GuessFuse: Hybrid Password Guessing With Multi-View

Zhijie Xie, Fan Shi, Min Zhang, Huimin Ma, Huaixi Wang, Zhenhan Li, and Yunyi Zhang

***Abstract*— Password guessing is a primary method for password strength evaluation. Despite various password guessing models have been proposed, there is still a significant gap between their guessing effectiveness and the actual cracking capabilities of attackers. Integrating multiple models for password guessing, also known as hybrid password guessing, could better capture the cracking capabilities of real attackers. However, the reason why hybrid password guessing can enhance cracking capabilities, and how to effectively integrate multiple heterogeneous password guessing models, are still not well understood. To address these issues, this paper draws inspiration from the concept of multi-view learning. We regard the guess lists generated by various password guessing models as multiple views of the data. Through a comprehensive analysis of these guess lists, we have identified the key reason why hybrid password guessing can enhance the cracking capabilities: *integrating more diverse views allows for the coverage of a wider range of heterogeneous password characteristics, and provides more detailed information on effective password distributions.* Based on the findings, we propose a new hybrid password guessing framework, named *GuessFuse*. *GuessFuse* employs the multi-view subset extraction module and the segment splitting selection module to accurately extract and reorganize the effective password from diverse guess lists. Experimental results on six large-scale datasets demonstrate the effectiveness of *GuessFuse*. By combining two (resp. five) guess lists, *GuessFuse* outperforms its foremost counterparts by an average of 11.00% $\sim$ 59.62% (resp. 4.70% $\sim$ 17.66%) within 10^7 guesses. *GuessFuse* can effectively improve the cracking success rate under a limited number of guesses, approaching the actual cracking capabilities of attackers.**

***Index Terms*— Password security, hybrid password guessing, password guess list, multi-view learning.**

I. Introduction

DESPITE the emergence of various alternative solutions, such as two-factor authentication [1] and biometric authentication [2], password-based authentication remains the most fundamental authentication method in the foreseeable future due to their ease of deployment and simplicity of use [3], [4], [5]. However, this method is highly susceptible to guessing attacks, such as offline trawling guessing [6], [7] and online targeted guessing [8], [9], especially when confronting human vulnerable behaviors, including choosing popular passwords [10], using personal information to create passwords [8], and reusing existing passwords [11]. To tackle this problem, web services usually adopt a Password Strength Meter (PSM) to prevent users from setting weak passwords [12], [13], [14]. Several studies have shown that well-designed PSMs indeed help users improve their password strength [13], [14]. In practice, accurately assessing the attacker's ability to crack passwords is found to be a crucial basis for the design of PSMs [15], [16], [17], [18], [19]. Therefore, password guessing methods are extensively studied in an attempt to accurately describe the attacker's password cracking capabilities.

Current mainstream password guessing methods concentrate on two aspects: rule-based and data-driven methods. Previous security community focused on rule-based methods [20], [21] using the summarized user behaviors, while recent works refer to data-driven password guessing models, such as PCFG [6], Markov [22], FLA [15], PassGAN [23], and RFGuess [24]. These data-driven models demonstrate their specific advantages in password guessing [16]. For example, PCFG [6] yields better guessing performance on passwords with simple structures (e.g., passsword123), while the Markov model [22] is more effective on passwords with contextual correlation features (e.g., 1qaz2wsx). However, there is still no model to take all in the real-world scenario with the ongoing developments of the password guessing techniques [25], [26], [27], [28]. Ur et al. [16] found that the cracking capabilities of real-world experts far exceed those of single automated password guessing models. Besides, it can be approximated by using multiple password guessing models in parallel. Although the parallel use of multiple password guessing models is merely a hypothetical strategy and cannot be practically applied in real scenarios, the advantage of integrating multiple password guessing models (i.e., hybrid password guessing) to enhance password guessing capabilities has begun to attract attention.

Research on hybrid password guessing is still in its primitive stages. Ur et al. [16] designed a Min_{auto} indicator as the upper bound of hybrid password guessing by applying multiple password guessing models in parallel. Several studies [29], [30], [31], [32], [33] have attempted to integrate the advantages of specific models at the structural level (e.g., applying the PCFG model and the Markov model respectively at the password structure level and the string level for password guessing [29]). On the other hand, some studies [32], [34] experimentally combine multiple guess lists generated from different password guessing models for more effective password guessing. However, the above methods either integrate two specific password guessing models or simply combine the

Manuscript received 28 July 2023; revised 19 December 2023; accepted 28 February 2024. Date of publication 11 March 2024; date of current version 6 May 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3100500. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Albert Levi. *(Corresponding author: Min Zhang.)*

The authors are with the College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China (e-mail: xzj9510@nudt.edu.cn; shifan17@nudt.edu.cn; zhangmindy@nudt.edu.cn; mahuimin17@nudt.edu.cn; wanghuaixi@nudt.edu.cn; lizhenhan17@nudt.edu.cn; zhangyyzyy@nudt.edu.cn).

Digital Object Identifier 10.1109/TIFS.2024.3376246

guess lists rather than solving this problem by mechanism. *They have not been able to effectively utilize the advantages of multiple heterogeneous password guessing models.* More importantly, to the best of our knowledge, *why integrating multiple password guessing models can enhance cracking capabilities has not yet been satisfactorily answered.*

To address these issues, for the first time, we apply multi-view learning to study hybrid password guessing. We regard the guess lists generated by various password guessing models as multiple views on the data. Through an in-depth analysis of the combination of diverse guess lists, we reveal the key reason why hybrid password guessing can enhance cracking capabilities. That is, *integrating more diverse views allows for the coverage of a wider range of heterogeneous password characteristics, and provides more detailed information on effective password distributions.* At the same time, we also discover that the passwords in the subsets among guess lists also follow Zipf's law.

Building upon these findings, we propose a new hybrid password guessing framework, named *GuessFuse*. *GuessFuse* enables higher cracking capabilities than the individual password guessing models according to the following workflow. *GuessFuse* first generates multiple password guess lists using individual password guessing models, such as PCFG and Markov models. Then, it extracts the intersecting and complementary subsets between multiple guess lists using a multi-view subset extraction method. Following that, *GuessFuse* splits the subsets into password segments according to the power law. Finally, *GuessFuse* reorganizes the effective password segments and outputs the optimize guess list. Extensive experiments demonstrate that *GuessFuse* outperforms its foremost counterparts and can effectively integrate the advantages of multiple password guessing models. We also applied *GuessFuse* to PSM for a broader comparative analysis. We find that the Min_{auto} indicators underestimate the strength of some passwords, which reduces the usability of password settings. Using *GuessFuse* can solve this problem.

We summarize our contributions as follows:

1) **A new analysis method for multiple password guessing models.** For the first time, we apply the concept of multi-view learning to analyze multiple password guessing models. By regarding the guess lists generated by different password guessing models as diverse views on the data, we delve into and demonstrate the key reason why hybrid password guessing can enhance cracking capabilities. That is, *integrating more diverse views allows for the coverage of a wider range of heterogeneous password characteristics, and provides more detailed information on effective password distributions.* We also find that the passwords in the multi-view subsets also follow Zipf's law.
2) **A new hybrid password guessing framework based on multi-view learning.** Based on the above findings, we propose a new hybrid password guessing framework, named *GuessFuse*. *GuessFuse* utilizes the multi-view subset extraction module and the segment splitting selection module to accurately extract and reorganize the effective password from diverse guess lists. Extensive experiments demonstrate that by integrating two(resp. five) password guessing models, *GuessFuse* outperforms its foremost counterparts by 11.00%(resp. 4.70%) $\sim$ 59.62%(resp. 17.66%) on average within 10^7 guesses.
3) **Two new insights into password guessing and PSM.** We present two substantive insights into password guessing and PSM: 1) *In the intra-site password guessing scenarios, the popular password list in the training set approaches the optimal upper limit of the cracking efficiency.* 2) *The Min-auto indicator significantly underestimates the strength of passwords, which can inconvenience users when setting passwords. Using a more precise hybrid password guessing method as the PSM indicator can mitigate this issue.*

II. Preliminaries

We now briefly review the previous studies on hybrid password guessing and describe the password guessing scenarios, the concept of multi-view learning, the details of the datasets and models that this work is based on.

A. Previous Studies

Over the past several decades, extensive research has been conducted on password guessing, resulting in a gradual evolution from heuristic methods [35], [36], [37] to data-driven machine learning models [6], [15], [22]. Earlier studies primarily concentrated on developing one individual password guessing model [25], [26], [27], [28], [38], continually optimizing it to achieve better outcomes. Nonetheless, *there is still a significant gap between the password cracking effectiveness of the password guessing models and the capabilities of the real attackers* [16]. In 2015, Ur et al. [16] found that the cracking capabilities of real-world experts significantly outperform single automated password guessing models. Besides, the performance could approximate that of experts by employing multiple carefully configured and fine-tuned password guessing models in parallel. Consequently, They introduced a conservative indicator, called Min_{auto}, which represents the minimum guess number required for each password to be cracked by all password guessing models. However, the Min_{auto} indicator is only a theoretical upper bound, and using multiple password guessing models in parallel takes several times more guesses than using a single model, which is inconsistent with the requirements in practical guessing scenarios.

Despite the fact that the strategy of using multiple password guessing models in parallel does not conform to real-world scenarios, the advantages of hybrid password guessing have begun to be recognized. In recent years, research on hybrid password guessing has been proposed, but it is still in its primitive stage. Based on the division of integrated objects, existing hybrid password guessing approaches can be classified into integration at the model architecture level and integration at the guessing list level.

1) Integration at the Model Architecture Level: In 2018, Zhang et al. [29] proposed a hybrid password guessing model called SPSR. SPSR applies the PCFG model [6] to the password structure layer and the Markov-chain model [22] to the password string layer. In the same year, they also proposed the SPRNN model [30], which combined structural division and Bidirectional Long Short-Term Memory recursive neural network (BiLSTM). Contrary to SPRNN, Xia et al. [31] proposed another hybrid password guessing model called PL. PL employed the PCFG model [6] for password string layer deconstruction and the LSTM model [15] for structure layer modeling. These dual approaches effectively utilized the advantages of two password guessing models at different password analysis granularity layers. However, these integrations at the model architecture level require targeted customization, limiting their ability to integrate a wider variety of password guessing models.

2) Integration at the Guessing List Level: In 2021, Wang et al. [32] combined multiple password guess lists equally and deduplicated the output. They found that integrating password guessing models with distinctly different password generation strategies (such as the RNN model [39] and PCFG model [6]) can effectively increase the cracking success rate. In 2022, Parish et al. [34] deduplicated the passwords in the training set and sorted them in descending order of password frequency to generate a guess list. They defined it as the *Identity* Guesser. They discovered that combining multiple guess lists generated by models with the *Identity* Guesser can significantly improve the cracking success rate. However, these two studies only made preliminary attempts at the guess list level. Combining guess lists equally does not accurately extract the effective parts from each guessing model.

