi, who confirmed that, although CPU architecture affects baseline performance, raw GPU computational power has a substantially greater influence on hash-cracking efficiency than other system parameters, such as memory capacity and channel configuration [13].

Collectively, these studies indicate that high-end parallelized hardware significantly accelerates WPA2 hash cracking. However, a critical limitation remains throughout the existing literature: the attack dictionary is generally treated as a static, unoptimized input variable. Although prior research focuses on maximizing hardware hash-rate capacity, it overlooks the structural efficiency of the data being processed. The present research addresses this gap by shifting the focus from hardware acceleration to the heuristic optimization and spatial sorting of wordlists, with the objective of further reducing TTC independently of the physical device.

C. REGIONAL PASSWORD PATTERNS

Although structural models and hardware determine the raw mechanics of an attack, the composition of target passwords is strongly influenced by demographic, linguistic, and cultural backgrounds. Han et al. demonstrated that linguistic background fundamentally alters password topography, observing that users instinctively rely on familiar phonetic patterns, such as Chinese Pinyin representations, rather than standard English vocabulary. Integrating these regional composition rules into password-guessing models significantly increased cracking efficiency [21].

This dependency extends to more granular geographical and social contexts. He et al. confirmed that geographical factors rooted in local dialects and naming conventions, together with the specific nature of an online service, play a significant role in determining a dataset’s resistance to password-cracking techniques [22]. Beyond geography, He et al. found that isolating datasets according to specific social behaviors or religious affiliations reveals concentrated similarities in popular passwords, structural patterns, and

character distributions. Incorporating this targeted sociocultural information into password-generation rules substantially improved cracking rates against the corresponding demographics [23].

Collectively, these studies establish a critical premise: contextualizing an attack dictionary through regional, linguistic, and cultural profiling can produce exceptionally high password-recovery rates. However, while Asian and Western linguistic patterns have been extensively analyzed, a substantial gap remains regarding Cyrillic-dependent demographics. In particular, vulnerabilities associated with Cyrillic-to-Latin transliteration and spatial Keyboard Layout Patterns, in which users type native Cyrillic words using a standard English QWERTY keyboard layout, remain largely unexplored. The present research directly addresses this gap by evaluating how specific sorting heuristics can exploit these uniquely Cyrillic-dependent structural patterns to reduce TTC for WPA2-protected networks.

III. METHODOLOGY

This section presents the comprehensive methodology used to evaluate the impact of wordlist structuring on WPA2 dictionary attacks. First, the selection and preprocessing of the primary attack and validation datasets are described in Section III-A. Next, the systematic extraction and categorization of target passwords, including cross-linguistic spatial analysis, are detailed in Section III-B. Section III-C introduces the heuristic sorting strategies developed and evaluated in this study. The hardware environment is then established in Section III-D, followed by a comprehensive description of the three-phase experimental setup, encompassing synthetic simulations and real-world GPU-accelerated execution, in Section III-E. The ultimate objective of this methodological pipeline is to extract actionable execution metrics, including the trade-off between preparation cost and TTC, and to construct an empirically grounded decision framework for digital forensics practitioners.

A. DATASETS AND PREPROCESSING

The widely recognized RockYou.txt dictionary [24] was selected as the primary dataset for this research. It contains 14,344,391 lines and is natively ordered according to password popularity, which makes it highly effective for guessing frequently used passwords. However, under the WPA2 specification, a valid Wi-Fi passphrase must contain between 8 and 63 characters [25]. Consequently, the original RockYou wordlist contains millions of invalid entries that consume computational resources during a dictionary attack without contributing valid candidate passwords. To address this limitation, a preprocessing phase was performed to remove every password that did not satisfy the WPA2 length constraint.

To validate the proposed sorting methods, five additional open-source wordlists were selected. Although these datasets are hosted in informal repositories, they were deliberately

included because they represent unsterilized, real-world credential leaks associated with specific online services and breach collections. Their raw composition preserves human password-creation biases and realistic entropy distributions, making them suitable for cross-dataset validation. The following datasets were used, with line counts reported after removing passwords outside the 8-to-63-character range:

- 1) **Wifi-Cracker** by sina-programer: 488,132 lines [26];
- 2) **Xato-net-10-million-passwords** by danielmiessler: 39,329 lines [27];
- 3) **Password-List** by Python-Fa: 28,462 lines [28];
- 4) **Password-Checker**, containing the alleged Gmail passwords, by breezy-codes: 2,240,055 lines [29];
- 5) **OneListForAll** by six2dez: 1,525,753 lines [30].

B. TARGET SELECTION AND CATEGORIZATION

To ensure a robust and statistically meaningful empirical evaluation, a custom dataset of 600 unique passwords was extracted from the original unfiltered RockYou dictionary. The targets were divided equally among six structural categories, with 100 passwords assigned to each category. A custom Python script employing strict regular-expression matching automated the extraction process and reduced human selection bias [31], [32]. In addition, a semantic dictionary of more than 200 frequently occurring Cyrillic base words was compiled from personal names, family-related terms, geographical locations, popular brands, and Internet slang to model regional password-creation behavior.

The selection criteria, regular expressions, numbers of valid matches in the baseline dictionary, and representative examples were defined as follows:

- 1) **Pure Numeric:** Decimal strings satisfying the WPA2 length constraint of 8 to 63 characters.
 - Regular expression: `^\d{8,63}$`
 - Total matches: 1,417,122
 - Example: 92063065406
- 2) **Dictionary Word:** Lowercase Latin strings of 8 to 63 characters without numeric or special-character affixes.
 - Regular expression: `^[a-z]{8,63}$`
 - Total matches: 2,404,951
 - Example: ilovegalo
- 3) **Transliterated:** Frequent Cyrillic-to-Latin semantic bases, such as *mama*, *brat*, and *papa*, with optional numeric prefixes, suffixes, or both.
 - Regular expression: `^\d*(mama|brat|papa|\ldots)\d*$`
 - Total matches: 18,542
 - Example: papa4513
- 4) **Keyboard Layout Pattern:** Strings produced by entering frequent Cyrillic words or regional names through the corresponding physical keys of an English QWERTY layout. For example, *vfvf* corresponds to the Cyrillic word “*мама*”. Optional numeric prefixes and suffixes were permitted.

- Regular expression:
 $\wedge d^* (gf hj km | vf vf | gf gf | \ldots) d^* \$$
 - Total matches: 105
 - Example: gfhjkm00
- 5) **Hybrid (Word + Digits)**: Strings beginning with one uppercase letter, followed by at least four lowercase letters and ending with at least two decimal digits.
- Regular expression:
 $\wedge [A-Z] [a-z] \{4, \} \wedge \{2, \} \$$
 - Total matches: 115,716
 - Example: Raffi1993
- 6) **Complex**: Strings of 12 to 63 characters containing at least one lowercase letter, one uppercase letter, one decimal digit, and one character from !@#\$%^&*.
- Regular expression:
 $\wedge (?. * [a-z]) (?. * [A-Z]) (?. * \wedge) (?. * [!@ \# \$ \% \wedge \& *]) . \{12, 63\} \$$
 - Total matches: 5,580
 - Example: RiKrDo260106!!

To resolve overlaps between regular expressions, passwords were evaluated sequentially using the following category precedence: Numeric, Dictionary Word, Transliterated, Keyboard Layout Pattern, Hybrid, and Complex. A first-match-wins rule was applied, meaning that each password was assigned to the first matching category and was not considered for subsequent categories. After categorization, all eligible passwords within each category were aggregated, and randomized sampling without replacement was used to select exactly 100 targets per group. Global deduplication additionally eliminated any remaining intra-category duplicates.

C. SORTING STRATEGIES

Because the WPA2 protocol imposes a minimum passphrase length of eight characters, and because cognitive password fatigue is well documented [15], [16], a substantial subset of user-generated Wi-Fi credentials may be expected to contain exactly eight characters. This behavioral and protocol-level predictability raises the question of whether a frequency-ordered dictionary provides the most computationally efficient baseline for WPA2 password recovery.

Based on this premise, this study proposes a novel sorting strategy termed *Structural*. Its hierarchical prioritization is grounded not only in cognitive password fatigue but also in the mathematical minimization of the expected search space. Let K denote the keyspace associated with a password of length L and alphabet size $|A|$. The keyspace is defined as

$$K = |A|^L. \quad (1)$$

The Structural algorithm prioritizes subsets with the smallest alphabet cardinalities. Priority 1 contains purely numeric strings, for which $|A| = 10$. Priority 2 contains lowercase alphabetic strings, for which $|A| = 26$. Priority 3 contains mixed alphanumeric strings, for which $|A| = 62$. Priority 4 contains strings drawn from the full printable ASCII set,

for which $|A| = 95$. By testing the lower-cardinality and behaviorally dense subsets first, the heuristic is designed to reduce the expected TTC for commonly occurring classes of user-generated credentials.

For comparative evaluation, six modified versions of the original RockYou dataset were created:

- 1) **RockYou Filtered**: The RockYou wordlist preserved in its original order but strictly filtered according to the 8-to-63-character WPA2 requirement.
- 2) **Alphabetical**: A RockYou variant in which passwords are sorted in ascending standard ASCII lexicographic order.
- 3) **Reverse Alphabetical**: A variant using the same ASCII lexicographic criterion as the Alphabetical strategy but sorted in descending order.
- 4) **Length**: A variant in which passwords are sorted by ascending length, beginning with eight-character strings.
- 5) **N-gram Probability**: A variant in which passwords are ranked dynamically according to the probabilities of their character sequences, using bigrams to model behavioral regularities in password creation. The statistical language model is generated directly from the length-filtered dataset before sorting by calculating the frequency of every observed bigram. Consequently, the resulting probabilities reflect the inherent structural distribution of the active wordlist rather than an external language corpus.
- 6) **Structural**: The proposed sorting method based on the four character-composition priorities defined above.

Each modified RockYou wordlist was purged of entries shorter than 8 characters or longer than 63 characters. This preprocessing step produced a finalized search space of 9,609,254 candidate passwords per modified dictionary. In total, seven wordlists were used in the comparative analysis: the six modified variants and the original unfiltered RockYou wordlist.

Each sorting strategy produces a single fixed global ordering of the candidates in a given input wordlist. This ordering is independent of the assumed target-password category and is used unchanged for all target categories; category labels are applied only during the subsequent analysis of recovery performance. For the N-gram strategy, a separate model is learned from each input wordlist, but the resulting ordering remains fixed across all target categories evaluated against that wordlist.

D. HARDWARE SELECTION

The empirical hash-cracking experiments were executed in a Windows 11 24H2 environment. While contemporary digital forensics laboratories frequently deploy advanced multi-GPU cloud clusters, the testing hardware was selected specifically to emulate the computational constraints faced by field operatives using standard-issue laptops. According to recent global consumer hardware distributions, the NVIDIA

GeForce RTX 4060 Laptop GPU represents one of the most widely adopted mid-range mobile GPUs, making it a realistic baseline for such devices [33].

However, executing the expanded experimental pipeline required processing 25,200 individual hash-cracking instances (600 target hashes $\times$ 7 sorting methodologies $\times$ 6 datasets). Based on the average recovery time, this volume corresponds to approximately 50 h of uninterrupted operation at 100% GPU utilization. As documented in thermal modeling research [34], [35], [36], modern portable computing platforms rely on dynamic voltage and frequency scaling (DVFS) to mitigate overheating during sustained high-power workloads. Subjecting a standard laptop to approximately 50 h of severe thermal stress would trigger hardware-level frequency scaling. Because DVFS dynamically adjusts GPU clock speeds to prevent overheating and hardware degradation, using a physical laptop would introduce unpredictable variations in processing speed, thereby reducing the accuracy, consistency, and reproducibility of the TTC measurements.

To isolate and measure the algorithmic impact of heuristic sorting without interference from DVFS throttling, a desktop-class AMD Radeon RX 5700 XT GPU was used. The RX 5700 XT delivered a stable WPA2/PMKID hash rate of approximately 450 kH/s, closely matching the reported peak performance of the target RTX 4060-class baseline, approximately 458,501 H/s [38]. This configuration also aligns with the established 400 kH/s baseline used to characterize a contemporary mid-range forensic system [37]. By maintaining thermal stability and consistent timekeeping throughout the approximately 50-h execution period, the controlled desktop environment established a reproducible upper-bound baseline. Consequently, reductions in TTC observed in this controlled setup can be evaluated independently of the additional performance degradation that may occur on thermally constrained mobile hardware.

The synthetic experiments and wordlist-sorting phases were executed on an AMD Ryzen 7 7700 processor capped at 5.0 GHz to provide a stable and reproducible CPU baseline comparable to contemporary high-performance mobile processors. Hashcat version 7.1.2 was used for all empirical cracking experiments. The workload profile was set to $-w\ 3$ to maintain high GPU utilization, yielding a stable, unthrottled baseline processing speed of approximately 450 kH/s across the evaluated scenarios.

E. EXPERIMENTAL SETUP

The experimental evaluation was conducted in three phases to systematically assess the influence of wordlist structure on password-recovery performance.

1) PHASE 1: SYNTHETIC TESTING

The first phase served as a theoretical prescreening mechanism. The five open-source validation datasets described in Section III-A were used as target lists, while the modified RockYou dictionaries served as the attack wordlists.

Password recovery was simulated through direct computational string matching rather than cryptographic hash computation. Because linear string matching is computationally distinct from GPU-parallelized hash cracking, this procedure was used only as a rapid preliminary indicator. It enabled large-scale analysis of how the sorting strategies affected search-space traversal before hardware-dependent variables were introduced.

Although this synthetic simulation provides an efficient preliminary indicator, it abstracts the hardware mechanisms involved in Hashcat execution, including GPU memory input/output, candidate batching, and workload scheduling. Accordingly, TTC was not treated as a valid evaluation metric during this phase because string-matching throughput differs substantially from real-world cryptographic hashing. Instead, the evaluation used a mathematical relationship between the percentage of the search space traversed and the percentage of recoverable targets identified.

The principal conclusions of this study are based on the empirical Hashcat results obtained in Phase 2. The purpose of Phase 1 was to determine whether isolated mathematical traversal curves could approximately predict the GPU hash-cracking efficiencies observed during empirical execution. To support this comparison, the synthetic evaluation included both the five external validation datasets and a targeted micro-evaluation. The same 600 structurally categorized target passwords described in Section III-B and designated for empirical GPU testing were subjected to theoretical string-matching prescreening. The resulting category-specific recovery curves established a mathematical baseline for direct comparison with the empirical TTC measurements obtained in Phase 2. For this study, a sorting strategy was classified as optimal during prescreening when it recovered more than 80% of identifiable targets within the first 50% of the traversed search space.

The conceptual transfer between the two stages is based on relative candidate position rather than execution time. In the synthetic stage, the ordering strategies are evaluated by measuring the fraction of the prepared candidate list that must be traversed before the selected targets are located. In the empirical stage, the same ordering procedures are applied to the controlled validation dictionaries, and the resulting wordlists are evaluated using Hashcat. Where an identical prepared wordlist is used in both stages, the synthetic traversal and Hashcat evaluation operate on the same candidate sequence; otherwise, the synthetic results are interpreted as category-level expectations rather than direct TTC predictions. Within a fixed empirical wordlist, a target appearing earlier in the candidate sequence is generally expected to produce a shorter TTC under approximately stable GPU throughput. However, this relationship is not assumed to be exact or strictly proportional because process initialization, GPU batching, memory transfer, workload scheduling, and termination latency introduce hardware- and implementation-dependent overhead. Phase 2 therefore serves as the empirical validation of whether the relative

ordering tendencies observed during synthetic prescreening persist under actual WPA2 evaluation. Accordingly, all conclusions regarding practical WPA2 cracking performance are derived from the measured Hashcat executions in Phase 2.

2) PHASE 2: EMPIRICAL HASH CRACKING

The second phase transitioned to real-world GPU-accelerated hash cracking. The 600 categorized target passwords described in Section III-B were tested against the original RockYou dictionary and its six modified variants.

Each target password was processed using the SSID Chalyi and encoded in the .hc22000 format recognized by Hashcat. The PMKID record followed the structure

WPA*01*PMKID*MAC_AP*MAC_STA*SSID_HEX***.

In this structure, PMKID denotes the target hash. The access-point and station MAC addresses were assigned the fixed values 112233445566 and AABBCDDDEEFF, respectively, and the SSID was set to Chalyi. The principal computational cost arises from the IEEE 802.11i key-derivation procedure, which uses PBKDF2-HMAC-SHA1 to deliberately increase the cost of password guessing [39], [40].

The PMKID targets were generated using a custom standards-compliant Python implementation. For each target password, the pairwise master key (PMK) was derived using PBKDF2-HMAC-SHA1 with the SSID as the salt, 4,096 iterations, and a 256-bit output. The corresponding PMKID was computed as the first 128 bits of HMAC-SHA1(PMK, "PMK Name" || MAC_{AP} || MAC_{STA}). A dataset-wide post-generation check confirmed that all 600 records contained structurally valid 128-bit PMKID values and that the 600 target passwords produced 600 distinct PMKIDs. Cryptographic validity was independently verified through successful recovery of the corresponding known passwords using Hashcat mode 22000. The PMK is derived as

$$\text{PMK} = \text{PBKDF2-HMAC-SHA1}(P, S, 4096, 256), \quad (2)$$

where P is the candidate passphrase, S is the SSID, 4096 is the iteration count, and 256 is the derived-key length in bits. The requirement to perform 4,096 iterations for every candidate emphasizes the need for the controlled GPU-accelerated environment described in Section III-D.

To ensure precise measurement while remaining within the approximately 50-hour hardware-execution budget, a custom automation script controlled the Hashcat processes. The script continuously monitored Hashcat's standard output, recorded the real-time processing rate in kH/s, and forcefully terminated each process immediately after a successful match was reported. The recorded stopping time was used to calculate the empirical TTC.

The primary evaluation criteria in this phase were empirical TTC and recovery-time stability. Rather than relying on theoretical estimates, conclusions concerning algorithmic efficiency were derived from the observed distributions of

recovery times. All execution metrics were stored in a master dataset. An automated processing pipeline cleaned the observations and calculated the minimum, maximum, arithmetic mean, median, first quartile (Q1), third quartile (Q3), and interquartile range (IQR) for each combination of password category and sorting strategy. Median TTC was used for category-specific performance comparisons and speedup calculations, whereas mean TTC was used to summarize aggregate recovery performance across password categories. Dispersion and stability were evaluated using Q1, Q3, and the IQR.

For the hardware configuration used in this study, operational stability thresholds were defined using the observed IQR distributions. Under the controlled 450 kH/s baseline, an IQR below 2.5 s was classified as *high stability*, indicating that recovery times remained consistent across target strings. An IQR from 2.5 to 6.0 s was classified as *moderate stability*. An IQR above 6.0 s was classified as *low stability*, indicating that recovery time varied substantially with the character composition and position of the individual password.

3) PHASE 3: RESULTS VALIDATION

To evaluate the generalizability of the sorting strategies across heterogeneous datasets, the empirical findings from Phase 2 were cross-validated using the five independent wordlists introduced in Phase 1, including Xato-net and Wifi-Cracker.

A controlled target-injection procedure was implemented to ensure that all sorting strategies were evaluated under equivalent spatial conditions. First, the validation script determined whether each benchmark target occurred naturally in a validation dataset. Naturally occurring targets retained their original line positions. Targets absent from a dataset were injected at uniformly randomized line indices in the original unsorted data. This randomized placement prevented systematic bias in baseline recovery positions. Only after target injection were the datasets subjected to WPA2 length filtering and the respective sorting methods. Consequently, changes in TTC following preprocessing could be attributed primarily to the sorting strategy rather than to unequal initial target placement.

A secondary validation procedure was conducted to assess the statistical adequacy of the initial sample size of $N = 100$ targets per category. This robustness analysis focused on two password categories, each selected for a distinct methodological reason. First, the Complex category was included because it reflects a conventional password-composition model that remains widely represented in enterprise password policies [41], [42]. Second, the Transliterated category was included to validate the distribution of regional linguistic heuristics because the initial Phase 2 results exhibited comparatively high variance, as measured by the IQR. This second analysis was intended to determine whether the conclusions concerning Cyrillic-dependent password behavior remained stable with a substantially larger sample.

The decision to limit the expanded 1,000-target stress tests to these two categories was based on the Phase

2 dispersion results. The best-performing heuristics for the Numeric, Dictionary Word, and Hybrid categories exhibited high stability, with IQR values below 2.5 s, using the initial 100-target samples. Increasing the sample sizes for these categories would therefore require substantial additional computation while offering limited improvement in statistical insight. An expanded evaluation of the Keyboard Layout Pattern category was not feasible because the unfiltered baseline dataset contained only 105 valid naturally occurring matches.

For both selected categories, expanded datasets containing 1,000 unique passwords were evaluated against the baseline RockYou dictionary and the respective best-performing sorting strategies. This procedure tested whether the observed performance metrics remained stable and representative at a larger scale. For the Transliterated stress test, the GPU was additionally operated under an overclocked profile that achieved approximately 480 kH/s. This supplementary condition assessed whether the proportional benefit of the selected sorting heuristics remained stable under increased hardware throughput.

IV. RESULTS

This section presents the empirical results obtained across all testing phases. In the first stage, the original `RockYou.txt` dataset was filtered according to the WPA2 password-length requirements. This preprocessing step removed 4,735,137 noncompliant entries, corresponding to 33% of the original dataset, in only 1.17 s. The resulting reduction in the effective search space provided an immediate basis for accelerating the subsequent cracking phase.

Subsequently, multiple modified wordlists were generated from the filtered dataset. To determine whether the computational overhead of offline dictionary generation could negate the time saved during hash cracking, the execution times of all filtering and sorting methods were evaluated. The Length, Alphabetical, and Reverse Alphabetical strategies imposed minimal preprocessing overhead, requiring only 0.53, 0.29, and 0.28 s, respectively. Structural sorting required 2.80 s because the algorithm evaluated and categorized each password according to four distinct heuristic rules. N-gram sorting incurred the highest preprocessing cost, requiring 29.05 s, because it involved constructing a statistical language model and calculating bigram probabilities across millions of strings.

A. SYNTHETIC TESTING

In this phase, synthetic tests were conducted using the selected wordlists as target datasets and the modified versions of `RockYou.txt` as attacking dictionaries. Fig. 1 presents the aggregated performance of all six wordlist variants across the five target datasets by plotting the average recovery rate against the proportion of the search space traversed.

Fig. 1 shows that, within the first 40% of the traversed search space, the original and Filtered RockYou wordlists, together with the Length-sorted wordlist, provided the most

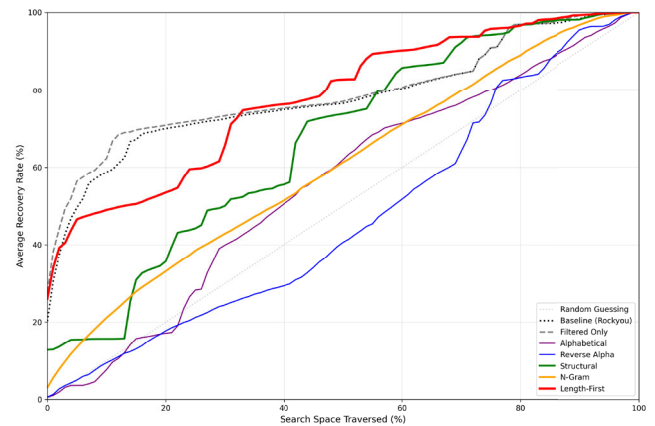


FIGURE 1. Aggregate performance averaged across five target datasets.

favorable balance between password recovery and search-space traversal. At the average recovery level of 60%, the original and Filtered RockYou wordlists achieved the best results. At the 80% recovery level, however, the Length-sorted wordlist outperformed the remaining strategies, followed by the original RockYou, Filtered RockYou, and Structural wordlists. These four strategies subsequently reached 100% recovery in the same order. In the aggregated results, the Alphabetical and Reverse Alphabetical strategies exhibited the weakest overall performance. The N-gram strategy demonstrated moderate performance and occupied an intermediate position among the evaluated approaches.

In the subsequent stage, targeted prescreening was conducted using the same 600 structurally categorized passwords, with 100 passwords per category, that were designated for the real-world GPU experiments. Fig. 2 illustrates the theoretical search-space efficiency of each wordlist strategy for the six password categories.