In 2022, Han et al. [33] introduced a hybrid guessing framework called *hyPassGu*. *hyPassGu* leverages the strengths of the PCFG and Markov models by restricting each model to generate targeted types of passwords and determining the number of guesses respectively. Despite the claim that *hyPassGu* can be applied to other models beyond PCFG and Markov, it requires prior knowledge about the model's architecture and the types of passwords it targets. Consequently, it cannot be directly applied to other models. Additionally, *hyPassGu* roughly divides the passwords into two categories based on their structural characteristics, resulting in a significant loss of effective passwords. Therefore, the cracking success rate of *hyPassGu* is lower than that of an individual model, and only marginally superior to the models which are restricted the generation of specific types of passwords.

3) Summary: Existing studies on hybrid password guessing have proven through methodological and experimental comparison that integrating various password guessing models can effectively improve the cracking success rate. However, to the best of our knowledge, there are still no satisfactory answers to the following key questions: *(1) What contributes to the improved cracking capabilities of hybrid password guessing? (2) How can we effectively leverage the strengths of multiple different models?* This paper focuses on addressing these questions.

B. Password Guessing Scenarios

Consistent with the majority of related studies [16], [29], [31], [32], [33], [34], this paper primarily focuses on hybrid password guessing using data-driven password guessing models in trawling password guessing scenarios. In these scenarios, attackers first build password guessing models based on leaked datasets, then generate guesses using the models, and finally attempt to crack all the target passwords using the guesses.

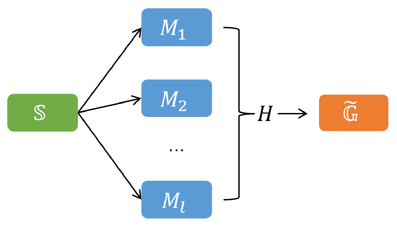


Fig. 1. Explanation of hybrid password guessing. Where, $\mathbb{S}$ is the training set, $\{M_1, M_2, \cdots, M_l\}$ are l different password guessing models, H is the hybrid password guessing approach, and $\widetilde{\mathbb{G}}$ is the guess list generated by H.

While rule-based password guessing tools [20], [21] can also be used for hybrid password guessing, these tools still rely on prior knowledge of the target dataset distribution to enhance the effectiveness of password guessing.

Note that, depending on whether the leaked dataset and the target passwords come from the same data source, trawling scenarios can be further classified into intra-site scenarios and cross-site scenarios. Although cross-site password guessing scenarios are more realistic, attackers generally guess passwords based on the premise that they have partial distribution information of the target data. Hence, to better describe the concept of hybrid password guessing, we formalize the password guessing task in intra-site trawling scenarios.

1) Formalization: Assume an attacker $\mathcal{A}$ wants to make k guesses to crack a target server $\mathbb{T}$. All the passwords in the target server are denoted as a multiset $\mathbb{T} = \{f_1 \cdot pw_1, f_2 \cdot pw_2, f_3 \cdot pw_3, \cdots, f_n \cdot pw_n\}$. Here, pw_i represents the i-th distinct password, and f_i denotes the frequency of pw_i in $\mathbb{T}$, satisfying the relation $f_1 \geq f_2 \geq f_3 \cdots \geq f_n \geq 1$. The k guesses are denoted as an ordered list $\mathbb{G} = (g_1, g_2, g_3, \cdots, g_k)$. Here, g_i represents the i-th guessed password. $\mathcal{A}$ sequentially selects g_i from $\mathbb{G}$ and compares it with any pw in $\mathbb{T}$. If $g_i = pw_j$, it indicates that the i-th guessed password successfully cracks f_j passwords in $\mathbb{T}$.

Fig. 1 shows the explanation of hybrid password guessing. $\mathcal{A}$ assumes that the training set $\mathbb{S}$ shares the same password distribution with the target server $\mathbb{T}$, and uses the hybrid password guessing approach H to generate an optimized guess list $\widetilde{\mathbb{G}}$ based on $\mathbb{S}$ and different password guessing models $\{M_1, M_2, \cdots, M_l\}$ modeled on $\mathbb{S}$. The above process can be formalized as $H(M_1, M_2, \cdots, M_l, \mathbb{S}) \rightarrow \widetilde{\mathbb{G}}$.

C. Multi-View Learning

The concept of multi-view learning aligns well with that of hybrid password guessing. Multi-view learning introduces a function to model a specific view and jointly optimizes all the functions to exploit redundant views of the same input data, thereby enhancing learning performance [40]. At a higher level, multi-view learning constructs multiple views and evaluates their performance, then devises functions to combine these views to improve learning outcomes. Similarly, hybrid password guessing aims to enhance the effectiveness of password guessing by utilizing multiple models that analyze data in different ways. Therefore, multi-view learning can be applied to address the challenge of integrating diverse password guessing models in hybrid password guessing.

Xu et al. [41] proposed that multi-view learning effectively leverages multiple views based on two key principles: *consensus* and *complementary*. The *consensus* principle aims to

TABLE I
BASIC INFORMATION ABOUT OUR SIX PASSWORD DATASETS

Data source	Web service	Language	When leaked	Total size	Size after cleaning	Removed %	Unique passwords
CSDN	Programmer	Chinese	Dec., 2011	6,427,687	6,427,688	0.003%	4,037,073
Dodonew	E-commerce	Chinese	Dec., 2011	16,114,381	16,098,391	0.099%	10,038,410
Tianya	Social forum	Chinese	Dec., 2011	15,367,500	14,941,784	2.770%	7,075,465
Linkedin	Job hunting	English	May, 2012	7,663,884	7,660,316	0.047%	5,406,624
Myspace	Social network	English	July, 2008	5,111,706	5,110,534	0.023%	3,309,962
ClixSense	Paid to click	English	Sep., 2016	3,635,006	3,634,274	0.020%	1,581,894

maximize the agreement among different views. The *complementary* principle states that each view may contain unique knowledge that is not present in others. By effectively utilizing the *consensus* and *complementary* aspects of multiple views, a comprehensive and accurate description of the data can be achieved. In this paper, we explore hybrid password guessing based on these principles.

D. Our Datasets

In this paper, we employ six datasets (see Table I) containing 54 million plain-text passwords. Among these datasets, three are sourced from English and three from Chinese websites, covering six distinct service types. The diversity of data sources helps to reduce bias in our analysis. As all our datasets are ever publicly available on the Internet, the results of this work are reproducible.

1) Data Cleaning: The raw datasets contain anomalies, such as undecrypted hash strings, excessively long passwords that are unlikely to be user-generated, and passwords containing non-printable characters that do not conform to standard password policies. Therefore, we first perform data cleaning on the datasets. We remove passwords containing characters outside the 95 printable ASCII characters and delete passwords with lengths > 30. This data cleaning strategy is also prevalent in the existing password guessing literature [24], [26], and [42].

2) Ethical Considerations: Although the datasets we use are ever publicly available and widely used in password guessing research [8], [27], [32], [33], [34], these datasets contain sensitive personal information. Therefore, we only analyze the distribution characteristics of the password data and report aggregated statistical information. We do not use any data for purposes other than academic research, thus not increasing the risk for affected individuals.

E. Password Guessing Models

We employ four leading password guessing models (such as Markov, PCFG, FLA, and RFGuess) in our study. To mitigate the impact of other factors on the analysis results, we focused on the intra-site password guessing scenario. This scenario is considered ideal because the training and test sets have the same data distribution, ensuring consistency in our analysis. We randomly sampled subsets of size 10^6 from the dataset as the test sets, while the remaining data was used as the training sets. We found that the results were sufficiently stable for analysis based on this test set size. This phenomenon is consistent with the papers [9], [43], [44]. The setup of each model for generating the guess list is as follows:

1) **PCFG.** We utilized version 4.0 of the PCFG model [6], which is available on the GitHub website.[1] The grammar of this model includes six segment categories: alpha A, digits D, other characters O, keyboard D, special string X and years Y.
2) **Markov.** We chose the 4-order Markov model and adopted the Laplace smoothing and end symbol regularization as used in [26] to generate guesses.
3) **FLA.** We utilized the open-source code of FLA, which is available on the GitHub website,[2] and followed the recommended parameters specified in the [15]. We trained a model consisting of three LSTM layers with 128 cells in each layer and two dense layers, a total of 20 epochs.
4) **RFGuess.** Referring to [24], We trained a random forest with 30 decision trees. Its minimum number of leaf nodes is 10, the maximum ratio of features is 80%, and the rest are in default of the scikit-learn framework [45].

III. ANALYSIS

Previous studies on hybrid password guessing have only demonstrated through experimental results that integrating multiple password guessing models can effectively improve cracking capabilities, yet the fundamental reason for this enhancement remained undetermined.

In this section, through multi-view analysis, we sequentially address the following questions: (1) What are the key reasons that hybrid password guessing can improve cracking capabilities? (2) What benefits can be gained from using guess lists for hybrid password guessing?

A. Essence of Hybrid Password Guessing

We demonstrate the differences between password guessing models by comparing the guess lists generated by each model. As is shown in Fig. 2, the guess list generated by the PCFG model includes passwords such as "dearbook123456" and "DEARBOOK", demonstrating a bias towards generating passwords containing common phrases and simple structures. The guess list produced by the Markov model contains passwords like "woshili" and "1989123", reflecting a bias towards guessing based on the character correlation within passwords. The RFGuess model's guess list includes passwords such as "1234567>" and "1234567}", indicating a tendency to combine common phrases with various characters. The FLA model, similar to the Markov model, generates a guess list with passwords like "45665456" and "qwqwqwqww", showcasing its bias for sequential context coherence in passwords. Nevertheless, the guess list generated by the FLA model also differs from that of the Markov model.

Furthermore, there are passwords in the central area, such as "dearbook" and "12345678". It demonstrates that despite

[1]https://github.com/lakiw/pcfg_cracker

[2]https://github.com/cupslab/neural_network_cracking

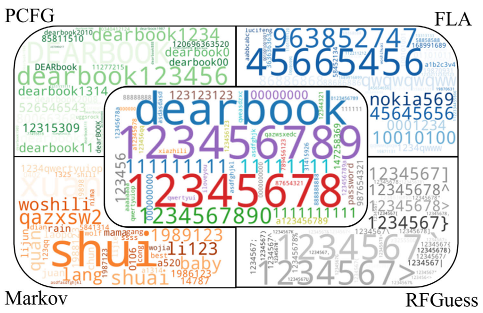


Fig. 2. Word cloud representation of the repeated and unique passwords extracted from the top-10^4 guess lists generated by four models based on the CSDN dataset. The size of the passwords reflects their ranking in the guess lists. The passwords in the central area repeatedly appear in the guess lists generated by various models, while the passwords in the four surrounding areas appear independently in the guess lists generated by a single model.

each model's unique bias, they have reached a consensus on certain passwords. These passwords are likely commonly used and easily predicted by various models. The similarities and differences in the guess lists highlight that, *even when based on the same dataset, existing password guessing models can generate guess lists that encompass distinct password characteristics.*

Essentially, password guessing models are designed based on the behavior of users setting vulnerable passwords. By accurately identifying and matching the characteristics of weak passwords, these models can generate guess lists with a high cracking success rate. However, due to the heterogeneous diversity of user password-setting behaviors [42], even models like PCFG, which already encompass a variety of weak password characteristics (see Section II-E), cannot completely cover all features within a limited number of guesses. *Utilizing a combination of multiple password guessing models with different biases allows for more comprehensive identification and matching of password features, thereby enhancing the performance of password guessing.*

A new challenge arises: how can we effectively utilize a variety of heterogeneous password guessing models for hybrid password guessing? Designing a new password guessing model based on the specific principles of feature extraction from different models can indeed enhance cracking capabilities. However, leveraging multiple heterogeneous password guessing models at the structural level is quite challenging and not applicable to newly proposed password guessing models (e.g., RFGuess), which deviates from the initial idea of using multiple heterogeneous models. Fortunately, password guessing models ultimately generate guess lists. Given the natural suitability of multi-view learning for hybrid password guessing (see Section II-C), we consider the guess lists generated by the password guessing models as multiple views of the data.