For the Numeric password category, the Structural and Alphabetical strategies achieved the highest efficiency, recovering 100% of the targets within the first 20% of the traversed search space. The N-gram strategy also performed strongly, recovering approximately 80% of the targets by the 50% traversal point. By contrast, the original RockYou, Filtered RockYou, and Reverse Alphabetical wordlists produced similarly low recovery efficiency.

For the Dictionary password category, only the Structural strategy demonstrated high efficiency, reaching a 100% recovery rate within the first 40% of the traversed search space. The remaining wordlists exhibited similarly weak performance and required traversal of approximately 60% of the search space to recover 80% of the targets.

For the Transliterated password category, most sorting strategies and the original RockYou baseline demonstrated strong performance, recovering 80% of the targets within the first 40% of the traversed search space. In contrast, the Alphabetical and Structural strategies performed poorly in this scenario.

For the Keyboard Layout category, which comprised Cyrillic words typed using an English keyboard layout, the

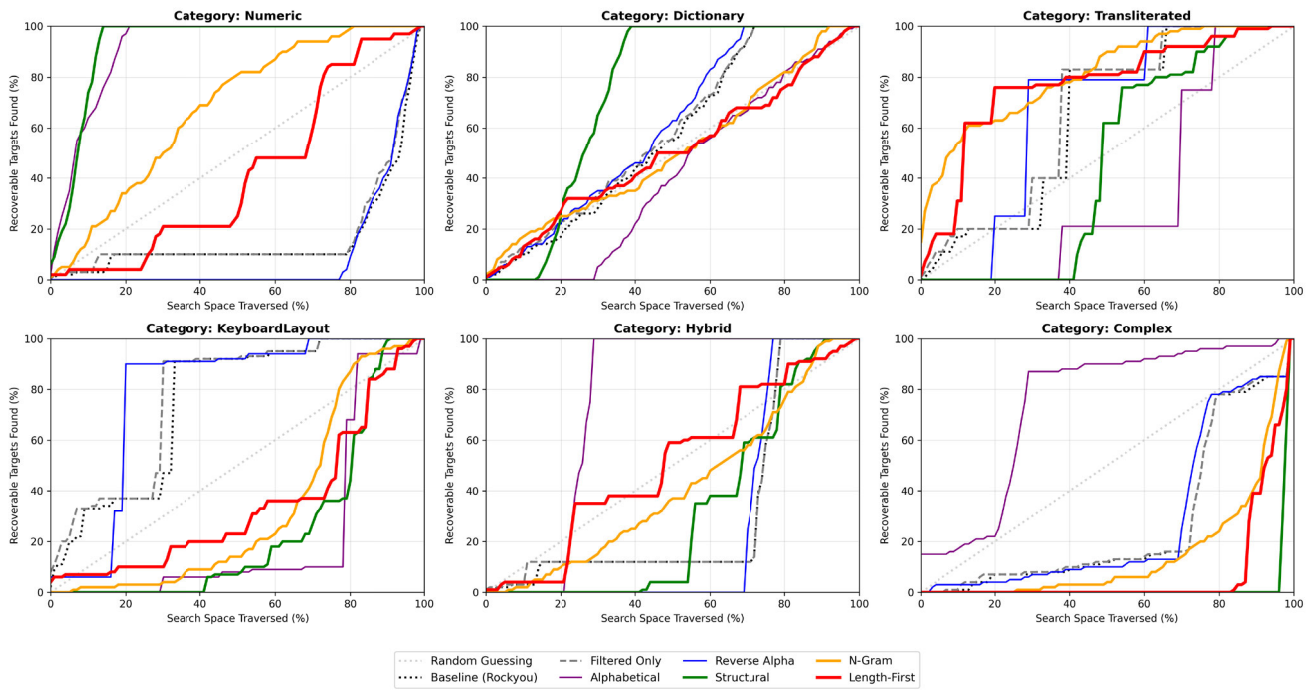


FIGURE 2. Theoretical prescreening results showing search-space efficiency by target-password category.

Reverse Alphabetical strategy achieved the highest efficiency, recovering 90% of the targets within the first 20% of the traversed search space. The original RockYou wordlist and its Filtered variant also performed well, reaching 90% recovery by the 30% traversal point. The remaining strategies performed poorly and required traversal of approximately 75%–80% of the search space to recover 80% of the targets.

For the Hybrid password category, the Alphabetical strategy substantially outperformed the other approaches, recovering 100% of the targets within the first 25% of the traversed search space. All remaining strategies, including the original RockYou baseline, exhibited low efficiency and required traversal of at least 65% of the search space to recover 80% of the targets.

For the Complex password category, the observed trend closely resembled that of the Hybrid category. In this case, the Alphabetical strategy recovered 87% of the targets within the first 25% of the traversed search space. All remaining strategies performed poorly and required traversal of approximately 80% of the search space to reach an 80% recovery rate.

B. REAL-WORLD EXPERIMENTS

A custom Python script was used to generate the target files in .hcc2000 format. Each file contained the SSID, PMKID, and assigned access-point and station MAC addresses. Successful ingestion and parsing by Hashcat confirmed the structural validity of the generated target hashes. Fig. 3 summarizes the median Time-to-Crack achieved by the evaluated

wordlist strategies across the six password categories, with greener cells indicating shorter recovery times.

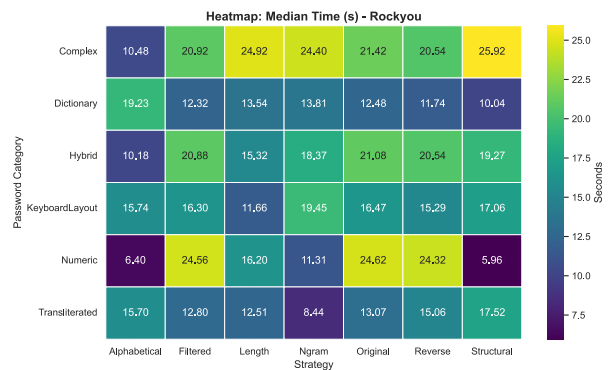


FIGURE 3. Median TTC achieved by the Original RockYou wordlist and its sorted variants across the six password categories.

Alphabetical sorting achieved the lowest TTC for the Complex (10.48 s) and Hybrid (10.18 s) categories and also performed efficiently for Numeric targets (6.40 s). However, it produced the highest TTC for the Dictionary category (19.23 s) and moderate values for the Transliterated (15.70 s) and Keyboard Layout Pattern (15.74 s) categories.

The Original and Filtered RockYou wordlists demonstrated nearly identical performance, with the Filtered variant providing a marginal advantage. The two baselines produced TTC values of 12.48 and 12.32 s, respectively, for Dictionary passwords; 21.42 and 20.92 s for Complex passwords;

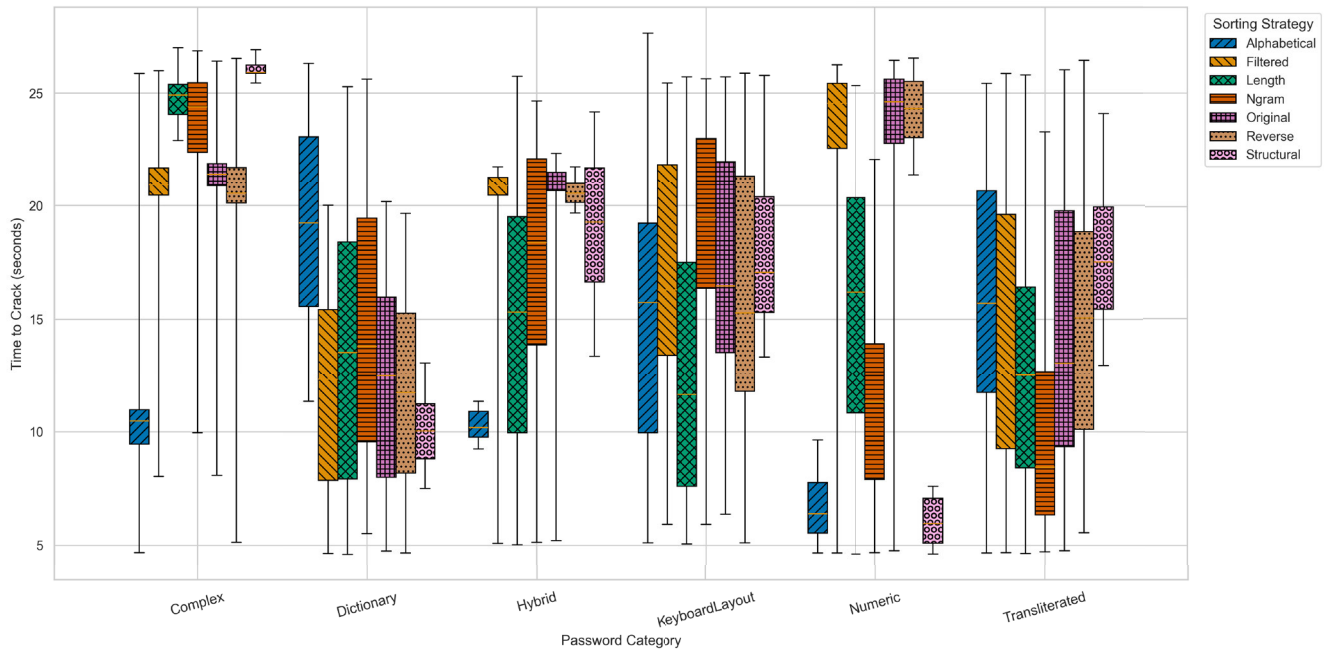


FIGURE 4. TTC distributions for the Original RockYou wordlist and its sorted variants across the six password categories.

13.07 and 12.80 s for Transliterated passwords; and 16.47 and 16.30 s for Keyboard Layout Patterns. Conversely, they produced the highest TTC values among the evaluated strategies for Numeric targets, at 24.62 and 24.56 s, and for Hybrid targets, at 21.08 and 20.88 s.

Length sorting achieved the lowest TTC for the Keyboard Layout Pattern category (11.66 s) while producing moderate results across the remaining categories. N-gram sorting achieved the lowest TTC for Transliterated targets (8.44 s) but performed poorly for Keyboard Layout Patterns (19.45 s) and Complex passwords (24.40 s). It produced moderate TTC values for Dictionary (13.81 s) and Hybrid (18.37 s) passwords and a comparatively low TTC for Numeric targets (11.31 s).

Reverse Alphabetical sorting produced a high TTC for Numeric targets (24.32 s) and moderate results across the remaining categories. The proposed Structural strategy achieved the lowest TTC for Numeric (5.96 s) and Dictionary (10.04 s) targets, together with moderate results for Keyboard Layout Patterns (17.06 s) and Hybrid passwords (19.27 s). However, it produced the highest TTC for Complex (25.92 s) and Transliterated (17.52 s) passwords.

To identify the strongest overall strategy, the mean TTC across all six password categories was calculated for each wordlist configuration. Alphabetical sorting achieved the lowest aggregate mean TTC of 12.96 s. Length sorting followed at 15.69 s, while N-gram and Structural sorting both achieved 15.96 s. Reverse Alphabetical sorting produced the highest mean TTC among the custom strategies, at 17.92 s. Nevertheless, every custom sorting strategy achieved a lower

mean TTC than the Original (18.19 s) and Filtered (17.96 s) RockYou baselines.

Although mean TTC provides an aggregate performance measure, the stability and dispersion of the recovery times must also be evaluated. Accordingly, the minimum, maximum, first quartile (Q1), third quartile (Q3), and interquartile range (IQR) were calculated for every combination of password category and wordlist strategy. Fig. 4 presents the corresponding TTC distributions.

C. RESULTS VALIDATION

To validate the findings, the five datasets originally used during the synthetic phase were repurposed as attacking wordlists. The target password hashes remained identical to those used in the real-world experiments. Individual heatmaps and boxplots detailing the performance of each tested wordlist against the target hashes are provided in the supplementary material as Figs. S1–S10, while Fig. 5 presents the averaged results of the cross-validation experiments.

As shown in Fig. 5, performance on the smallest datasets, including Xato-net and Python-Fa, plateaued at approximately 4.25–4.32 s across all sorting methods. By contrast, visible reductions and variations in average cracking time were observed for larger datasets, including OneListForAll and Password-Checker. When the cross-validation results were aggregated across all password categories, the Original and Filtered baseline wordlists produced moderate mean recovery times of 5.33 and 5.32 s, respectively. Among the modified wordlists, Alphabetical sorting demonstrated the best overall performance, with a mean TTC of 4.99 s.