We describe the concept of hybrid password guessing with a schematic explanation. As illustrated by Fig. 3, the guess lists $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \cdots, \mathbb{G}_l\}$ generated by different password guessing models can cover different parts of the test set $\mathbb{T}$. Assuming a hybrid password guessing method H can effectively extract the password covering $\mathbb{T}$ from $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \cdots, \mathbb{G}_l\}$ to generate an optimized guess list $\mathbb{G}$. The more guess lists are combined, the greater the coverage of the test set, resulting in better cracking capabilities. Note that the schematic depicted in Fig. 3 may not be general, as there are typically overlaps in

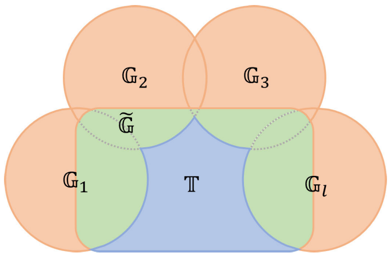


Fig. 3. Schematic explanation describing hybrid password guessing based on multiple guess lists. The rectangle (containing blue and green) represents the test set $\mathbb{T}$. The four circles (containing orange and green) represent the multiple guess lists generated by different models $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \cdots, \mathbb{G}_l\}$. The green area signifies the optimized guess list $\widetilde{\mathbb{G}}$ generated by the optimal combination through the hybrid password guessing method.

passwords among the guess lists. However, the causality that combining more guess lists can cover more of the test set is prevalent. We will prove this through the following analysis.

B. Advantages of Multi-View Integration

For clarity in this section, we have generated the top-10^6 password guess lists based on the CSDN dataset as an example. A multitude of experiments has confirmed that the results we have described are consistently observed across six datasets, thereby affirming their ubiquity.

1) Quantitative Feature Analysis: First, we conduct a quantitative comparison of the statistical features of the combined guess lists to demonstrate the guessing effectiveness brought by multi-view integrations. We combine the guess lists generated by each model in equal size and deduplicate them, then compare the result with single guess lists. For example, when comparing guess lists of size 10^3, we combine the top-10^3 guesses generated by the PCFG and Markov models, removing any duplicate passwords. The output is defined as two views integration. Note that, this combination strategy is identical to the calculation of the Min_{auto} indicator [16] and does not reflect a real-world scenario. Our aim is merely to demonstrate the feasibility of enhancing password cracking capabilities through the quantification of features across multiple views.

Table II provides a statistical analysis of four types of features. Note that, we analyze the number of password structures in the guess list based on the PCFG algorithm. For instance, if the guess list contains passwords such as "dearbook134", "dearbook309", and "123dearbook", it encompasses two types of password structures: "A8D3" and "D3A8". As for the ratio of effective passwords, a password in the guess list is considered effective if it can match a password in the test set.

The ratio of effective passwords to the size of single guess lists (denoted as Effect.) and the cracking success rate on the test set (i.e., the Min_{auto} indicators) illustrate the improvement in guessing effectiveness resulting from multi-view integrations. As shown in Table II, compared to the average cracking success rate in a single view, integrating two views can increase the Min_{auto} indicators by 9.3% on average, three-view integrations by 13.1%, and four-view integrations by 15.5%. Similarly, compared to the average ratio of effective passwords contained in a single view, integrating two views can increase the average ratio of effective passwords by 42.7% on average, three-view integrations by 67.0%, and four-view integrations by 85.58%.

TABLE II

STATISTICAL FEATURES OF HYBRID PASSWORD GUESSING USING MULTIPLE VIEWS[1]

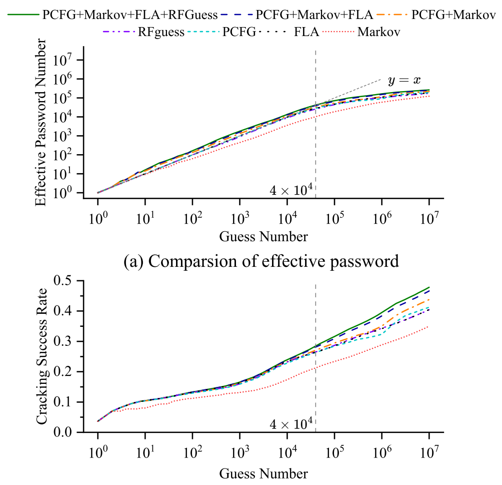
Guess Number	Statistical Features	Single View				Two Views						Three Views				Four Views
		F	P	M	R	P+M	P+F	P+R	M+R	M+F	F+R	P+M+F	M+F+R	P+F+R	P+M+R	P+M+F+R
10^3	Uniq.	100.0%	99.7%	100.0%	100.0%	**186.9%**	131.3%	140.2%	184.4%	185.7%	135.2%	214.6%	219.3%	159.4%	**222.7%**	**241.9%**
	Effect.	**99.9%**	98.9%	44.4%	90.5%	130.6%	130.4%	129.9%	119.3%	130.0%	125.6%	158.2%	154.1%	149.0%	156.9%	**176.0%**
	Min_{auto}	**16.3%**	15.9%	13.2%	16.0%	16.3%	**17.0%**	16.8%	16.1%	16.5%	16.7%	17.1%	16.8%	**17.2%**	16.9%	**17.2%**
	Struct.	76	48	**95**	91	102	84	101	**134**	119	102	122	**140**	107	137	**143**
10^4	Uniq	100.0%	99.8%	100.0%	100.0%	177.3%	121.4%	124.8%	177.0%	**177.3%**	122.5%	197.2%	199.0%	138.9%	**200.1%**	**214.0%**
	Effect	**97.9%**	95.6%	35.7%	89.3%	109.5%	**115.5%**	110.2%	102.3%	111.1%	110.1%	**127.7%**	122.7%	122.8%	122.3%	**134.7%**
	Min_{auto}	**23.4%**	22.8%	17.4%	23.2%	23.4%	**24.0%**	23.9%	23.3%	23.5%	23.8%	24.2%	24.0%	**24.3%**	24.0%	**24.4%**
	Struct	220	137	228	**285**	266	248	306	**375**	316	322	333	**404**	337	386	**413**
10^5	Uniq	100.0%	99.6%	100.0%	100.0%	**181.2%**	154.7%	155.9%	176.5%	178.0%	142.9%	227.8%	216.0%	191.1%	**227.9%**	**260.2%**
	Effect	46.7%	**47.0%**	18.8%	44.4%	53.5%	**63.1%**	60.9%	48.2%	50.7%	56.2%	66.4%	59.0%	**69.4%**	64.2%	**71.8%**
	Min_{auto}	28.5%	28.5%	23.4%	**28.7%**	29.5%	**30.6%**	30.6%	29.0%	28.9%	29.9%	30.9%	30.1%	**31.4%**	30.9%	**31.6%**
	Struct	689	368	849	**1,053**	960	759	1,112	**1,414**	1,139	1,189	1,194	**1,517**	1,232	1,460	**1,555**
10^6	Uniq	100.0%	99.9%	100.0%	100.0%	**191.4%**	186.0%	186.1%	179.1%	179.2%	155.8%	**264.3%**	230.8%	239.4%	264.2%	**314.0%**
	Effect	**10.9%**	9.1%	6.0%	10.4%	11.5%	**14.9%**	14.2%	10.9%	11.4%	12.6%	15.3%	12.9%	**16.1%**	14.6%	**16.3%**
	Min_{auto}	34.2%	32.4%	28.8%	**34.3%**	35.1%	**38.0%**	37.7%	34.8%	34.6%	36.3%	38.4%	36.5%	**39.3%**	38.1%	**39.5%**
	Struct	2,387	660	**3,440**	3,396	3,575	2,506	3,488	**5,091**	4,364	4,020	4,453	**5,543**	4,101	5,166	**5,613**

[1] These statistical features are based on the CSDN dataset. The columns highlighted in blue indicate that these multi-view integrations have a relatively superior guessing effect compared to the same quantitative multi-view integrations. The **Bold numbers** indicate the maximum value of that feature within the same quantitative multi-view combinations. Guess lists generated by single models are considered as single view, such as FLA (F), PCFG (P), Markov (M), and RFGuess (R). The combinations of multiple views are indicated by a "+". For instance, a combination of PCFG and Markov guess lists is denoted as "P+M". Statistical features include the ratio of unique passwords to the size of a single guess list (Uniq.), the ratio of effective passwords to the size of a single guess list (Effect.), the cracking success rate of guesses on the test set (Min_{auto}), and the number of password structures (Struct.).

On the other hand, the ratio of unique passwords to the size of single guess lists (denoted as Uniq.) and the counts of password structures (denoted as Struct.) demonstrate the enhancement of diversity achieved by integrating multiple views. Excluding the minor duplicate passwords that may be generated by the PCFG model, the number of unique passwords in a single view equals the size of single guess lists. Compared to a single view, the integrations of two views can increase the number of unique passwords by an average of 63.8%, three-view integrations by 113.3%, and four-view integrations by 157.5%. Additionally, the integrations of two views also can increase the number of password structures by an average of 45.5%, three-view integrations by 77.1%, and four-view integrations by 102.9%. The above enhancement of the four quantitative features indicates that *integrating more diverse views can increase the variety of guesses, cover more password characteristics, and thereby improve the effectiveness of passphrase guessing.*

2) Effective Password Analysis: Combining multiple guess lists results in a number of unique passwords that far exceed the size of a single guess list. If all are used for the final cracking, it would lead to an excessive consumption of cracking attempts. Fortunately, Fig.4(a) shows that when the guess number grows to approximately 4×10^4, the number of effective passwords from combining multiple guess lists begins to fall below the size of a single guess list. In this case, by selecting a portion of these effective passwords, even if the optimized guess list size is smaller than that of a single guess list, the success rate of guessing can still reach the Min_{auto} indicators of multi-view integrations.

Based on the premise of knowing the test set, we hypothesize an optimal hybrid password guessing method that can effectively integrate multiple guess lists to produce an optimized guess list. The optimized guess list generated by this hybrid password guessing method contains the most efficient passwords from each guess list. A guessed password



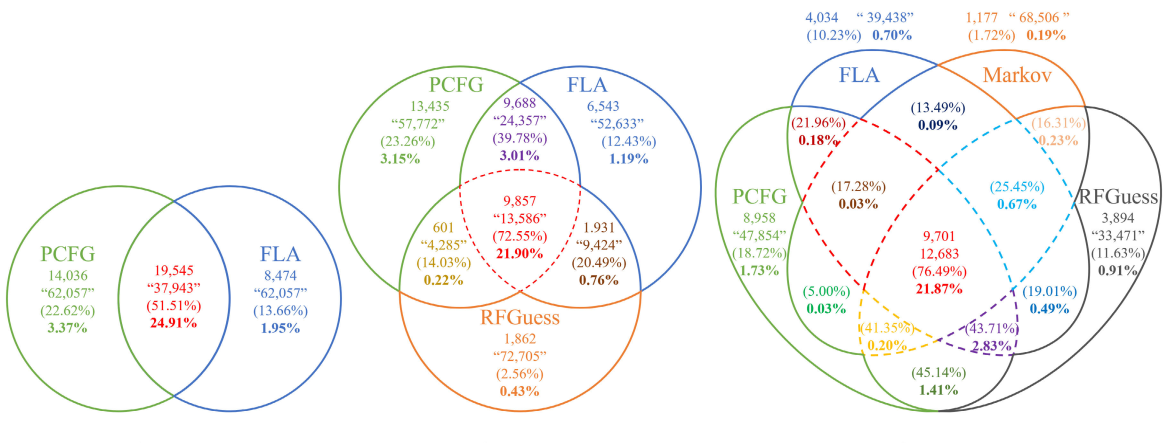
(a) Comparsion of effective password

(b) Comparsion of cracking success rate

Fig. 4. The comparison of the number of effective passwords and the cracking success rate between integrating multiple views and single views.

is considered to have the highest efficiency if it cracks the passwords with the highest frequency in the test set. For example, when integrating the guess lists of size 10^3, we first deduplicate the guessed passwords and compare them with the test set, calculating the number of test passwords each guessed password can crack. Then, we select the guessed passwords in descending order of the number of cracks and add them to the optimal guess list until the size of the optimal guess list also reaches 10^3.

We compare the cracking success rate of the optimal guess lists with that of the guess lists generated by single password guessing models (see Fig.4(b)). As is shown in Fig.4(b), when guesses number $< 4 \times 10^4$, compared to the PCFG model, the cracking success rate of the optimal guess lists integrating two views increased by an average of 1.3%, integrating three views increased by 2.6%, and integrating four views increased by 2.8%. Although the number of effective passwords exceeds the



(a) 2 Views (b) 3 Views (c) 4 Views

Fig. 5. Venn diagram of multi-view subsets based on CSDN dataset. The features are listed from top to bottom as follows: the number of effective passwords in the subset (represented by unsigned numerical values), the number of unique passwords (enclosed in quotation marks ""), the proportion of effective passwords in the subset (enclosed in parentheses ()), and the subset's cracking success rate (indicated by **bold**).

size of a single guess list, selecting a portion of the equivalent size of guessed passwords with the highest efficiency can still enhance the cracking capabilities. The above results indicate that *accurately extracting effective passwords from multiple views is key to conducting effective hybrid password guessing.*

3) Multi-View Subset Analysis: To discover an effective way for extracting effective passwords from multiple views, we conduct an analysis of the subsets among multiple views. Specifically, we perform set operations on guess lists of equal size generated by different models to obtain the intersecting and complementary subsets.

We analyze the variations in multi-view subsets when integrating different numbers of views (see Fig. 5). As illustrated in Fig. 5, integrating two views generates three subsets, three views integrations result in seven subsets and four views integrations lead to 15 subsets. Moreover, as the number of integrated views increases, the distribution of effective passwords within each subset becomes more concentrated. The proportion of effective passwords in the intersecting subset (indicated by the red numerical value in the middle) increases with the addition of integrated views. In the experiments based on the CSDN dataset, the proportion of effective passwords in the intersecting subset is 51.51% when integrating two views. This proportion increases by 21.0% when integrating three views, and by an additional 3.9% when integrating four views. These results demonstrate that *integrating multiple views provides incremental information on the distribution of effective passwords. As more views are integrated, the incremental information increases, and the distribution of effective passwords is more concentrated.*

We continue to analyze the distribution of effective passwords across various subsets. Currently, as we extract the intersecting and complementary subsets among guess lists solely through set operations, the passwords within the subsets do not include a sequential order. Therefore, we evaluate the distribution of effective passwords within the subsets by analyzing the average cracking success rates of subsets among the guess lists of different sizes (see Fig. 7). For ease of representation, we use Fig. 6 to indicate the subsets represented by each average cracking success rate. Specifically, we name each subset using a 4-bit binary code, where each bit represents the

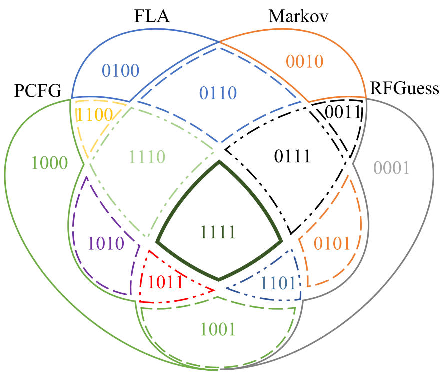


Fig. 6. Venn diagram of four guess lists. The binary encoding represents the names of each subset. The outer circle of each subset corresponds to the characteristics (i.e., color and dashed lines) of the subset curves in Fig. 7.

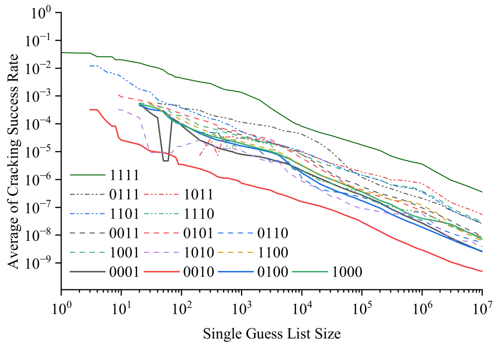


Fig. 7. Distribution of the average cracking success rate of the subsets in the multiple guess lists. The y-axis represents the average cracking success rate of passwords within each subset (calculated as $\frac{\text{cracking success rate of subset}}{\text{size of subset}}$). Note that the y-axis is in logarithmic form.

appearance of a password in the guess lists generated by a specific model, from left to right, respectively: PCFG, FLA, Markov and RFGuess. For instance, "1000" represents that the passwords in this subset appear only in the PCFG guess lists.

As shown in Fig. 7, the overall cracking efficiency of the intersecting subset "1111" is significantly superior to other subsets. However, the latter part of the passwords in the intersecting subset does not outperform the earlier part of the passwords in other subsets. For example, the average cracking success rate of intersecting subset "1111" is 3.57×10^{-6} within

TABLE III

Fitting Results of the Correlation Between the Average Cracking Success Rate of Subsets and the Size of Single Guess Lists Using the PDF-Zipf Model[1]

Subset Label	$\log fr = \log C - s \cdot \log r$				RMSE
	$\log C$		s		
	value	SE	value	SE	
0001	-2.47	0.44	-0.85	0.09	0.23
0010	-3.55	0.34	-0.82	0.07	0.13
0011	-1.77	0.64	-0.88	0.12	0.15
0100	-2.08	0.46	-0.92	0.09	0.08
0101	-1.81	0.40	-0.89	0.09	0.10
0110	-2.29	0.58	-0.86	0.11	0.10
0111	-1.39	0.48	-0.87	0.10	0.18
1000	-2.14	0.43	-0.86	0.09	0.08
1001	-2.19	0.42	-0.74	0.09	0.09
1010	-3.12	0.39	-0.74	0.08	0.24
1011	-2.64	0.55	-0.63	0.10	0.21
1100	-2.09	0.46	-0.85	0.09	0.09
1101	-1.65	0.34	-0.84	0.07	0.08
1110	-2.28	0.60	-0.83	0.11	0.12
1111	-0.85	0.31	-0.78	0.07	0.12

[1] Corresponding to Fig. 7. SE represents the standard error, and RMSE represents the root mean square error.

10^1 guesses, while that of complementary subsets "0010" is 2.61×10^{-5} within 10^1 guesses.

Interestingly, the average cracking success rate of the subsets shows an approximate linear relationship with the size of single guess lists. This becomes even more evident when we consider the 15 curves as a whole. This may adhere to Zipf's law in passwords [46]. We fit the data using the PDF-Zipf model [46]: $\log fr = \log C - s \cdot \log r$, where fr and r correspond to the average cracking success rate and the size of single guess lists, respectively. As shown in Table III, $C \in [-3.55, -0.85]$ and $s \in [-0.92, -0.63]$ are constants. The RMSE of the fitting is within the range of $[0.08, 0.24]$.

This phenomenon aids in leveraging the advantages of multiple password guessing models in hybrid password guessing. Although Zipf's law in password distribution is commonly used to explain user behavior in setting passwords, there are also studies [33], [47] that improved password guessing models by analyzing the presence of Zipf's law within them. In our case, Zipf's law indicates that *the efficiency of guessed passwords declines with their descending order of rank within the subset.* In other words, passwords ranked higher in the subset are considered to have higher cracking efficiency.

4) Summary: Through analyzing different guess lists as multiple views of data, we have revealed the key reasons why hybrid password guessing can enhance cracking capabilities. That is, *integrating more diverse views allows for the coverage of a wider range of heterogeneous password characteristics, and provides more detailed information on effective password distributions.* We also discovered that subsets generated by integrating multiple perspectives have a more concentrated distribution of valid passwords, and the efficiency of passwords within subsets follows Zipf's law. We will utilize this in the following design.

IV. *GuessFuse*

In this section, we use the aforementioned analysis findings to design a hybrid password guessing framework based on multi-view learning, naming it *GuessFuse*. We sequentially answer the following questions: (1) Why do we design *GuessFuse*? (2) How do we implement it?

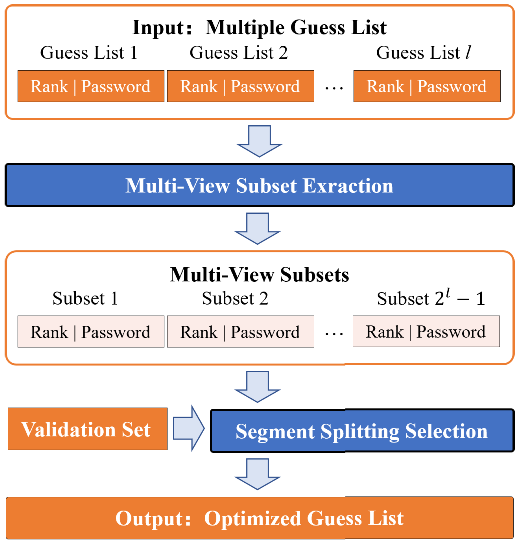


Fig. 8. Overall framework of *GuessFuse*.

A. Overall Framework

In Section III, we demonstrate that the more password guessing models are integrated, the higher cracking capabilities. However, existing approaches cannot effectively integrate multiple heterogeneous password guessing models. Fortunately, since password guessing models ultimately generate guess lists, the integration of guess lists can well solve this problem. Not only that, in Section III, we reveal the biases of various password guessing models through word cloud analysis, and these biases are also well reflected in the guess lists generated by the models. At the same time, the intersecting and complementary subsets between multiple guess lists well reflect the consensus and complementary aspects between different models, which can also help integrate the advantages of multiple password guessing models. Therefore, we propose a new hybrid password guessing framework, *GuessFuse*, to effectively integrate the guess lists generated by multiple heterogeneous password guessing models.