FIGURE 5. Average password-recovery time across the validation datasets.

Structural sorting also produced strong results, with a mean TTC of 5.21 s, slightly outperforming Length sorting at 5.27 s. N-gram and Reverse Alphabetical sorting produced the highest mean cracking times, at 5.69 and 5.65 s, respectively.

In the final validation stage, expanded stress tests were conducted using 1,000 unique Complex passwords and 1,000 unique Transliterated passwords. The Complex-password experiment required approximately 41 h of computation. The resulting recovery-time distributions are presented in Fig. 6.

The expanded empirical results demonstrated high statistical stability for Complex passwords and moderate stability for Transliterated targets, corroborating the initial findings. For the 1,000 Complex passwords, Alphabetical sorting remained the best-performing strategy, achieving a mean TTC of 10.05 s. Its narrow interquartile range (IQR = 1.61) also indicated high performance consistency. The remaining sorting strategies and the Original RockYou baseline formed relatively similar performance clusters, with recovery times approximately twice those of the Alphabetical strategy. The narrow IQR values observed across the sorting methods in this expanded experiment indicate that the initial 100-target sample provided a representative estimate of performance for the Complex category.

Similarly, the evaluation of 1,000 Transliterated passwords reinforced the advantage of N-gram sorting for regional password behavior. N-gram sorting remained the best-performing strategy, achieving a median TTC of 7.16 s. The statistical dispersion observed during the initial phase was also reduced: the N-gram IQR shifted from the previously defined low-stability range to the moderate-stability range. The remaining sorting strategies and the Original RockYou baseline broadly reproduced the performance distributions observed in the initial 100-target evaluation, supporting the specific effectiveness of N-gram sorting for transliterated structural patterns.

V. DISCUSSION

This section provides a comprehensive interpretation of the findings across all testing phases, encompassing the analysis of synthetic simulations, real-world empirical tests,

and cross-dataset validation, culminating in practical recommendations for forensics specialists. The experimental results presented in this study reveal specific dependencies that highlight the practical differences between synthetic simulations and real-world empirical tests. Across all cases, the baseline Filtered RockYou dataset, which discards 33% of the original invalid passwords, consistently increased cracking speed. This improvement occurs because Wi-Fi passwords are strictly constrained to lengths between 8 and 63 characters, removing non-compliant entries inherently streamlines the search space traversal.

However, the magnitude of this difference heavily depends on the hardware utilized. The most significant impact of this filtering is observed during CPU-based processing. In the synthetic tests utilizing a CPU, the Filtered RockYou dictionary saved an average of 40.6% in execution time across large datasets compared to the original list. Conversely, in the real-world experiments leveraging a powerful GPU, the hash rate (kH/s) is orders of magnitude higher, compressing the performance gap between the filtered and original RockYou to a mere 0.6%. This leads to the conclusion that as the computational power of the cracking device increases, the relative benefit of basic length filtering decreases. Considering that the initial filtering process took 1.17 seconds, executing this pre-processing manipulation on high-end devices may introduce unjustified computational overhead if the forensics objective is to crack a single hash in a one-off scenario.

This cost-benefit paradigm also applies to the sorting algorithms. N-gram sorting required 29.05 seconds to organize the RockYou dictionary based on probabilistic character distributions. While a 29.05-second preparation time is mathematically trivial for a one-time, offline laboratory compilation, it introduces a strict operational bottleneck in ad-hoc, on-the-fly scenarios. If a forensics operative attempts to generate an N-gram dictionary directly in the field immediately prior to execution, this CPU-bound preprocessing delay eclipses the marginal seconds saved during the actual GPU hashing phase against a single target. Similarly, Structural sorting required 2.8s, nearly ten times less than N-gram, making it a more computationally optimal solution. Yet, in real-world results, the marginal reduction in Time-to-Crack does not consistently compensate for these preprocessing delays when dealing with isolated targets.

In contrast, the computational time required to execute Length, Alphabetical, and Reverse Alphabetical sorting is remarkably low (under 1 second), allowing them to consistently provide an optimal return on investment from a preprocessing perspective. Furthermore, while contemporary penetration testing often utilizes dynamic, rule-based generation (e.g., Hashcat rulesets) to mutate dictionaries on the fly, this approach inherently introduces computational overhead on the GPU and exponentially amplifies the search space. The static heuristic methodologies proposed herein offload pattern evaluation entirely to the CPU during offline

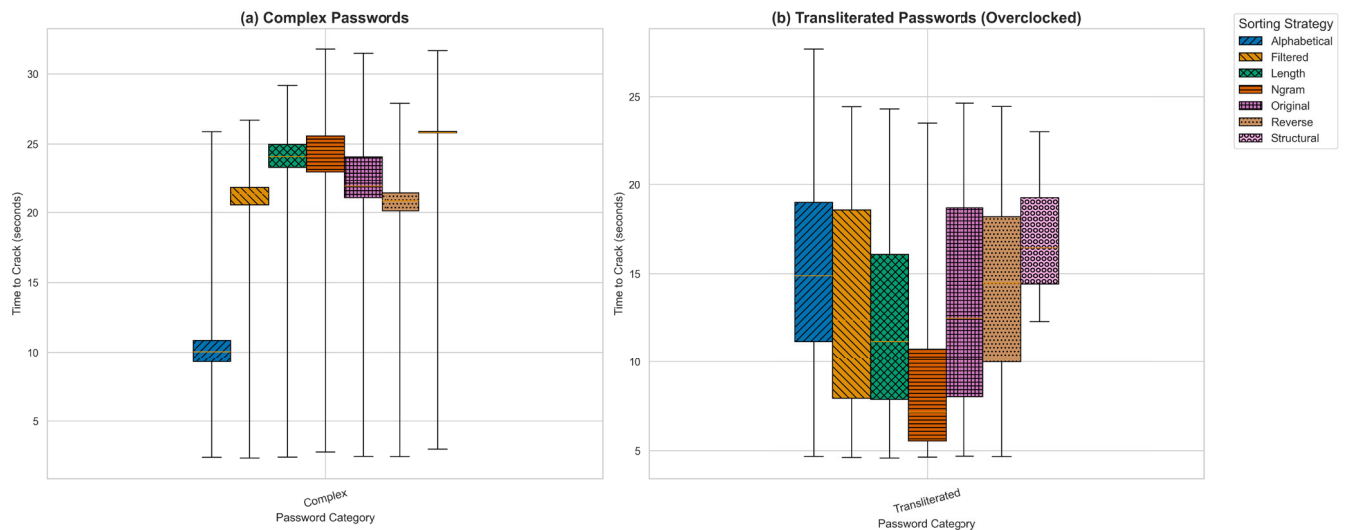


FIGURE 6. RockYou TTC boxplots for the expanded 1,000-target stress tests: (a) the Complex password category evaluated under the standard hardware profile (~ 450 kH/s), and (b) the Transliterated password category evaluated under an overclocked GPU profile (~ 480 kH/s) to assess heuristic scalability under increased hardware throughput.

preprocessing, allowing the GPU to dedicate 100% of its clock cycles to raw hash calculations. Crucially, complex regional behaviors, such as Cyrillic-to-Latin transliterations or spatial Keyboard Layout patterns, do not map naturally to standard English-centric mutation rules. This makes statically pre-sorted, targeted dictionaries a highly efficient, lightweight, and tool-agnostic asset for forensics practitioners. Nevertheless, each sorting method presents unique contextual benefits and disadvantages, which are analyzed in detail below.

A. ANALYSIS OF SYNTHETIC RESULTS AND THEIR CONNECTION TO REAL-WORLD EXPERIMENTS

In the synthetic experiments, five datasets containing a vast array of diverse passwords were analyzed. On average, the baseline RockYou reached a 70% recovery rate within a short traversal period (the first 20% of the search space). This occurs because most of the evaluation datasets are comprised of highly popular passwords, which natively reside at the very beginning of the frequency-sorted RockYou list. Consequently, RockYou performs exceptionally well during the initial traversal phase and remains the optimal solution for simple, mainstream user-created passwords.

However, as target passwords become increasingly unpopular or obscure, RockYou begins to lag behind other sorting methods. At the intersection where 75% of targets were found and 33% of the search space was traversed, the Length sorting strategy surpassed RockYou. Beyond this threshold, Length dictionaries consistently recovered targets earlier than the baseline. This shift is mathematically logical: because RockYou is sorted purely by global popularity, highly unpopular passwords of the minimum required length (8 characters) are relegated to the extreme end of the

frequency distribution. The Length wordlist naturally evaluates these 8-character strings immediately, allowing it to isolate these specific targets much faster post-crossover. The exact same logic applies to the Structural wordlist, which overtook the RockYou performance curve after finding 80% of the targets at the 55% traversal mark. In terms of aggregate performance, Reverse Alphabetical sorting exhibited the worst space-to-recovery ratio.

Ultimately, the aggregate synthetic performance (where modified wordlists attack entire external dictionaries) exhibited low correlation with actual GPU hash-cracking efficiencies, rendering macro-level string matching a suboptimal tool for predicting real-world Time-to-Crack. While synthetic pre-screening demonstrated low correlation with absolute real-world TTC due to the absence of GPU hardware bottlenecks, Phase 1 remains a foundational methodological necessity. It mathematically decouples algorithmic traversal efficiency from complex hardware-dependent bottlenecks, such as GPU memory I/O constraints, candidate batching, and SIMT (Single Instruction, Multiple Threads) block-scheduling quirks. By establishing the theoretical search space boundaries first, this pre-screening phase strongly indicates that the empirical TTC reductions observed in Phase 2 are not mere artifacts of Hashcat's workload tuning, but are fundamentally rooted in the mathematical optimization of the wordlists themselves.

Conversely, targeted pre-screening utilizing the exact 600 structurally categorized passwords provided a significantly more accurate forecast. For the Numeric, Dictionary, Transliterated, Hybrid, and Complex categories, the theoretical search space efficiencies directly predicted the optimal sorting strategies observed in the real-world GPU experiments. For instance, synthetic pre-screening correctly

anticipated that Structural sorting would dominate the Dictionary category, a fact subsequently verified by its superior empirical TTC (10.04s). However, for the Keyboard Layout category, the synthetic recovery curves did not align with the empirical GPU results, indicating that linear pre-screening methods may struggle to predict computational efficiency when dealing with highly unusual spatial complexity.

The N-gram sorting strategy demonstrated consistently strong performance when targeting Cyrillic behaviors. In the synthetic experiment addressing Cyrillic-to-Latin transliteration, the N-gram method yielded the best ratio between recovered targets and search space traversed. This high efficiency is explained by the fact that transliterated passwords frequently rely on natural phonetic syllables (e.g., alternating consonants and vowels like “ma”) combined with highly predictable numeric suffixes (e.g., birth years starting with “19”). While a standard frequency dictionary like RockYou relies on exact full-string matches, the N-gram model mathematically evaluates the underlying probabilities of character sequences, establishing it as the superior theoretical solution for transliterated targets.

Regarding the direct Keyboard Layout Pattern, the Filtered RockYou and Reverse Alphabetical wordlists demonstrated the best theoretical recovery ratios. This phenomenon occurs because many popular Cyrillic words, such as *vfyf* (мама/mother) and *repty* (кузен/cousin), map to English QWERTY keystrokes that begin with letters located toward the end of the standard Latin alphabet (e.g., ‘v’, ‘r’). Consequently, the Reverse Alphabetical strategy mirrors the natural performance curve of these specific strings. The primary reason the original and Filtered RockYou datasets maintained a superior ratio in this specific synthetic test is dataset composition: the targets were generated based on highly popular regional names. Because these foundational nouns are inherently frequent, RockYou effortlessly outperformed other wordlists linearly. However, as demonstrated in Phase 2, while Structural and Length wordlists may traverse more space theoretically, their actual parallelized GPU effectiveness for this specific password type yields far superior real-world TTCs.

B. ANALYSIS OF REAL-WORLD RESULTS

The real-world GPU experiments generally confirmed the conclusions derived from the targeted prescreening phase, with the notable exception of the Keyboard Layout Pattern. Each password category was evaluated under GPU-accelerated conditions. To avoid repeatedly presenting absolute TTC values, the following analysis reports median-based speedup factors relative to the Original RockYou baseline, together with the IQR where relevant. For each password category and sorting strategy, the speedup factor was calculated as the ratio of the Original RockYou median TTC to the corresponding heuristic median TTC.