As shown in Fig. 8. *GuessFuse* integrates multiple guess lists generated by different password guessing models. It includes two modules, namely the multi-view subset extraction module and the segment splitting selection module. By inputting l guess lists, the module outputs $2^l - 1$ multi-view subsets. The multi-view subsets contain information about the ranking of the passwords in the original guess lists, so the module is more than just a simple set operation. The splitting selection module receives the multi-view subset collection, extracts the effective parts of the passwords in the multi-view subset based on the validation set, and reorganizes them to generate the final optimized guess list. In the following subsections, we will elaborate on the motivation and concrete implementation of each module.

B. Multi-View Subset Extraction

The multi-view subset extraction module extracts the intersecting and complementary passwords between the guess lists

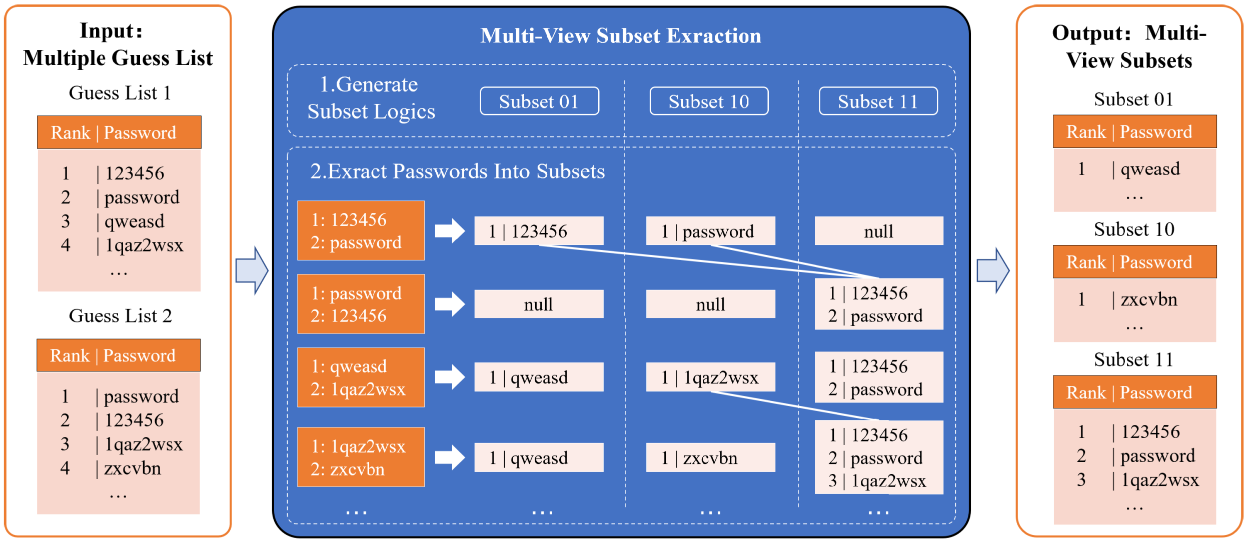


Fig. 9. An example of multi-view subset extraction on two guess lists 1 and 2 with the priority $prior(1) > prior(2)$.

to generate multi-view subsets. Unlike the intersection and complement sets in set operations, the multi-view subsets include the ranking order of the guessed passwords.

1) Motivation: There are duplicate and independently appearing passwords between multiple guess lists (see Fig. 2). If multiple guess lists are simply combined in equal amounts, a considerable part of the passwords will be used for cracking repeatedly, heavily wasting the opportunities of crack. Extracting and integrating the intersecting and complementary subsets between multiple guess lists can effectively solve this problem. Not only that, in Section III, we show that integrating more guess lists can produce more subsets, and effective passwords will be more concentrated in specific subsets (see Fig. 5). Retaining the order of the guessed passwords can assist in extracting effective passwords from the subset, which we will apply in the segment splitting selection module.

2) Implementation: To better describe the process of the multi-view subset extraction module, we use an example to describe the flow of the module's operation (see Fig. 9). During the process, the module first generates $2^l - 1$ subsets according to the number of input guess lists l and assigns logical labels to them one by one. In Fig. 9, for the input of two guess lists, the module first generates $2^2 - 1 = 3$ subsets. Then, a two-bit binary code is used to represent the logical relationship between the subset and the guess list, in the same way as shown in Fig. 6. The module maintains these subsets throughout the processing. Next, the module processes the guessed passwords round by round according to the ranking, comparing whether the input password already exists in each subset. If the password does not appear in the subset, it is added to the complementary subset corresponding to the guess list. If it already exists in the subset, it is added to the corresponding intersection subset according to the logical rules. Note that, in a single round of processing, there is a priority for processing passwords in multiple guess lists. For example, in Fig. 9, the priority of the guess lists is the prior(1) > prior(2). In the second round of processing, guess lists 1 and 2 input the passwords "123456" and "password" respectively. The module first queries the password "123456" in the subset. If it exists, it is moved into the subset "11" according to the logical rules. Then the password "password" is queried and moved into the subset "11". This process is repeated until the generated subsets contain enough quantities to generate an optimized guess list. Finally, the module outputs the maintained multi-view subsets.

As this module processes guessed passwords round by round according to the ranking, our framework can also be directly applied to integrate the password guessing models and generate guessed passwords without the need to pre-generate guess lists. The application of the next segment splitting selection module only needs to limit the size of the subset output by the multi-view subset extraction module. This emphasizes that *GuessFuse* can be widely used for the integration and optimization of existing password guessing models. In the practical application of *GuessFuse* for experimental verification, we use the power-law distribution interval generated by Algorithm 1 as sampling points to evaluate the optimized guess list. After outputting the multi-view subsets containing a certain number of guess passwords to the segment splitting selection module and generating the optimized guess list, the multi-view subset extraction module retains these multi-view subsets and removes the passwords that have been used. Then, in the next sampling interval, the module integrates new guess passwords into these multi-view subsets for output.

C. Splitting Selection

After receiving the multi-view subsets, the splitting selection module further refines each subset into multiple password segments according to the power law interval. Based on the average successful cracking rate calculated by the validation set, it reorganizes the effective password segments to generate the final optimized guess list.

1) Motivation: The Zipf's law of passwords in the subset indicates that the cracking efficiency of passwords is inversely proportional to their rank in a power-law relationship. Fig. 7 also shows that, although the overall performance of the intersecting subset is better than other subsets, the lower-ranked passwords in the intersection are less efficient than the higher-ranked passwords in other subsets. Therefore, splitting

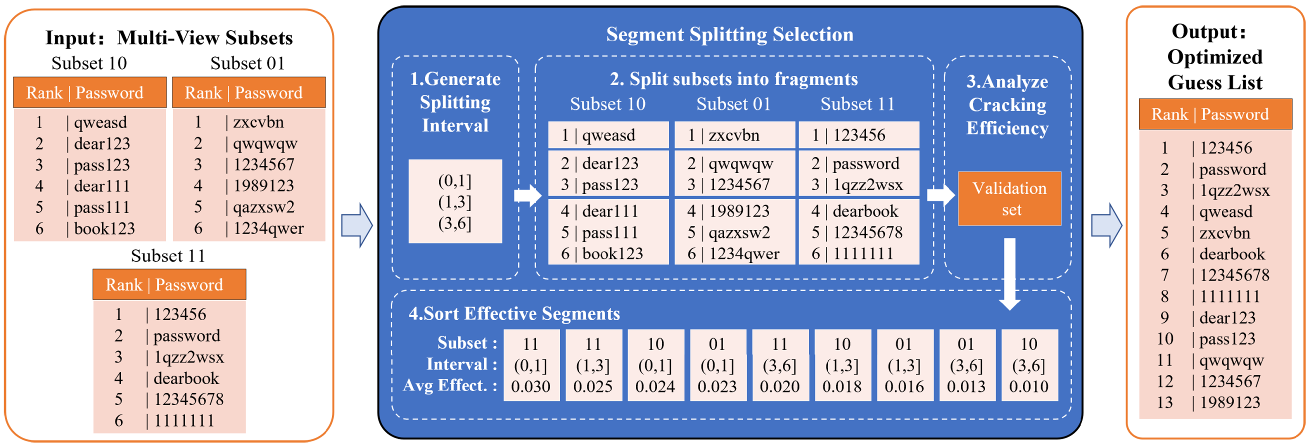


Fig. 10. An example of splitting selection into three multi-view subsets. Note that, for ease of presentation, the presented interval sequence is not strictly generated according to Algorithm 1.

the subsets into multiple password segments according to the power law can accurately extract the effective parts for sorting.

In the scenario of intra-site password guessing, using data samples with the same distribution of the test set to evaluate the cracking efficiency of password segments in the subset can effectively sort them. However, repeated use of the training set may lead to over-fitting of the data and consume excessive computational resources. Therefore, a small part of the training set can be split out as a validation set, used solely for evaluating the cracking efficiency of password segments, and does not participate in the training of individual models. This still has corresponding real-world scenarios in cross-site password guessing, which we will elaborate on in the experimental setup section of Section IV.

Algorithm 1 Power-Based Interval Sequence Generation

```
Require: Optimized guess list size $k$.
Ensure: Interval list $L$.
  $tolerance = 1, temp = 0, basetime = 10, L = [\ ]$
  while $temp < k$ do
    if $temp < basetime$ then
      $L$ append $[temp, temp + tolerance]$
      $[temp, temp + tolerance]$
    else
      $tolerance = basetime$
      $L$ append $[temp, temp + tolerance]$
      $temp += tolerance$
      $basetime = tolerance \times 10$
    end if
  end while
  Return $L$
```

2) Implementation: Similar to Section IV-B, we also introduce an example to illustrate how to implement the segment splitting selection module (see Fig. 10). In order to generate an optimized list containing k guessed passwords, the segmentation selection module initially employs Algorithm 1 to produce a power-based interval sequence with a maximum value of k. Subsequently, the module utilizes this interval sequence to segment the input subsets into finer-grained password segments. Following this, the module employs a validation set to evaluate the cracking efficiency of these password segments, specifically, the average cracking success rate. In detail, the module calculates the average cracking success rate of each password segment with the validation set and then sorts the password segments based on this efficiency. Finally, the module selects the password segments with the highest cracking efficiency, sequentially concatenates them into an optimized guess list, and truncates and outputs this list once it reaches k entries.

At present, we have not found an effective way to pre-estimate the number of guess passwords or the size of the guess list that each integrated password guess model needs to generate in order to create an optimized guess list containing k passwords. We are uncertain about the number of passwords each subset will contain among multiple guess lists, and the maximum interval that needs to be segmented within each subset. Therefore, we can only limit the original guess list and the maximum segmentation interval based on the k size of the optimized guess list. When the passwords in a subset are insufficient to fill the segmentation interval, we discard the tail interval and treat the remaining passwords as a password segment. Fortunately, our analysis of Fig. 4 in Section III and our evaluation experiments indicate that when using an optimized guess list of the same size as the original guess list, its cracking ability also shows a significant improvement.

V. Evaluation

In this section, we conduct a comprehensive and in-depth evaluation of the proposed hybrid password guessing framework. The experiments we designed mainly aim to answer the following three questions: (1) Are all the modules included in *GuessFuse* necessary? (2) Can *GuessFuse* effectively integrate multiple password guessing models? (3) Compared to existing approaches, can the *GuessFuse* further enhance the performance of hybrid password guessing?

A. Experimental Setups

1) Data Preparation: We randomly sampled the datasets into training sets, validation sets, and test sets at a ratio of 8:1:1. Taking the ClixSense dataset as an example, which has the least amount of data among the six datasets, the training set contains approximately 2.9 million passwords. The test set and validation set each contain about 3.6×10^5 passwords.

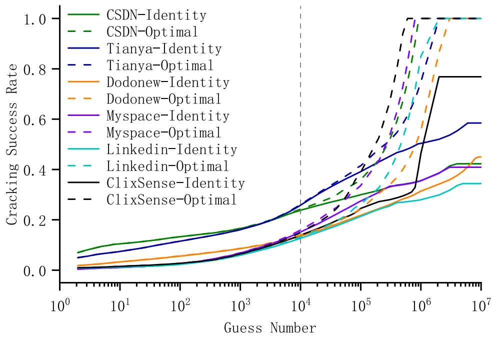


Fig. 11. Comparison of the cracking success rate between the *Identity* guesser and the *Optimal* guesser in the intra-site password guessing scenarios.

2) Experiment Details: To address the three questions posed in the introduction of this section, we designed three experimental schemes: ablation study, effectiveness analysis, and comparative experiments. Firstly, we generated five types of guess lists, a total of 30, based on six training sets. In addition to the guess lists generated by the four password guessing models mentioned earlier (i.e., PCFG, Markov, FLA and RFGuess), we also referred to related researches [34] to generate another type of guess list. We sorted the passwords in the training set according to their frequency to generate a password list. We defined it as the guess list generated by the *Identity* Guesser [34]. Through our comparative analysis of the cracking success rates of five types of guess lists, we set the priority of the guess lists as $prior$(*Identity*) > $prior$(RFGuess) > $prior$(PCFG) > $prior$(FLA) > $prior$(Markov).

We define the cumulative distribution of the password samples in the test set as the cracking capability curve of the *Optimal* Guesser and compare it with the *Identity* Guesser (see Fig11). In the context of in-site password guessing, the *Identity* guesser can approximate the optimal cracking efficiency by guessing the popular passwords in the training set (usually guess number $< 10^4$), which is difficult for other individual password guessing models to achieve. Therefore, when integrating the *Identity* Guesser, we first use the top-10^4 guess lists generated by the *Identity* Guesser as the head of the optimized guess lists, and then use *GuessFuse* to integrate the remaining parts of the guess lists.

In the **ablation study**, we removed the multi-view subset extraction module and segment splitting selection module from *GuessFuse* respectively to generate variants of *GuessFuse* for comparisons. The specific implementations of the variants of *GuessFuse* are as follows:

1) **noSE-*GuessFuse*.** We removed the multi-view subset extraction module and directly used the segment splitting selection module to integrate guess lists.
2) **noSS-*GuessFuse*.** We removed the segment splitting selection module. For the generated multi-view subsets, we sequentially selected those subsets that have the highest cracking success rate for the validation set. We added these subsets to the optimized guess list. Finally, we removed the tail password part that exceeded the required size from the list.

In the **effectiveness analysis**, we used *GuessFuse* and *EqualFuse* to integrate different numbers of password guess lists respectively, and compared the min-auto indicators of different integrations with *GuessFuse*. Among them, *EqualFuse* is a hybrid password guessing method at the guess list level. We implemented it by referring to papers [32], [34]. More specifically, to generate an optimized guess list containing k passwords, we combine the first k/l guessed passwords from l guess lists and deduplicate the combination results.

In the experiments and analyses of this paper, our sampling points are also based on Algorithm 1. This is because the results of the password guessing experiments are widely power-law distributed. That is, when the sampling points are small, the results of the experimental analysis change dramatically, and gradually flatten as the sampling points increase. For some situations where sampling points cannot be accurately used, we will skip this sampling point. For example, when applying *EqualFuse* to integrate four guess lists, using a sampling point of 10 obviously cannot combine passwords in equal amounts, so we skipped this sampling point.

In the **comparative experiments**, we compared *GuessFuse* with *EqualFuse*, and *hyPassGu*. Based on the research [33], we implemented another hybrid password guessing method, *hyPassGu*. This method integrates the PCFG model and the Markov model through model pruning and guess number allocating. Specifically, *hyPassGu* restricts each model to generate only specific types of guess passwords. Then, *hyPassGu* calculates the number of guess passwords that each model needs to generate by statistically analyzing the distribution of specific types of passwords in the training set. Finally, these guess passwords are combined as output.

We did not compare *GuessFuse* with other hybrid password guessing models that integrate at the model architecture level. This is because *GuessFuse* is fundamentally different from them in terms of integration. Precisely because of this, *GuessFuse* directly integrates on the guess list, which can solve the generalization problem that exists when integrating at the model architecture level.

B. Experimental Results

1) Ablation Study: We employ *GuessFuse* and its two variants to integrate the aforementioned five types of guess lists. Subsequently, we quantify the relative improvements of the *GuessFuse* method and its variants within 10^7 guesses in the in-site password guessing scenarios (see Table IV). In addition, we also quantify the relative improvements of the variants using a single module compared to the *Identity* guessers, which have relatively the best cracking capabilities compared to the single guessing models.

As shown in Table IV, *GuessFuse* outperforms noSE-*GuessFuse* by an average of 2.908% to 6.245% across six datasets. This indicates that the use of multi-view subset extraction modules can enhance the cracking capabilities of *GuessFuse*. On the other hand, noSS-*GuessFuse* outperforms the *Identity* Guesser by an average of 0.896% to 5.567%. noSS-*GuessFuse* represents the method of using the multi-view subset extraction module alone and combining it simply. These results reveal that the multi-view subset extraction module significantly affects the cracking success rate.

As for the segment splitting selection module, the cracking capabilities of noSE-*GuessFuse* are on average 0.304% to 2.923% lower than that of the *Identity* guessers, but a max-

TABLE IV

Statistical Information for the Ablation Study. Where A Vs B Represents the Improvement of A Relative to B, Which Is Calculated as $\frac{A-B}{B}$. "noSS" Represents noSS-*GuessFuse*, and "noSE" Represents noSE-*GuessFuse*

Statistic	Dataset	*GuessFuse* vs *		* vs *Identity*	
		noSS	noSE	noSS	noSE
Avg	CSDN	0.08%	2.91%	1.87%	-0.87%
	Dodonew	0.27%	4.30%	0.92%	-2.92%
	Tianya	0.14%	3.25%	0.90%	-2.09%
	Linkedin	0.18%	4.41%	3.92%	-0.15%
	Myspace	0.05%	6.25%	5.57%	-0.30%
	ClixSense	0.29%	3.83%	2.80%	-0.38%
Max	CSDN	0.79%	8.36%	14.32%	13.07%
	Dodonew	3.32%	7.62%	7.69%	4.84%
	Tianya	1.10%	7.63%	9.53%	8.53%
	Linkedin	1.12%	11.77%	25.59%	23.75%
	Myspace	0.57%	15.35%	34.82%	31.77%
	ClixSense	4.42%	11.02%	31.28%	31.93%

imum increase of 4.84% to 31.93%. Meanwhile, GuessFuse, compared to noSS-*GuessFuse* which does not use this module, has an average increase of 0.05% to 0.29%, and a maximum increase of 0.57% to 4.42%. The impact of this module on GuessFuse is not as obvious as that of the multi-view subset extraction module. This module cannot make the average cracking rate of the optimized guessing list higher than the *Identity* guesser. However, its maximum increase relative to the *Identity* guesser and the overall increase of GuessFuse still show its contribution to the integration of the guess lists.

We compared GuessFuse and its two variants and found that GuessFuse can effectively integrate the guess lists. This is mainly because the multi-view subset extraction module can pre-divide the repeated passwords and independent passwords in the guess lists. The repeated passwords in the guess lists will cause simple combinations to generate already guessed passwords, thus wasting the number of cracks. This is also why the cracking ability of noSE-*GuessFuse* is lower than that of the *Identity* guesser. *The repeated and independent parts of the password respectively reflect the consensus and complementary information extracted by different models from the dataset, which should be treated separately.* The experimental results also show that the segment splitting selection module is effective. However, we emphasize that in order to achieve the ideal performance, it needs to be used in conjunction with the multi-view subset extraction module.

2) Effectiveness Analysis: We first used *GuessFuse* and *EqualFuse* to integrate different numbers of guess lists respectively, and analyzed their cracking success rates to compare the effectiveness of them in integrating multiple guess lists. As shown in Fig 12, the cracking success rates of the compared optimized guess lists from high to low are, PFRM-*GuessFuse* > PFR-*GuessFuse* > PF-*GuessFuse* > PF-*EqualFuse* > PFR-*EqualFuse* > PFRM-*EqualFuse*. As the number of integrated guess lists grows, the cracking capabilities of the optimized guess lists output by GUESSFUSE continue to improve. Contrary to *GuessFuse*, the cracking success rate of the optimized guess lists generated by *EqualFuse* decreases as the number of integrations increases. This indicates that *GuessFuse* can effectively integrate multiple guess lists, while simple combinations cannot achieve this effect.

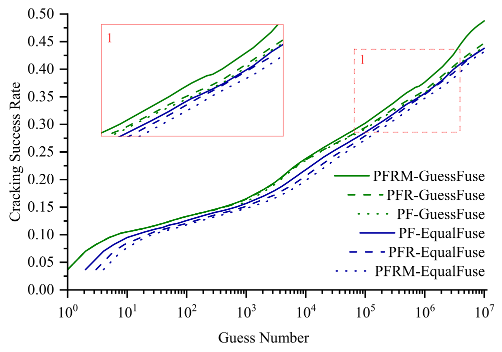


Fig. 12. Comparison of the effectiveness of integrating multiple guess lists in the intra-site password guessing scenarios based on the CSDN dataset. The first half of the legend represents the integrated list of guesses. For example, PFRM-*GuessFuse* represents the optimized guess list generated by integrating PCFG (termed P), FLA (termed F), RFGuess (termed R), and Markov (termed M) models using *GuessFuse*.

TABLE V

The Statistical Information for the Comparisons. Where *coverate* Represents the Success Rate of the Guess List, n Represents the Number of Guesses and the Values Are Calculated as $\frac{A}{B}$ for A Vs B

Statistic	Dataset	*GuessFuse* vs Min_{auto}		PM-*GuessFuse* vs PM-Min_{auto}	
		coverate	n	*coverate*	n
Avg	CSDN	95.18%	39.68%	97.63%	57.47%
	Dodonew	90.19%	43.86%	94.02%	61.91%
	Tianya	94.01%	46.48%	95.33%	60.00%
	Linkedin	90.53%	42.46%	96.73%	59.05%
	Myspace	87.82%	41.49%	90.97%	60.64%
	ClixSense	89.90%	39.32%	96.35%	58.00%
Min	CSDN	88.87%	26.98%	94.46%	52.33%
	Dodonew	84.64%	30.76%	85.07%	55.81%
	Tianya	89.47%	33.45%	89.47%	54.42%
	Linkedin	82.15%	29.86%	91.72%	54.40%
	Myspace	56.27%	25.00%	3.16%	50.00%
	ClixSense	77.26%	29.63%	87.27%	53.86%

We further compared the gap between the cracking capabilities of *GuessFuse* and its upper limit (i.e., the Min_{auto} indicators) when integrating multiple guess lists. Table V presents statistical information on the comparisons. It provides two types of comparisons: *GuessFuse* vs. Min_{auto} reflects the effectiveness of integrating five single guess lists, and PM-*GuessFuse* vs. PM-Min_{auto} reflects the effectiveness of integrating two single guess lists. We calculated the fraction of the cracking success rate (termed $coverate$) achieved by *GuessFuse*, generating optimized guess lists of the same size as the single guess lists, compared to the Min_{auto} indicator. Additionally, we analyzed the fraction of the number of password guesses (termed n) required by the single guess lists compared to the Min_{auto} indicators.