For the **Numeric** password category, the Structural wordlist demonstrated both the highest statistical stability (IQR = 1.97) and the most significant performance advantage

(a $4.13\times$ speedup). This occurs because pure numeric strings are evaluated first under its rigid heuristic sorting rules. Alphabetical sorting also exhibited strong stability (IQR = 2.24) alongside excellent performance (a $3.85\times$ speedup), as it sequentially evaluates strings starting with numbers due to their early ASCII character placement. The N-gram method performed exceptionally fast (a $2.18\times$ speedup) because common numeric bigrams like ‘10’, ‘02’, and ‘13’ accumulate massive probabilistic frequency scores, though it proved lower stability than the aforementioned strategies (IQR = 6.02). Conversely, despite maintaining reasonable stability (IQR = 2.83), the original RockYou yielded the worst performance for this target type, because long, pure-numeric strings are globally less popular, they are relegated deep into the frequency-sorted dataset.

Results for the **Transliterated** password category corroborated the pre-screening findings, solidifying N-gram as the optimal solution (a $1.55\times$ speedup). Because phonetic bigrams such as “ma” and “pa” hold exceptionally high probability scores in the language model, these passwords were aggressively shifted to the very beginning of the sorted wordlist, allowing the algorithm to recover them almost instantly. However, this method exhibited low stability (IQR = 6.36). In contrast, the Structural wordlist displayed the best stability for this category (IQR = 4.5) but yielded the poorest TTC performance. This is structurally logical: transliterated passwords often contain numbers (e.g., birth years), classifying them as mixed alphanumeric strings, which fall into the Structural algorithm’s third, heavily populated sorting priority. Because a significant portion of targets in this category consisted of eight characters, they were also recovered efficiently by the Length wordlist (a $1.04\times$ speedup) with low stability (IQR = 8.0). The baseline RockYou exhibited poor stability (IQR = 10.41) and average recovery times, indicating that the original frequency-sorted wordlist is a suboptimal solution for regionally transliterated credentials.

For the **Keyboard Layout Pattern**, the empirical GPU scenarios did not perfectly mirror the synthetic pre-screening results. In the real-world execution, the Length strategy achieved the best median TTC, delivering a $1.41\times$ speedup over the Original RockYou baseline. This occurred because the vast majority of generated passwords in this category adhered strictly to short lengths. However, as established in Section V-A, many Cyrillic words typed on an English layout begin with letters located toward the end of the Latin alphabet. This spatial quirk granted the Reverse Alphabetical strategy a slight advantage (a $1.08\times$ speedup). Furthermore, because a subset of these targets included numerical prefixes, the Alphabetical strategy also demonstrated a marginal outperformance (a $1.05\times$ speedup). Crucially, the statistical stability across all sorting strategies (including Original RockYou) was exceptionally wide and volatile for this category, rendering definitive conclusions challenging. It is vital to note that this was the only category inherently constrained by a severe lack of dataset representation (only

105 natural matches found in the unfiltered RockYou). Consequently, passwords generated via the Keyboard Layout Pattern lack a uniquely tailored sorting solution, as their structural composition frequently overlaps with broader categories, morphing into standard Dictionary or Complex strings depending on the presence of capitalization and symbols.

Regarding the **Dictionary** password category, the Structural strategy exhibited the best results (a $1.24\times$ speedup). This aligns closely with its heuristic design, as strings consisting purely of lowercase letters trigger its second highest sorting priority. Interestingly, the Reverse Alphabetical sorting performed slightly better than the Original RockYou (a $1.06\times$ speedup). When evaluating Alphabetical versus Reverse Alphabetical performance in this category, the Time-to-Crack is strictly dictated by the distribution of the initial characters. The empirical results highlight that within this specific 100-target random sample, a larger proportion of the passwords began with letters located toward the latter half of the alphabet. This statistical distribution directly explains the favorable result for the Reverse Alphabetical list and the comparatively poor performance of the standard Alphabetical approach. As for stability, only the Structural strategy demonstrated a tight IQR (2.42), confirming its definitive benefit for this password type. The wider dispersion observed across other wordlists indicates that their efficiency is highly dependent on the specific character composition of the individual password.

The **Hybrid** category revealed a significant and distinct advantage for the Alphabetical strategy (a substantial $2.07\times$ speedup). According to the structural definition established in Section III-B, all passwords in this category mandate an initial uppercase letter. In the standard ASCII table, uppercase letters (decimal codes 65–90) are positioned prior to lowercase letters (codes 97–122). Consequently, strict alphabetical sorting mathematically prioritizes these strings, placing them near the top of the search space. All other sorting strategies also demonstrated advantages over the Original RockYou, ranging from $1.03\times$ to $1.38\times$. This collective outperformance is logical: because randomly extracted, security-compliant Hybrid passwords are inherently less popular among general users, their natural placement within the frequency-sorted Original RockYou is exceptionally deep, causing a severe computational bottleneck. The Alphabetical strategy's dominance is further cemented by its high statistical stability (IQR = 1.13). Conversely, the Length sorting method exhibited low stability (IQR = 9.55) because the target passwords varied significantly in length, directly skewing the TTC. Similarly, the N-gram strategy, which relies on bigram probabilities, yielded unstable results (IQR = 8.23) because security-compliant passwords often intentionally disrupt natural linguistic sequences.

The final category, **Complex** passwords, further demonstrated the clear dominance of the Alphabetical sorting method (a $2.04\times$ speedup) alongside high stability (IQR = 1.52). Similar to the Hybrid category, this is explained by

ASCII character mechanics: Complex passwords require special symbols (e.g., !@#\$%), which occupy the very beginning of the ASCII table (decimal codes 33–47). Consequently, Alphabetical sorting instantly floats these high-entropy strings to the top of the dictionary. In this category, the Reverse Alphabetical list showed a slight outperformance ($1.04\times$), while all other custom sorting strategies fell behind the baseline. Thus, barring the Alphabetical method, the Original RockYou remains a solid secondary choice for high-entropy targets.

Across all categories, Alphabetical sorting achieved the highest mean speedup over Original RockYou ($1.40\times$), followed by Length ($1.16\times$), while N-gram and Structural sorting each achieved approximately $1.14\times$. Reverse Alphabetical and Filtered RockYou provided only marginal aggregate improvements. These averages should not be interpreted as universal rankings because the strongest effects were profile specific: Alphabetical favored Hybrid and Complex targets, Structural favored Numeric and Dictionary targets, N-gram favored Transliterated targets, and Length favored short Keyboard Layout Patterns. Operationally, Alphabetical and Length sorting are suitable for on-demand use because their preparation costs were below 1 s. Structural and especially N-gram sorting should normally be precomputed, since their respective 2.80-s and 29.05-s preparation costs can exceed the TTC reduction for an isolated target.

C. VALIDATION ANALYSIS

Additional cross-validation experiments utilizing five external datasets largely corroborated the core findings while exposing critical hardware-bound scalability thresholds. For highly compact wordlists containing approximately 40,000 lines (e.g., Xato-net and Python-Fa), the application of custom sorting techniques did not provide a practically meaningful or consistently measurable speedup. This plateau is inherently hardware-bound: modern GPUs evaluate such minuscule search spaces virtually instantaneously. Crucially, this plateau becomes even more pronounced when utilizing high-end forensics builds, as the absolute time difference between sorting methods diminishes correspondingly with the hardware's outperformance of the baseline RX 5700 XT. Consequently, the proportional performance gap between sorting heuristics shrinks entirely to the margin of error. Specifically, within the Python-Fa wordlist, the N-gram method yielded the worst average performance, establishing it as a highly suboptimal solution for small datasets when factoring in its substantial preprocessing time constraint.

Performance divergences only began to materialize when evaluating medium-scale datasets, such as Wifi-cracker (488,132 lines). For this volume, the Alphabetical sorting strategy demonstrated a clear advantage (a $1.04\times$ speedup) over the original RockYou baseline, closely mirroring the empirical GPU results from Phase 2. However, this threshold indicates that complex sorting heuristics yield negligible practical returns for wordlists containing fewer than half a million entries.

As the dataset volume expanded to the macro-level (e.g., OneListForAll with 1,525,753 lines), the Alphabetical advantage persisted, albeit reduced to a $1.13\times$ speedup. This reduction is logically attributed to the dataset's native composition, which inherently contained a lower proportion of passwords initiating with uppercase letters or special symbols. During this specific cross-validation phase, a minor computational paradox was observed: the baseline Filtered dictionary occasionally yielded a slightly higher execution time than its larger, unfiltered counterpart. This counterintuitive behavior is not an algorithmic failure but rather a byproduct of statistical variance (I/O jitter) and the inherent block-scheduling nature of GPU SIMT architecture. Such micro-deviations fall comfortably within the expected margin of hardware tolerance and do not undermine macroscopic traversal trends.

The largest validation dataset, Password-Checker (2,240,055 lines), conclusively reaffirmed the Alphabetical strategy as the optimal solution. Interestingly, aside from the Alphabetical method, the Original RockYou outperformed the other custom sorting techniques on this specific dataset. This phenomenon is directly tied to the dataset's origin as a real-world credential leak (alleged Gmail passwords). Such datasets naturally reflect human psychological biases and password fatigue. The Original RockYou is natively sorted by global popularity, aligning well with human cognitive patterns. Applying rigid structural or length-based heuristics to these datasets inadvertently destroys this natural frequency distribution, delaying the recovery of highly common credentials.

Aggregating the empirical TTC across all validation datasets confirms the universal advantage of Alphabetical ($1.07\times$), Structural ($1.02\times$), and Length ($\sim 1.0\times$) sorting strategies compared to the baseline. However, the relatively low overarching percentage boosts firmly dictate a practical operational rule: deploying complex sorting methodologies on small-to-medium wordlists is computationally unjustified. The nominal milliseconds saved during the GPU hashing phase mathematically cannot compensate for the static CPU time required to build and sort the dictionaries offline. In such constrained scenarios, relying on standard frequency-sorted lists remains the most pragmatic approach.

Because Complex strings represent the definitive gold standard recommended by cybersecurity experts, validating this specific category remained a priority despite its inherent stability. The targeted stress test utilizing 1,000 randomly selected Complex passwords strictly verified the Phase 2 outcomes. The Alphabetical sorting strategy maintained high effectiveness (a $2.18\times$ speedup) alongside high statistical stability (IQR = 1.61). The remaining methodologies also mirrored the initial empirical distributions, universally demonstrating high stability (IQR < 2.5), with the exception of the N-gram and Original baselines, which exhibited moderate stability. This expanded validation provides substantial evidence that the established sorting efficiencies

scale consistently, even under mathematically rigorous, high-entropy conditions.

To further validate the regional patterns identified in Phase 2, an expanded dataset of 1,000 unique Transliterated passwords was evaluated under an overclocked hardware profile of approximately 480 kH/s. N-gram sorting retained its proportional advantage and achieved a $1.75\times$ speedup relative to Original RockYou. The expanded experiment produced a median TTC of 7.16 s, compared with 8.44 s in the initial 100-target experiment, corresponding to an observed ratio of approximately 1.18 between the two experiments. However, because the experiments used different target samples, sample sizes, and hardware profiles, the observed difference cannot be attributed exclusively to the additional 30–40 kH/s of GPU throughput. The expanded experiment nevertheless confirmed that N-gram sorting remained the best-performing evaluated strategy for Transliterated targets under the higher-throughput profile.

D. REGION-SENSITIVE PASSWORD RISK MODELING

Cyrillic-to-Latin transliteration and spatial Keyboard Layout Patterns represent fundamentally different cognitive mechanisms in region-sensitive password risk modeling. Rather than functioning solely as additional target categories, these behaviors alter the effective password search space and influence the relative performance of different sorting heuristics.

To quantify this vulnerability, the localized computational risk associated with a target profile θ is defined as

$$R_\theta = P_\theta R_c, \quad R_c = \frac{T_{\text{Baseline}}}{T_{\text{Heuristic}}}, \quad (3)$$

where P_θ denotes the estimated probability of observing profile θ in the target population and R_c denotes the Computational Risk Factor. The latter is calculated as the ratio between the empirical baseline TTC and the TTC achieved with the profile-specific optimized heuristic. Thus, P_θ may be estimated from regional open-source intelligence (OSINT), breach demographics, or other authorized contextual evidence, whereas R_c provides a normalized speedup factor for comparisons performed under the same experimental hardware and workload conditions. Values of $R_c > 1$ indicate that the optimized heuristic reduces TTC relative to the baseline.

Transliteration relies primarily on phonetic familiarity, whereby users map sounds from their native language to Latin characters. Under a conventional brute-force model, a password of length L drawn from an alphabet A has the keyspace $K = |A|^L$, assuming independent character selection. Transliteration violates this independence assumption because the resulting strings are assembled from recurring and predictable phonetic units. An idealized transliteration-aware keyspace can therefore be expressed as

$$K_{\text{trans}} \approx |T|^m 10^d, \quad (4)$$

where $|T|$ is the number of plausible regional phonetic tokens, such as bigrams or syllables including ma and pa ; m is the number of tokens in the password; and d is the number of appended or prepended decimal digits. For example, $d = 4$ can represent a four-digit year. Equation 4 is an idealized model intended to demonstrate the reduction of the effective search space. Real transliterated passwords may contain interspersed digits, symbol substitutions, mixed capitalization, or inconsistent transliteration conventions. Nevertheless, the model isolates the principal entropy reduction produced when independent character combinations are replaced by a constrained set of recurrent phonetic tokens. Because this process preserves probable linguistic sequences, it benefits from N-gram Probability sorting, which prioritizes frequently observed character transitions. In the present experiments, this strategy produced an empirical Computational Risk Factor of $R_c = 1.55$ for Transliterated targets.

By contrast, the Keyboard Layout Pattern occurs when users enter native Cyrillic words while the input layout remains configured for standard English QWERTY. This behavior is driven primarily by motor memory and keyboard-layout mismatch rather than by Latin-language phonetics. Consequently, the resulting strings do not necessarily preserve the phonetic transitions favored by the N-gram Probability strategy. Instead, their ordering can be affected by an asymmetric distribution of Latin characters produced by the Cyrillic-to-QWERTY key mapping. In the evaluated dataset, Length and Reverse Alphabetical sorting produced Computational Risk Factors of $R_c = 1.41$ and $R_c = 1.08$, respectively, for Keyboard Layout Pattern targets. These values are empirical and dataset-dependent rather than universal properties of all Cyrillic keyboard mappings.

The two regional patterns therefore require distinct models. A Transliterated password is primarily phonetic and is best prioritized by N-gram Probability sorting; defensive screening should consequently include common regional transliterations and recurring phonetic sequences. A Keyboard Layout Pattern is driven by motor memory and layout mismatch rather than phonetic similarity; Length or profile-dependent Reverse Alphabetical sorting is therefore more appropriate, while defensive screening should apply explicit Cyrillic-to-QWERTY layout transformations.

E. DERIVATION AND APPLICABILITY BOUNDARIES OF THE DECISION FRAMEWORK

The decision framework introduced in this section is not presented as a pre-existing universal model. It is a practical decision rule derived from the observations obtained across the three-phase experimental pipeline. Its purpose is to assist authorized digital forensics practitioners in selecting a wordlist-sorting heuristic according to the suspected password profile, dictionary scale, hardware throughput, and balance between preprocessing cost and hash-cracking time.

The framework is derived from the following empirical components:

- the WPA2/PMKID Hashcat results obtained across the evaluated password categories;
- the measurable trade-off between dictionary-preparation time, T_{prep} , and GPU execution time, T_{crack} ; and
- the dataset-volume thresholds observed during cross-dataset validation, which indicate when wordlist restructuring becomes operationally worthwhile under the evaluated hardware profile.

1) CONDITIONS OF VALIDITY

The resulting decision rules are contextual and are intended for use only under conditions comparable to those evaluated in this study:

- an authorized WPA2/PMKID offline static-dictionary recovery scenario;
- static, precompiled, or ad hoc generated wordlists;
- the password categories and regional behaviors defined in this study;
- a Hashcat-style parallelized processing workflow; and
- mid-range hardware profiles with performance comparable to the evaluated platform.

2) EXPLICIT EXCLUSIONS

The framework is not intended as a general model for all password-guessing techniques or authentication protocols. It does not model the following dynamic or generative approaches:

- Probabilistic Context-Free Grammar generation;
- algorithmic brute-force or mask attacks;
- dynamic rule-based wordlist mutations, including Hashcat rules applied during execution; or
- neural-network-based or other AI-driven password-generation models.

3) THE HEURISTIC DECISION RULE TREE

To transition from the diverse empirical observations, encompassing both standard structural constraints and the newly established region-sensitive risk models, to an applied methodology, this study proposes a practical decision aid for forensics practitioners that is valid for WPA2 offline dictionary attacks under the evaluated datasets. The framework dynamically matches the evaluated sorting heuristics to specific OSINT-derived target profiles rather than relying on static frequency datasets. The objective of this framework is to minimize the total operational time, defined as Effective Time (T_{eff}), which consists of the offline dictionary preparation cost (T_{prep}) and the GPU-accelerated hashing phase (T_{crack}).

The decision to apply a specific sorting algorithm (S) instead of the baseline frequency dictionary is strictly viable only if the following inequality is satisfied:

$$T_{\text{prep}}(S) + T_{\text{crack}}(N, H, \theta, S) < T_{\text{crack}}(N, H, \theta, \text{Baseline}). \quad (5)$$

where:

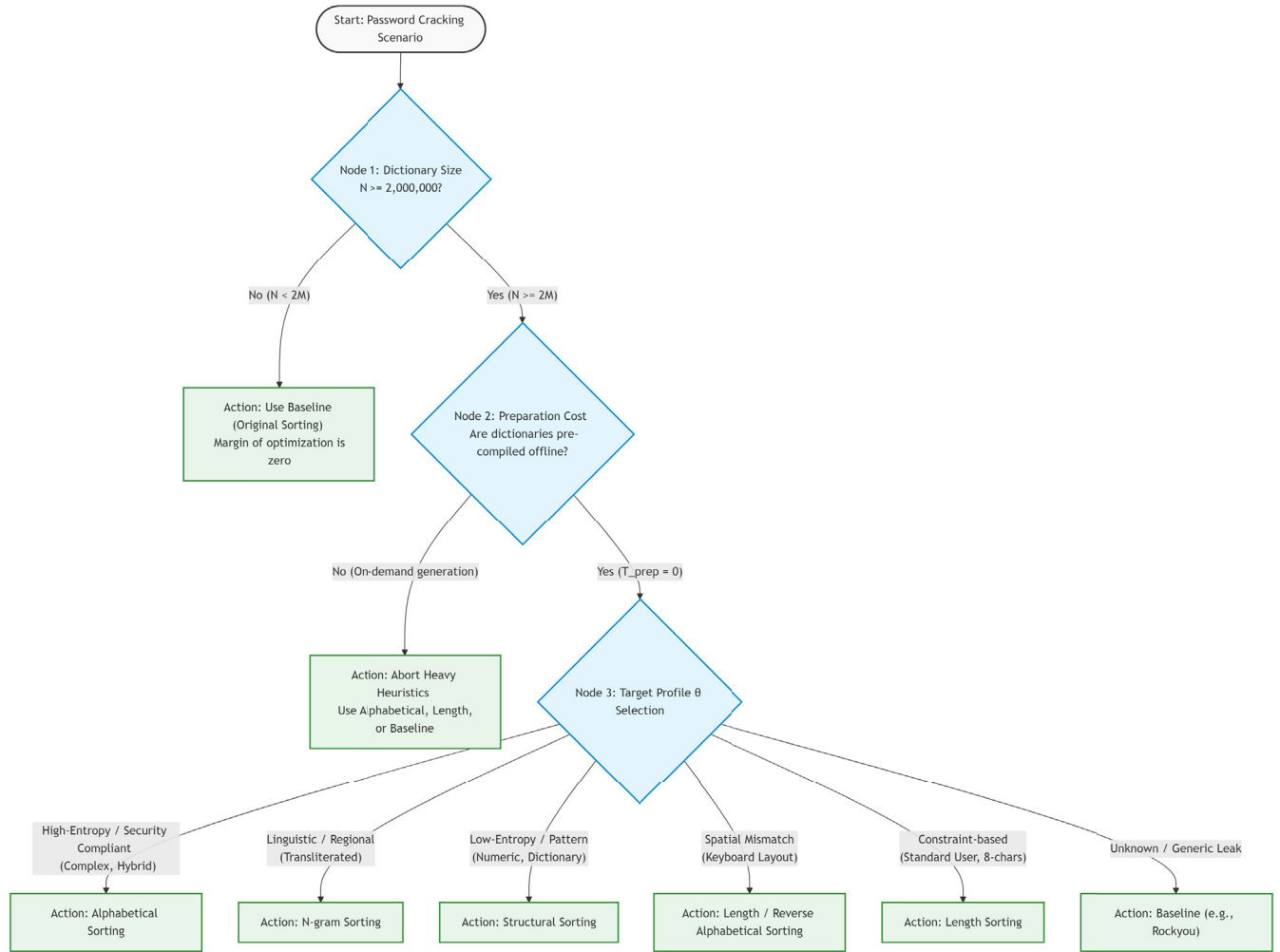


FIGURE 7. Proposed Heuristic Decision Tree for password cracking, optimizing dictionary selection based on the volume threshold N , preparation cost, and OSINT-derived target profile θ .

- N denotes the dictionary size, measured as the number of entries;
- H denotes the available hash rate and represents the hardware capabilities;
- θ denotes the target password profile derived from Open-Source Intelligence (OSINT); and
- S denotes the sorting strategy, including Alphabetical, Structural, N-gram Probability, and Length.

Based on the experimental constraints established in Sections V-B and V-C, the following Decision Rule Tree was formulated to dynamically select the optimal strategy, as illustrated in Fig. 7.

a: NODE 1: DICTIONARY-SIZE THRESHOLD

If $N < 2,000,000$, the measured GPU execution time is sufficiently short that the available reduction in T_{crack} is unlikely to offset preprocessing. Under the evaluated conditions, the appropriate action is therefore to avoid ad hoc sorting and use the baseline ordering. If $N \geq 2,000,000$, the search space is sufficiently large to justify evaluating

structural optimization, and the process continues to Node 2. The two-million-entry value is an empirical threshold derived from the evaluated datasets and hardware, not a protocol-independent constant.

b: NODE 2: HARDWARE AND PREPARATION CONSTRAINTS

If a wordlist must be generated on demand and the selected strategy has a high preparation cost, such as the approximately 29 s observed for N-gram Probability sorting, the preprocessing overhead may exceed the attainable reduction in GPU execution time. In this case, a low-cost strategy such as Alphabetical or Length, with observed preparation times below 1 s, should be preferred unless the expensive wordlist has already been precompiled. If the required variants are available offline, $T_{\text{prep}} = 0$ for the active recovery operation and the process continues to Node 3.

c: NODE 3: TARGET-PROFILE SELECTION

The preferred heuristic is selected using the suspected password behavior:

- **Branch A, High-Entropy or Conventionally Compliant** ($\theta \in \{\text{Complex, Hybrid}\}$): use Alphabetical sorting. It produced the lowest TTC for both categories and maintained high stability for Complex targets.
- **Branch B, Linguistic or Regional Transliteration** ($\theta \in \{\text{Transliterated}\}$): use N-gram Probability sorting, preferably with offline precomputation. This strategy prioritizes recurring phonetic transitions such as *ma* and *pa* and produced the strongest Transliterated performance.
- **Branch C, Low-Entropy or Composition-Based** ($\theta \in \{\text{Numeric, Dictionary Word}\}$): use Structural sorting. Its character-composition priorities isolated digit-only and lowercase alphabetic candidates effectively and produced low TTC values with strong category-specific performance.
- **Branch D, Spatial Keyboard Mismatch** ($\theta \in \{\text{Keyboard Layout Pattern}\}$): use Length sorting as the primary evaluated option, with Reverse Alphabetical as a secondary profile-dependent alternative. These strategies exploit the observed length and character-order properties of native-language words entered through an English QWERTY layout. Because this category exhibited substantial variance and contained only 105 natural baseline matches, the recommendation should be treated as provisional.
- **Branch E, Average User Under WPA2 Length Constraints** ($\theta \in \{\text{Average User}\}$): use Length sorting when there is evidence that minimum-length passwords are likely. This option should be applied cautiously because its advantage decreases when target lengths depart from the eight-character WPA2 minimum.
- **Branch F, Unknown or Generic Breach Profile** ($\theta \in \{\text{Unknown}\}$): retain the Original RockYou ordering when no reliable target-profile evidence is available. Its native popularity ordering provides a frequency-oriented baseline and avoids potentially unproductive preprocessing.