The results show that when combining two single guess lists, *GuessFuse* can select a subset of passwords that represents an average fraction of 57.47% to 61.91% of the total passwords, achieving a success rate ranging from 90.97% to 97.63% of the upper limit indicated by Min_{auto}. It is worth noting that the case with the lowest $coverate$ of only 3.16% in the Myspace dataset is not representative as it involves the fusion of only two passwords. However, *GuessFuse* consistently demonstrates its effectiveness in achieving higher success rates for the other datasets.

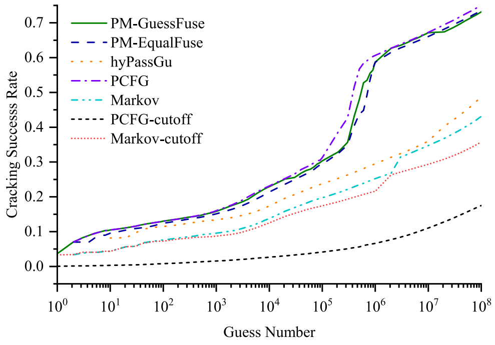


Fig. 13. Comparison of the cracking success rate in the intra-site password guessing scenarios based on CSDN dataset.

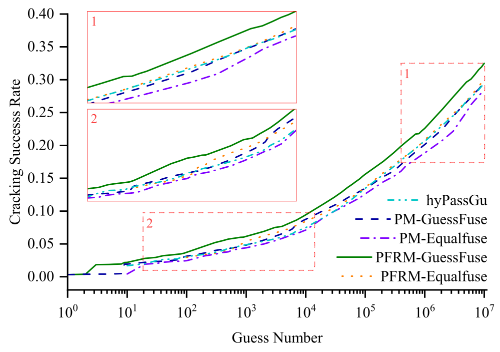


Fig. 14. Comparison of the cracking success rate in the cross-site password guessing scenarios. The experiment extracted the training set from the CSDN dataset, and generated the validation and test sets from the Dodonew dataset using sampling without replacement.

Furthermore, when combining five single guess lists, *GuessFuse* selects a subset of passwords that represents an average proportion of 39.32% to 46.48% of the total passwords, achieving a success rate ranging from 87.82% to 95.18% of the upper limit indicated by Min_{auto}. These findings highlight the effectiveness of *GuessFuse* and indicate that the fusion of more guess lists using *GuessFuse* can effectively reduce resource waste on invalid passwords.

3) Comparative Experiment: We first compared the effectiveness of *GuessFuse*, *EqualFuse*, and *hyPassGu* to integrate guess lists. *hyPassGu* can only integrate guess lists generated by the PCFG and Markov models. Therefore, in the experiments of the intra-site password guessing scenarios, we used *GuessFuse* and *EqualFuse* to integrate the PCFG and Markov models. Due to the restriction of *hyPassGu* on the type of password generated by models, we customized the guess lists generated by the PCFG and Markov models. We deleted the single-character passwords in the PCFG guess list and also deleted the passwords containing more than two types of characters in the Markov guess list. In this way, we formed the PCFG-cutoff and Markov-cutoff guess lists.

As shown in Fig.13, the cracking success rate of PM-*GuessFuse* surpasses its competitors for most guess numbers. Although for certain specific guess numbers (e.g., guess numbers $> 10^7$), the cracking success rate of PM-*GuessFuse* is lower than that of PM-*EqualFuse*. In contrast, the cracking success rate of *hyPassGu* is lower for most guess numbers, only surpassing the guess lists generated solely by the Markov model and the customized guess lists PCFG-cutoff and Markov-cutoff. This indicates that the effectiveness of *GuessFuse* to integrate guess lists surpasses that of *EqualFuse* and *hyPassGu*.

Note that, the cracking success rate of the guess list generated by the PCFG model is superior to other guess lists, and there is a noticeable increase around the guess number of 10^5. This may be due to the specific distribution of the CSDN dataset. We analyzed the distribution of the dataset and found that the passwords with a frequency of 2 to 4 are significantly more than the passwords with a frequency of 1. On the other hand, the bias of the PCFG model and Markov model represented in Fig.2 reveal that the reason for the poor integration effectiveness of *hyPassGu* is that its restriction method for the password types of the models is too crude, resulting in a large waste of effective passwords.

We further analyzed the effectiveness of *GuessFuse*, *EqualFuse*, and *hyPassGu* in cross-site password guessing scenarios.

TABLE VI

STATISTICAL INFORMATION COMPARING *GuessFuse* WITH THE GENERALLY OPTIMAL SINGLE GUESS LISTS AND EXISTING HYBRID PASSWORD GUESSING APPROACHES

Statistic	Dataset	*GuessFuse* vs *		PM-*GuessFuse* vs *	
		EqualFuse	*Identity*	*hyPassGu*	PCFG
Avg	CSDN	4.70%	1.13%	20.19%	0.78%
	Dodonew	5.53%	1.02%	13.27%	1.40%
	Tianya	4.91%	0.78%	23.75%	1.16%
	Linkedin	10.09%	2.44%	45.04%	0.54%
	Myspace	12.74%	2.27%	11.00%	7.73%
	ClixSense	17.66%	2.54%	59.62%	0.09%
Max	CSDN	9.93%	5.51%	34.05%	2.59%
	Dodonew	8.81%	7.89%	25.46%	10.11%
	Tianya	10.91%	5.82%	47.76%	6.94%
	Linkedin	19.89%	17.39%	74.61%	5.63%
	Myspace	69.58%	19.02%	66.61%	56.16%
	ClixSense	87.79%	30.73%	87.67%	2.37%

Specifically, we extracted training samples from one dataset. Based on this, we generated a guess list by a single model. Then, we drew a validation set and a test set from another dataset without replacement. These were used to optimize and test the integration of the guess list. This experimental scenario is realistic. Attackers often use simpler means than password guessing models (such as weak password attack [48]) to obtain a part of the target data. Then, they use the target data to further expand the attack results [12], [27].

As illustrated in Fig.14, the cracking success rate of PFRM-*GuessFuse* significantly surpasses that of other guess-list generation methods, including PM-*GuessFuse*. This suggests that the capabilities of cracking can be substantially enhanced by integrating more guess lists using *GuessFuse*. Despite *hyPassGu* demonstrating comparable performance to PM-*GuessFuse* and PFRM-*EqualFuse* for a majority of guess numbers, its inability to integrate other guess-lists hinders any further improvement in its cracking capability in cross-site password guessing scenarios.

We compare the effectiveness of *GuessFuse* relative to existing methods in the context of intra-site password guessing scenarios. Table VI presents the statistical information for the comparative experiments. The results show that *GuessFuse* can effectively improve the cracking success rate of the optimized guess lists when integrating five or two guess lists.

The dual-integration guess lists generated by *GuessFuse* outperform the generally optimal single guess lists (i.e.,

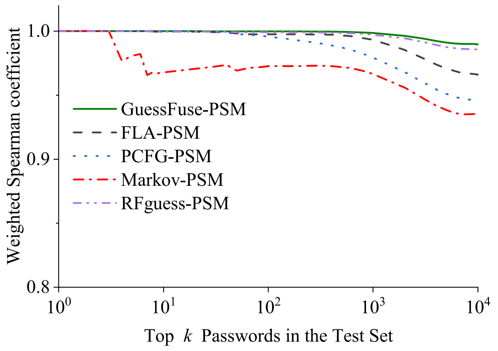


Fig. 15. Weighted Spearman correlation coefficient of five PSMs. *GuessFuse*-PSM outperforms other PSMs in the ranking evaluation of the top-10^4 passwords in the test set.

TABLE VII

THE COUNT OF INCORRECTLY EVALUATED PASSWORDS. THE NUMBERS HIGHLIGHTED IN RED REPRESENT THE COUNT OF PASSWORDS THAT HAVE BEEN OVERESTIMATED, WHILE THE NUMBERS HIGHLIGHTED IN GREEN REPRESENT THE COUNT OF PASSWORDS THAT HAVE BEEN UNDERESTIMATED

Four model-generated guess lists	*GuessFuse* $> 10^0$	$> 10^4$	$> 10^5$	$> 10^6$
$> 10^0$	25154	3564	555	71
$> 10^4$	8220	47297	21169	1389
$> 10^5$	1816	27112	55623	25272
$> 10^6$	1187	2928	58525	78598
$> 10^7$	512	3429	12528	24251

PCFG), with an average improvement in success rate ranging from 0.09% to 7.73%. The five-integration guess lists generated by *GuessFuse* outperform the generally optimal single guess lists (i.e., *Identity*), with an average improvement in success rate ranging from 0.78% to 2.54%. The variation in improvement is because the performance of password guessing also depends on the distribution of the dataset.

Compared to existing hybrid password guessing approaches, *GuessFuse* demonstrates superior performance. On average, *GuessFuse* achieves a higher success rate than *EqualFuse*, by 4.70% to 17.66%. Furthermore, compared to *hyPassGu*, *GuessFuse* achieves an average relative improvement of 25.46% to 87.67% in success rate.

4) Summary: Through ablation study, effectiveness analysis, and comparative experiments, we obtained the following results. (1) Both modules of *GuessFuse* are effective in integrating multiple guess lists. Among them, the multi-view subset extraction module makes a major contribution to enhancing the effectiveness of integration. The conjunction of the two modules can achieve better results. (2) *GuessFuse* can effectively integrate multiple guess lists, and the more it integrates, the stronger the cracking capability. (3) *GuessFuse* performs better than existing hybrid password guessing methods in terms of generalization and effectiveness.

VI. APPLICATION & DISCUSSION

1) Application of PSM: We applied *GuessFuse* mentioned in Section V to create a PSM, named *GuessFuse*-PSM, and compared it with PSMs implemented using four individual password guessing models (i.e., PCFG, Markov, FLA and RFGuess). We calculated the Weighted Spearman correlation coefficient (Wspearman) between the top-10^4 ranked samples of the ideal PSM (i.e., the top-10^4 passwords in the *Optimal* guess lists in the test sets) and the five PSMs mentioned above (i.e., the rank of passwords in each model-generated guess list). The experimental setup was consistent with [49] and [50]. We used the CSDN dataset as an example, and the Wspearman coefficient curves are presented in Fig. 15. It should be noted that *GuessFuse*-PSM includes the *Identity* guess list in its fusion process, which results in the same Wspearman coefficient as the *Identity* guess list. The comparison revealed that *GuessFuse*-PSM achieved the best performance within the top-10^4 passwords.