The proposed Heuristic Decision Framework converts the experimental observations into a cost-aware selection procedure. By explicitly relating dataset volume N , preprocessing overhead T_{prep} , hardware throughput H , and the OSINT-derived target profile θ , an authorized practitioner can assess whether a specialized wordlist order is operationally justified. The framework remains bounded by the evaluated WPA2/PMKID static-dictionary scenario, datasets, profile definitions, and hardware conditions. The following Recommendations section translates these findings into operational guidance and defensive password-policy controls.

F. RECOMMENDATIONS

The experimental results indicate that no single dictionary-ordering strategy is universally optimal. Its effectiveness depends on the expected target profile, the size and

composition of the candidate dataset, and the available computational hardware. When the target password profile is unknown, forensic practitioners should generally begin with the Original RockYou wordlist as a frequency-ordered baseline. This approach requires no preprocessing and reflects common password-selection biases, making it effective for widely used credentials.

If the baseline attempt is unsuccessful and larger secondary wordlists are available, practitioners may apply targeted sorting heuristics. However, an important operational constraint identified in this study should be considered: under the evaluated hardware conditions, complex sorting methods applied on demand to wordlists containing fewer than approximately 2,000,000 entries generally offer insufficient benefit to offset their preprocessing cost. For larger datasets, the ordering strategy should be selected according to legally obtained case context and open-source intelligence (OSINT) profiling:

- **Alphabetical sorting:** This strategy is recommended when the target is expected to use a security-compliant password containing special symbols, an initial upper-case letter, or a numeric prefix. These character classes occur relatively early in ASCII-based lexical ordering, moving many Complex, Hybrid, and numeric-prefixed candidates toward the beginning of the search space. In the present experiments, this strategy also exhibited high statistical stability for the relevant categories.
- **Structural sorting:** This strategy is recommended for low- to medium-complexity targets expected to consist of a Dictionary word or a purely Numeric string. It provided short TTC values and high stability for these specific profiles.
- **N-gram sorting:** This strategy is recommended when the available case context suggests transliterated credentials associated with Cyrillic-dependent language use. Its probabilistic scoring of phonetic character sequences made it the most effective approach for the Transliterated category.
- **Length sorting:** This strategy is most appropriate when contextual evidence suggests that the target may use the minimum eight-character WPA2 passphrase length. Its performance is highly sensitive to the actual target length, and its comparatively high variability should therefore be considered.

From a preprocessing perspective, N-gram and Structural dictionaries should preferably be generated and stored offline before field deployment. Their preparation times, 29.05 and 2.80 s, respectively, can outweigh the GPU hashing-time reduction if they are generated on demand for a single target. Alphabetical and Length sorting, by contrast, can be performed in less than 1 s under the evaluated conditions. For highly personalized passwords incorporating information such as names or number sequences, a legally authorized, case-specific candidate dictionary may remain more effective than a general-purpose wordlist.

Although field investigations typically target isolated hashes rather than the prolonged batch workloads used to characterize sustained performance, forensic laptops operate within constrained thermal envelopes. A long-running attack against a high-entropy target may activate dynamic voltage and frequency scaling (DVFS) and reduce mobile-GPU throughput. Ordering candidate dictionaries so that profile-relevant entries are evaluated early may therefore improve the probability of recovery during the system's initial high-performance thermal window. This consideration further supports the use of low-overhead, profile-matched strategies such as Alphabetical or Length sorting when supported by the available case context.

From a defensive perspective, users should select credentials that are both structurally strong and absent from known breach-derived dictionaries. Users in Cyrillic-dependent regions should not treat keyboard-layout conversion as a substitute for password strength because such patterns can be modeled and prioritized by targeted recovery methods. Long, unique, and unpredictable passphrases of at least 16 characters provide substantially greater resistance than short passwords based on words, names, transliterations, keyboard patterns, or predictable numeric suffixes.

The strongest protocol-level mitigation considered in this study is the exclusive use of WPA3-Personal with Simultaneous Authentication of Equals (SAE), where supported by all required devices. WPA2/WPA3 Transition Mode retains WPA2 compatibility and may therefore preserve attack paths associated with the WPA2 portion of the deployment. As discussed in prior work [13], the security benefits of WPA3 should not be assumed to apply fully while legacy WPA2 authentication remains enabled. Consequently, organizations should disable transition mode after compatibility requirements have been resolved and should combine WPA3-only deployment with strong, unique credentials, current firmware, and appropriate network monitoring.

G. LIMITATIONS

The reported results are bounded by four main limitations. First, small datasets exhibited a 4.25–4.32-s execution plateau dominated by tool initialization, dictionary loading, and host-to-device transfer; dictionary ordering therefore became measurable only for sufficiently large candidate sets. Second, absolute TTC values depend on the evaluated Hashcat configuration and the approximately 450-kH/s RX 5700 XT baseline. Thermally constrained mobile GPUs may show greater variance, whereas multi-GPU systems may compress differences among strategies. Third, the primary experiment used 100 targets per category. Although the Complex and Transliterated conclusions were subsequently evaluated with 1,000-target samples, the Keyboard Layout category was limited to 105 naturally occurring RockYou matches; conclusions for this profile are therefore provisional and dataset dependent. Finally, the proposed framework applies to static WPA2/PMKID dictionary ordering and does not cover brute force, masks, PCFG generation, dynamic Hashcat

rules, neural password generation, or other protocols. Future work should evaluate these approaches using larger localized datasets and include preprocessing, storage, transfer, thermal, and recovery-probability costs in a unified effective-time model.

H. FUTURE WORK

This study focuses on static dictionary attacks and offline wordlist optimization for authorized WPA2 password recovery. It does not evaluate dynamic candidate-generation scenarios in which forensic practitioners apply rule-based transformations, such as Hashcat rulesets, or targeted mask-based searches during execution. Future work will compare optimized static dictionaries with rule-based and mask-based candidate generation under equivalent hardware, target, and time constraints.

A particular focus of future research will be the computational trade-off between storing and traversing large pre-computed wordlists and generating smaller, profile-specific candidate streams dynamically on the GPU. This comparison should incorporate preprocessing time, storage requirements, host-to-device transfer costs, GPU utilization, thermal behavior, recovery probability, and total TTC. Additional work should also expand the evaluation of Cyrillic transliteration and keyboard-layout patterns using larger localized datasets and language-specific N-gram models. Future work should further evaluate whether bigram probability tables derived from the length-filtered RockYou corpus transfer effectively to non-English or linguistically distinct breach datasets, and whether corpus-specific recalculation improves their performance.

VI. CONCLUSION

Reducing the computational cost of authorized WPA2 offline dictionary recovery remains relevant to digital forensics and wireless-network security assessment. Whereas much previous work has emphasized hardware accelerators and cracking software, this study examined candidate-data optimization. Through systematic filtering and ordering experiments, the research shifted attention from raw throughput alone to algorithmic reduction of TTC, particularly in resource-constrained forensic environments.

The study also contributes to the analysis of localized password behavior by evaluating Cyrillic-dependent transliteration and keyboard-layout patterns. The results demonstrate that regional linguistic and spatial characteristics can alter candidate ordering and recovery performance. These behaviors should therefore be considered both when constructing authorized forensic workflows and when designing localized password-strength controls.

The principal contributions correspond to the three research questions introduced in Section I:

- 1) **C1: Filtering and sorting efficacy and their computational trade-offs.** Applying WPA2-specific length filtering produced only a $1.01\times$ speedup during

the real-world GPU experiments because the high candidate-processing throughput reduced the practical effect of invalid entries. More substantial gains were obtained through structural reordering of the candidate set. The results also established that preprocessing time (T_{prep}) must be evaluated together with cracking time (T_{crack}). Alphabetical and Length sorting provided the most favorable end-to-end trade-off across broad scenarios because they required less than 1 s to generate.

2) **C2: Result stability and the Heuristic Decision Framework.** Cross-dataset validation identified dependencies on dataset scale and target complexity. Under the evaluated configuration, applying complex sorting methods on demand to datasets containing fewer than approximately 2,000,000 entries was generally not justified by the measured runtime gains. The resulting profile-aware Heuristic Decision Framework identifies Alphabetical sorting as the strongest evaluated option for High-Entropy and Security-Compliant credentials, including Complex and Hybrid passwords. In the 1,000-target Complex-password stress test, it produced a speedup greater than $2.0\times$ with an IQR of 1.61. Structural sorting was the strongest evaluated strategy for low-entropy Numeric and Dictionary patterns.

3) **C3: Exploitation of regional Cyrillic specificity.** The experiments confirmed that regional password-generation habits can create measurable recovery advantages for targeted ordering strategies. For Cyrillic-derived transliterated passwords, N-gram sorting achieved a $1.55\times$ speedup in the initial empirical experiment because it prioritized recurring phonetic sequences such as *ma* and *pa*. The expanded 1,000-target experiment retained the proportional advantage under higher GPU throughput, indicating that the observed N-gram benefit was not limited to the original sample or clock profile. Keyboard Layout Pattern passwords also exhibited distinctive spatial behavior. Reverse Alphabetical sorting benefited from some mappings that began with letters in the latter part of the Latin alphabet, but the high variability and limited natural sample size mean that this result should be treated as preliminary.

These findings also have defensive implications. Password-policy tools and localized password checkers should account for region-specific transliteration and mismatched keyboard-layout patterns rather than relying only on English-language dictionaries. Detection mechanisms may incorporate phonetic sequence analysis and spatial keyboard transformations to identify predictable regional constructions. Because standard complexity patterns can themselves be prioritized through structural ordering, users should be encouraged to adopt long, unique, and unpredictable

passphrases. Where device compatibility permits, these measures should be combined with WPA3-only configurations.

Overall, the empirical evidence provides lightweight and reproducible guidance for authorized digital forensics operations under hardware constraints. The proposed decision logic transforms dictionary selection from an intuitive choice into a profile-aware process based on dataset size, preprocessing cost, statistical stability, and expected password structure.

ACKNOWLEDGMENT

During manuscript preparation, the author used Microsoft Copilot for language editing, grammar correction, academic rewriting, LaTeX formatting assistance, and assistance with drafting Python automation code. Microsoft Copilot was not used to generate experimental observations, execute the reported experiments, determine statistical results, or create bibliographic references. All AI-assisted text, code, and LaTeX content were critically reviewed, revised where necessary, tested where applicable, and verified by the author, who takes full responsibility for the manuscript.

DATA AVAILABILITY

The source code, custom Python scripts, generated target hashes in `.hc22000` format, heuristically sorted wordlists, and aggregated empirical metrics produced during this study are available in the Zenodo replication package [43]. The original third-party credential datasets used during empirical validation remain available through their respective open-source repositories, as specified in the References section.