Furthermore, we analyzed the *count of incorrectly evaluated passwords*. This evaluation metric is similar to the concept of "*unsafe errors*" in previous studies [15], [28]. We believe that incorrect password evaluations not only provide insecure guidance to users (i.e., overestimating password strength), but also reduce the usability of password setting (i.e., underestimating password strength, thus restricting users from setting passwords that are actually strong enough). Therefore, we deliberately named this metric "*count of incorrectly evaluated passwords*" instead of "*unsafe errors*."

We used the CSDN dataset as an example, and the analysis results are presented in Table VII. Based on Finding 2, we consider that the strength of the top-10^4 ranked passwords should belong to the same category (i.e., common passwords). We have also analyzed the Wspearman coefficient for the top-10^4 passwords in *GuessFuse*-PSM. Hence, we grouped them into the same interval. We assume that passwords within the same magnitude should have the same evaluation of password strength. It can be observed that the PSMs of the four individual models tend to overestimate or underestimate the strength of certain passwords to different extents compared to *GuessFuse*-PSM. However, none of them demonstrated the same level of password guessing capability as modeled by *GuessFuse*-PSM. Therefore, utilizing *GuessFuse*-PSM not only enhances the security of user-set passwords but also improves the usability of password settings for users.

2) Application of This Study: *GuessFuse* requires an optimization after each password guessing model generates a guess list, which consumes more computing power than a single password guessing model. However, *GuessFuse* can effectively improve the cracking success rate of a single password guessing model within a limited number of guesses. In online guessing scenarios or situations where password encryption is complex (i.e., slow-HASH [51] and memory-hard HASH [52], which are widely used to prevent offline password guessing attacks [53], [54], [55]), the cost of testing a password is much higher than the resource consumption involved in generating a password. Therefore, *GuessFuse* still holds significant practical value. On the other hand, *GuessFuse* only performs optimization at the guess list level, which has universal adaptability. Compared with other hybrid password guessing approaches, users of our framework can enhance them without understanding the specific design of every single model. They only need to obtain the guess lists generated by every single model to achieve a higher password cracking success rate.

3) Future Work: Our proposed password guessing framework provides a new research direction for hybrid password guessing: finding the practical parts of password guess lists generated by each guessing model. In our analysis, we find that the password distribution of the model output guess list is strongly related to the distribution of the training dataset, so machine learning models can be used to associate the dataset distribution with the guess list quantity output by multiple models for prediction, thus restricting the number of model guesses during the guess list generation process. We will further investigate this in the future.

VII. Conclusion

In this paper, we have made a substantial step towards hybrid password guessing based on guess lists. More specifically, we systematically analyze the multiple guess lists generated from various password guessing models. Through analysis, we have obtained six valuable findings for hybrid password guessing. Based on this, we design a new hybrid password guessing framework, named *GuessFuse*. Extensive experiments demonstrate that *GuessFuse* can effectively combine multiple password guess lists, enhance the success rate of the output, and outperform existing hybrid password guessing approaches. This work pioneers a novel research trajectory in hybrid password guessing.

References

[1] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Trans. Depend. Secur. Comput.*, vol. 15, no. 4, pp. 708–722, Sep. 2016.

[2] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: A tool for information security," *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 2, pp. 125–143, Jun. 2006.

[3] C. Herley and P. van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Secur. Privacy*, vol. 10, no. 1, pp. 28–36, Jan. 2012.

[4] J. Bonneau, C. Herley, P. C. V. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 553–567.

[5] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Commun. ACM*, vol. 58, no. 7, pp. 78–87, Jun. 2015.

[6] M. Weir, S. Aggarwal, B. D. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. 30th IEEE Symp. Secur. Privacy*, May 2009, pp. 391–405.

[7] J. Blocki, B. Harsha, and S. Zhou, "On the economics of offline password cracking," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 35–53.

[8] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1242–1254.

[9] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 417–434.

[10] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 538–552.

[11] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 657–666.

[12] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from Markov models," in *Proc. NDSS*, 2012, pp. 1–14.

[13] B. Ur et al., "How does your password measure up? The effect of strength meters on password creation," in *Proc. 21st USENIX Secur. Symp.*, 2012, pp. 65–80.

[14] D. L. Wheeler, "zxcvbn: Low-budget password strength estimation," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 157–173.

[15] W. Melicher et al., "Fast, lean, and accurate: Modeling password guessability using neural networks," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 175–191.

[16] B. Ur et al., "Measuring real-world accuracies and biases in modeling password guessability," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 463–481.

[17] Q. Dong, C. Jia, F. Duan, and D. Wang, "RLS-PSM: A robust and accurate password strength meter based on reuse, leet and separation," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4988–5002, 2021.

[18] D. Wang, D. He, H. Cheng, and P. Wang, "FuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2016, pp. 595–606.

[19] P. G. Kelley et al., "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 523–537.

[20] J. Steube. (2018). *Hashcat*. [Online]. Available: https://hashcat.net/hashcat/

[21] A. Peslyak. (Feb. 1996). *John the Ripper Password Cracker*. [Online]. Available: http://www.openwall.com/john/

[22] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM Conf. Comput. Commun. Secur.*, Nov. 2005, pp. 364–372.

[23] B. Hitaj, P. Gasti, G. Ateniese, and F. Perez-Cruz, "PassGAN: A deep learning approach for password guessing," in *Proc. ACNS*, 2019, pp. 217–237.

[24] D. Wang, Y. Zou, Z. Zhang, and K. Xiu, "Password guessing using random forest," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 965–982.

[25] R. Veras, C. Collins, and J. Thorpe, "On the semantic patterns of passwords and their security impact," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–16.

[26] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 689–704.

[27] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 1382–1399.

[28] M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, "Chunk-level password guessing: Towards modeling refined password composition representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 5–20.

[29] M. L. Zhang, Q. H. Zhang, W. F. Liu, X. X. Hu, and J. H. Wei, "A method of password attack based on structure partition and string reorganization," *Chin. J. Comput.*, vol. 42, no. 4, pp. 914–928, 2019. [Online]. Available: http://cjc.ict.ac.cn/online/onlinepaper/zml-201941793554.pdf

[30] M. Zhang, Q. Zhang, X. Hu, and W. Liu, "A password cracking method based on structure partition and BiLSTM recurrent neural network," in *Proc. 8th Int. Conf. Commun. Netw. Secur.*, Nov. 2018, pp. 79–83, doi: 10.1145/3290480.3290501.

[31] Z. Xia, P. Yi, Y. Liu, B. Jiang, W. Wang, and T. Zhu, "GENPass: A multi-source deep learning model for password guessing," *IEEE Trans. Multimedia*, vol. 22, no. 5, pp. 1323–1332, May 2020.

[32] D. Wang, Y. Zou, Y. Tao, and B. Wang, "Password guessing based on recurrent neural networks and generative adversarial networks," *Chin. J. Comput.*, vol. 44, no. 8, pp. 1519–1534, 2021. [Online]. Available: http://cjc.ict.ac.cn/online/onlinepaper/wd-20218794702.pdf

[33] W. Han et al., "Parameterized hybrid password guessing method," *J. Comput. Res. Develop.*, vol. 59, p. 2708, Jul. 2022. [Online]. Available: https://crad.ict.ac.cn/article/doi/10.7544/issn1000-1239.20210456

[34] Z. Parish, C. Cushing, S. Aggarwal, A. Salehi-Abari, and J. Thorpe, "Password guessers under a microscope: An in-depth analysis to inform deployments," *Int. J. Inf. Secur.*, vol. 21, no. 2, pp. 409–425, Apr. 2022.

[35] R. Morris and K. Thompson, "Password security: A case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, Nov. 1979.

[36] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proc. 2nd USENIX Secur. Workshop*, 1990, pp. 5–14.

[37] T. D. Wu, "A real-world analysis of Kerberos password security," in *Proc. NDSS*, 1999, pp. 13–22.

[38] 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.

[39] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[40] S. Sun, "A survey of multi-view machine learning," *Neural Comput. Appl.*, vol. 23, no. 7, pp. 2031–2038, 2013.

[41] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," 2013, *arXiv:1304.5634*.

[42] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1537–1555.

[43] D. Wang, Y. Zou, Y. A. Xiao, S. Ma, and X. Chen, "Pass2Edit: A multi-step generative model for guessing edited passwords," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 983–1000.

[44] M. Dell'Amico and M. Filippone, "Monte Carlo strength evaluation: Fast and reliable password checking," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 158–169.

[45] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[46] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2776–2791, Nov. 2017.

[47] Y. Wu, D. Wang, Y. Zou, and Z. Huang, "Improving deep learning based password guessing models using pre-processing," in *Proc. ICICS*, vol. 13407, 2022, pp. 163–183.

[48] E. R. Verheul, "Selecting secure passwords," in *Topics in Cryptology—CT-RSA 2007*. Berlin, Germany: Springer, 2006, pp. 49–66.

[49] D. Wang, X. Shan, Q. Dong, Y. Shen, and C. Jia, "No single silver bullet: Measuring the accuracy of password strength meters," in *Proc. 32nd USENIX Secur. Symp. (USENIX Security)*, 2023, pp. 1–28.

[50] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1567–1582.

[51] P. Sriramya and R. Karthika, "Providing password security by salted password hashing using Bcrypt algorithm," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 13, pp. 5551–5556, 2015.

[52] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: New generation of memory-hard functions for password hashing and other applications," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, May 2016, pp. 292–302.

[53] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital identity guidelines," NIST, Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-63-3, 2017, vol. 800, pp. 3–63.

[54] K. Moriarty, B. Kaliski, and A. Rusch, *PKCS #5: Password-Based Cryptography Specification Version 2.1*, document RFC 8018, 8018. [Online]. Available: https://www.rfc-editor.org/info/rfc8018

[55] F. G. G. Meade, "Information assurance technical framework (IATF)," Nat. Secur. Agency (NSA), Fort Meade, MD, USA, Tech. Rep. ADA606355, 2002.

Zhijie Xie received the M.S. degree from the National University of Defense Technology, where he is currently pursuing the Ph.D. degree in cyberspace security. His research interests include password-based authentication security, social engineering, and data mining.

Fan Shi received the M.S. degree from the Electronic Engineering Institute. Currently, he is an Associate Professor with the National University of Defense Technology. His research interests include cyberspace surveying and mapping, network security knowledge graphs, and cyberspace content security.

Min Zhang received the Ph.D. degree from Anhui University. He is currently a Full Professor with the National University of Defense Technology. His research interests include intelligent information processing, machine learning, and data mining.

Huimin Ma received the M.S. degree from North China Electric Power University. She is currently an Associate Professor with the National University of Defense Technology. Her research interests include network security protection.

Huaixi Wang received the Ph.D. degree from Peking University. He is currently an Associate Professor with the National University of Defense Technology. His research interests include cryptography, information security, and cloud computing security.

Zhenhan Li received the M.S. degree in communication engineering from Xiamen University. He is currently with the National University of Defense Technology. His research interests include network security protection and social engineering.

Yunyi Zhang received the M.S. degree from Sun Yat-sen University. He is currently pursuing the Ph.D. degree in cyberspace security with the National University of Defense Technology. His research interests include network security, DNS security, and cybercrime.