REFERENCES

- [1] D. K. Vinutha, S. Lebaka, M. Rajakumar, Z. A. Alsalami, S. Z. Rustambekovna, and D. Dhanasekaran, "Secure wireless connectivity frameworks for financial internet services," *J. Internet Services Inf. Secur.*, vol. 15, no. 3, pp. 298–317, Aug. 2025, doi: [10.58346/jisis.2025.i3.021](https://doi.org/10.58346/jisis.2025.i3.021).
- [2] N. Miguélez-Gómez and E. A. Rojas-Nastrucci, "RF fingerprinting: hardware-trustworthiness enhancement in the hardware trojan era: RF fingerprinting-based countermeasures," *IEEE Microw. Mag.*, vol. 24, no. 11, pp. 35–52, Nov. 2023, doi: [10.1109/MMM.2023.3303591](https://doi.org/10.1109/MMM.2023.3303591).
- [3] K. Pahlavan and P. Krishnamurthy, "Evolution and impact of Wi-Fi technology and applications: A historical perspective," *Int. J. Wireless Inf. Netw.*, vol. 28, no. 1, pp. 3–19, Nov. 2020, doi: [10.1007/s10776-020-00501-8](https://doi.org/10.1007/s10776-020-00501-8).
- [4] C. P. Kohlios and T. Hayajneh, "A comprehensive attack flow model and security analysis for Wi-Fi and WPA3," *Electronics*, vol. 7, no. 11, p. 284, Oct. 2018, doi: [10.3390/electronics7110284](https://doi.org/10.3390/electronics7110284).
- [5] R. Nayanajith, "WPA3 deployment challenges," presented at the Wireless LAN Professionals Conf. (WLPC), Feb. 2026. [Online]. Available: https://mrnciew.com/wp-content/uploads/2026/02/wpa3-deployment-challenges_wlpc2026.pdf
- [6] A. Carballal, J. P. Galego-Carro, N. Rodriguez-Fernandez, and C. Fernandez-Lozano, "Wi-Fi handshake: Analysis of password patterns in Wi-Fi networks," *PeerJ Comput. Sci.*, vol. 8, p. e1185, Dec. 2022, doi: [10.7717/peerj-cs.1185](https://doi.org/10.7717/peerj-cs.1185).
- [7] Verizon. (2025). *2025 Data Breach Investigations Report*. Accessed: Aug. 25, 2026. [Online]. Available: <https://www.verizon.com/business/resources/T16f/reports/2025-dbir-data-breach-investigations-report.pdf>
- [8] O. Chalyi, K. Driaunys, Š. Grigaliūnas, and R. Brūzgienė, "Standard-oriented architecture for AI-powered information security risk management," *Electronics*, vol. 15, no. 6, p. 1282, Mar. 2026, doi: [10.3390/electronics15061282](https://doi.org/10.3390/electronics15061282).
- [9] *Information Security, Cybersecurity and Privacy Protection-information Security Management Systems-requirements*, Standard ISO/IEC 27001, Oct. 2022.

- [10] F. Palmese, A. Enrico Cesare Redondi, and M. Cesana, "Resource optimization for evidence collection and preservation in IoT forensics-ready access points," *IEEE Trans. Netw. Service Manage.*, vol. 22, no. 5, pp. 4495–4508, Oct. 2025, doi: [10.1109/TNSM.2025.3586336](https://doi.org/10.1109/TNSM.2025.3586336).
- [11] L. Klasién, N. Fock, and R. Forchheimer, "The invisible evidence: Digital forensics as key to solving crimes in the digital age," *Forensic Sci. Int.*, vol. 362, Sep. 2024, Art. no. 112133, doi: [10.1016/j.forsciint.2024.112133](https://doi.org/10.1016/j.forsciint.2024.112133).
- [12] C. Chen and B. Dong, "Digital forensics analysis based on cybercrime and the study of the rule of law in space governance," *Open Comput. Sci.*, vol. 13, no. 1, May 2023, Art. no. 20220266, doi: [10.1515/comp-2022-0266](https://doi.org/10.1515/comp-2022-0266).
- [13] O. Chalyi, "Assessing Wi-Fi security protocols: A study of dictionary attack performance," *Baltic J. Modern Comput.*, vol. 13, no. 3, pp. 592–623, 2025, doi: [10.22364/bjmc.2025.13.3.03](https://doi.org/10.22364/bjmc.2025.13.3.03).
- [14] V. Taneski, M. Heričko, and B. Brumen, "Systematic overview of password security problems," *Acta Polytechnica Hungarica*, vol. 16, no. 3, pp. 143–165, May 2019, doi: [10.12700/APH.16.3.2019.3.8](https://doi.org/10.12700/APH.16.3.2019.3.8).
- [15] D. Ranghetti Pilar, A. Jaeger, C. F. A. Gomes, and L. M. Stein, "Passwords usage and human memory limitations: A survey across age and educational background," *PLoS ONE*, vol. 7, no. 12, Dec. 2012, Art. no. e51067, doi: [10.1371/journal.pone.0051067](https://doi.org/10.1371/journal.pone.0051067).
- [16] M. Awad, Z. Al-Qudah, S. Idwan, and A. Halim Jallad, "Evaluating password behavior at a small university," *J. Comput. Sci.*, vol. 15, no. 1, pp. 1–9, Jan. 2019, doi: [10.3844/jcssp.2019.1.9](https://doi.org/10.3844/jcssp.2019.1.9).
- [17] S. Houshmand, S. Aggarwal, and R. Flood, "Next gen PCFG password cracking," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1776–1791, Aug. 2015, doi: [10.1109/TIFS.2015.2428671](https://doi.org/10.1109/TIFS.2015.2428671).
- [18] K. Moissinac, D. Ramos, G. Rendon, and A. Elleithy, "Wireless encryption and WPA2 weaknesses," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 1007–1015, doi: [10.1109/CCWC51732.2021.9376023](https://doi.org/10.1109/CCWC51732.2021.9376023).
- [19] F. Prodani and A. Rista, "Analyzing password security in Wi-Fi networks using the airgeddon script," in *Proc. Int. Workshop Quantum Biomed. Appl., Technol., Sensors (Q-BATS)*, 2024, pp. 93–98, doi: [10.1109/Q-BATS63267.2024.10873935](https://doi.org/10.1109/Q-BATS63267.2024.10873935).
- [20] I. Alkhwaja, M. Albugami, A. Alkhwaja, M. Alghamdi, H. Abahussain, F. Alfawaz, A. Almurayh, and N. Min-Allah, "Password cracking with brute force algorithm and dictionary attack using parallel programming," *Appl. Sci.*, vol. 13, no. 10, p. 5979, May 2023, doi: [10.3390/app13105979](https://doi.org/10.3390/app13105979).
- [21] W. Han, Z. Li, L. Yuan, and W. Xu, "Regional patterns and vulnerability analysis of Chinese web passwords," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 258–272, Feb. 2016, doi: [10.1109/TIFS.2015.2490620](https://doi.org/10.1109/TIFS.2015.2490620).
- [22] D. He, B. Zhou, H. Yu, Y. Cheng, S. Chan, M. Zhang, and N. Guizani, "Group-based password characteristics analysis," *IEEE Netw.*, vol. 35, no. 1, pp. 311–317, Jan. 2021, doi: [10.1109/MNET.011.2000354](https://doi.org/10.1109/MNET.011.2000354).
- [23] D. He, H. Yu, B. Zhou, S. Zhu, M. Zhang, S. Chan, and M. Guizani, "How does social behavior affect your password?" *IEEE Netw.*, vol. 35, no. 5, pp. 284–289, Sep. 2021, doi: [10.1109/MNET.101.2000762](https://doi.org/10.1109/MNET.101.2000762).
- [24] D. Miessler. *SecLists: Passwords/Leaked-Databases/Rockyou.Txt.Tar.Gz*. Accessed: Aug. 25, 2026. [Online]. Available: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/rockyou.txt.tar.gz>
- [25] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements*, Standard IEEE Std 802.11i, Jul. 2004.
- [26] *WiFi_Cracker: Password-list.Txt*. Accessed: Aug. 25, 2026. [Online]. Available: https://github.com/sina-programer/WiFi_Cracker/blob/main/password-list.txt
- [27] D. Miessler. *SecLists: Passwords/Common-Credentials/Xato-Net-10-Million-Passwords-100000.Txt*. Accessed: Aug. 25, 2026. [Online]. Available: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/xato-net-10-million-passwords-100000.txt>
- [28] *Password-List: Password-List.Txt*. Accessed: Aug. 25, 2026. [Online]. Available: <https://github.com/Python-Fa/Password-List/blob/master/Password-List.txt>
- [29] *Password-Checker: Lists/alleged-gmail-passwords.txt*. Accessed: Aug. 25, 2026. [Online]. Available: <https://github.com/breezy-codes/Password-Checker/blob/main/Lists/alleged-gmail-passwords.txt>
- [30] Six2dez. *OneListForAll: Onelistforallshort.Txt*. Accessed: Aug. 25, 2026. [Online]. Available: <https://github.com/six2dez/OneListForAll/blob/main/onelistforallshort.txt>
- [31] N. S. Awang Abu Bakar, "Using regular expressions for mining data in large software repositories," in *Proc. 5th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M)*, Nov. 2014, pp. 1–6, doi: [10.1109/ICT4M.2014.7020649](https://doi.org/10.1109/ICT4M.2014.7020649).
- [32] Q. Chen, A. Banerjee, Ç. Demiralp, G. Durrett, and I. Dillig, "Data extraction via semantic regular expression synthesis," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA2, pp. 1848–1877, Oct. 2023, doi: [10.1145/3622863](https://doi.org/10.1145/3622863).
- [33] Valve Corporation. *Steam Hardware & Software Survey: Video Card*. Accessed: Aug. 25, 2026. [Online]. Available: <https://store.steampowered.com/hwsurvey/videocard/>
- [34] Y.-J. Yu and C.-J. Wu, "Designing a temperature model to understand the thermal challenges of portable computing platforms," in *Proc. 17th IEEE Intersociety Conf. Thermal Thermomechanical Phenomena Electron. Syst. (ITherm)*, May 2018, pp. 992–999, doi: [10.1109/ITHERM.2018.8419601](https://doi.org/10.1109/ITHERM.2018.8419601).
- [35] N. Chawla, A. Singh, H. Kumar, M. Kar, and S. Mukhopadhyay, "Securing IoT devices using dynamic power management: Machine learning approach," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16379–16394, Nov. 2021, doi: [10.1109/JIOT.2020.3021594](https://doi.org/10.1109/JIOT.2020.3021594).
- [36] H. Huang, M. Lin, Laurence. T. Yang, and Q. Zhang, "Autonomous power management with double-Q reinforcement learning method," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1938–1946, Mar. 2020, doi: [10.1109/TII.2019.2953932](https://doi.org/10.1109/TII.2019.2953932).
- [37] E. Carter. *DIY Cracking Rig 2025: Parts, Cost, Performance*. Accessed: Aug. 25, 2026. [Online]. Available: <https://www.onlinehashcrack.com/guides/password-recovery/diy-cracking-rig-2025-parts-cost-performance.php>
- [38] H. Semenov. *GPU Benchmark Rating Hashcat-ng (WPA-PBKDF2-PMKID+EAPOL) (not FOSS Version, Proprietary Fork)*. Accessed: Aug. 25, 2026. [Online]. Available: <https://gist.github.com/GermanAizek/f968d0bb213b2b7c8d9b35bbc4d031e4>
- [39] H. Choi and S. C. Seo, "Optimization of PBKDF2 using HMAC-SHA2 and HMAC-LSH families in CPU environment," *IEEE Access*, vol. 9, pp. 40165–40177, 2021, doi: [10.1109/ACCESS.2021.3065082](https://doi.org/10.1109/ACCESS.2021.3065082).
- [40] A. Abdullah S. AlQahtani, "Key derivation: A dynamic PBKDF2 model for modern cryptographic systems," *Cryptography*, vol. 9, no. 2, p. 39, Jun. 2025, doi: [10.3390/cryptography9020039](https://doi.org/10.3390/cryptography9020039).
- [41] P. A. Grassi, M. E. Garcia, and J. L. Fenton, *Digital Identity Guidelines: Authentication and Lifecycle Management*, Standard 800-63B, 2017.
- [42] C. Shen, T. Yu, H. Xu, G. Yang, and X. Guan, "User practice in password security: An empirical study of real-life passwords in the wild," *Comput. Secur.*, vol. 61, pp. 130–141, Aug. 2016, doi: [10.1016/j.cose.2016.05.007](https://doi.org/10.1016/j.cose.2016.05.007).
- [43] O. Chalyi, "Replication package for: 'Beyond hardware acceleration: A heuristic decision framework for optimizing WPA2 dictionary attacks,'" Zenodo, Geneva, Switzerland, Mar. 2026, doi: [10.5281/zenodo.19462501](https://doi.org/10.5281/zenodo.19462501).



OLEKSI CHALYI received the B.S. and M.S. degrees in cybersecurity from the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine, in 2022 and 2024, respectively, and the second M.S. degree in computing from Vytautas Magnus University, Kaunas, Lithuania, in 2024, where he is currently pursuing the Ph.D. degree in information engineering. Since October 2025, he has been a Junior Research Fellow with Vilnius University. He has authored or co-authored more than ten scholarly works, including journal articles, conference papers, and a master's thesis. His research interests include information security, password security, artificial intelligence, browser and wireless security, and cybersecurity risk management.

• • •