Honeywords Generation Mechanism Based on Zero-Divisor Graph Sequences

Yanzhao Tian, Lixiang Li, Haipeng Peng, Ding Wang, and Yixian Yang

***Abstract*—The identity authentication of most applications is based on a symbolic password. However, incidents of password leakage emerge one after another, which brings serious hidden danger to the users' information security. For decades, various schemes have been proposed to solve the problem of information protection. However, most schemes neglect the timely detection of password leakage. The present paper introduces a password leak detection method based on zero-divisor graph sequences. Specifically, it is to construct an algorithm for generating honeywords with high smoothness. First, we introduce the concept of the zero-divisor graph and construct zero-divisor graph sequences by using the corresponding zero-divisor matrices. Second, the honeywords with high flatness are constructed by using the sequence of zero-divisor graphs. Third, the security analysis verifies the effectiveness of the scheme. Fourth, compared with other honeywords schemes, our scheme has more obvious advantages, in the aspects of honeywords generated flatness, DoS resistance, and storage resources occupied by honeywords.**

Q1

Q2 ***Index Terms*—Authentication, honeywords, zero-divisor graph matrix, zero-divisor graph, graphic labeling, topological coding.**

I. Introduction

WITH the increase of network bandwidth and the reduction of its fees, people can realize office, shopping, entertainment, navigation, and payment through the network. However, with the enrichment of application scenarios, network security issues have become more and more prominent. Among them, password leakage for identity authentication is the most concerning security problem. Recently, many well-known websites, such as Yahoo, Facebook, Google, Bilibili [1], [2], [3], [4], have been exposed that the registered username and the registered user password have been leaked. To improve the security of password file protection, one of the ideas is to design an authentication method to replace text passwords. Typical image recognition authentication mechanism such as Windows, Shoulder surfing attack is the biggest defect faced by such authentication method. Another idea is to design a detection method for password leakage. If the leak of passwords can be detected in time and replaced or reset, the risk of attack can be effectively reduced, and the cost of this idea is relatively inexpensive. The References [5], [6] shows that some network service providers will use the recommended method to salt and hash passwords, but the attackers use machine learning guessing algorithms and general hardware such as GPUs to improve guessing speed and can recover a considerable number of passwords in an acceptable amount of time. So it is necessary to reconsider the password protection method from the perspective of password leakage detection.

Juels and Rivest [7] proposed a password leak detection technology-based the honeywords. The user's real password is mixed with $k-1$ honeywords (false password) as the user's 'password'. If the honeywords generation method is flat enough, the adversary can't differentiate the user's real password from the user's sweetwords file set, even if he reverses the hash file of the user's password. At the same time, the adversary logs in to the server with the honeywords, which can be detected by the system. At present, many honeywords generation schemes have been proposed [8], [9], [10], [11], [12]. These schemes have been proposed based on two strategies: One is to design a honeypot account to improve the security of user registration information. The other is to design honeywords (fake passwords) to generate $k-1$ fake passwords for each account, to enhance the privacy and complexity of the user's password. Guo et al. [13] constructed a matching attack model. In this attack model, some honeywords schemes meet the requirement of perfect flatness, but the adversary can still achieve a high attack success rate. Camenisch et al. [14] constructed a multi-server-oriented protocol for distributing authenticated passwords, which can resist offline dictionary attacks. Wang et al. [15] studied the generation of the honeywords based on the combination strategies of different

Manuscript received 8 May 2022; revised 14 October 2023; accepted 28 October 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1805403, in part by the National Natural Science Foundation of China under Grant 62032002, in part by 111 Project under Grant B21049, and in part by the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2023-1-07.
Q3 Recommended for acceptance by F. Liu. *(Corresponding author: Lixiang Li.)*

Yanzhao Tian is with the School of Cyberspace Security, Hainan University, Haikou 570208, China, and with the Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: tiany2048@hainanu.edu.cn).

Lixiang Li, Haipeng Peng, and Yixian Yang are with the Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: lixiang@bupt.edu.cn; penghaipeng@bupt.edu.cn; yxyang@bupt.edu.cn).

Ding Wang is with the College of Cyber Science, Nankai University, Tianjin 300350, China, and also with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China (e-mail: wangdingg@mail.nankai.edu.cn).

Digital Object Identifier 10.1109/TSC.2023.3329013

attacks. Dionysiou et al. [16] proposed a honeyword generation approach based on word representation learning, and adjusted chaffing-by-tweaking by replacing letters with upper and lower case and selecting different probability symbols for specific symbols.

The key problem of the honeywords scheme is how to generate effective honeywords, which means to make them indistinguishable from the user's real password. In Ref. [17], the authors also clearly pointed out this point. The statistical characteristics of the user passwords meet the Zipf-like distribution [18], so it is not enough security for honeywords based on user passwords. From Refs. [19], [20], it was found that graph labeling can be used to construct various topological graphs, and the topological matrix corresponding to this topological graph has a huge generating space. Wang et al. [21] proposed graphical passwords based on the graphic labeling, which provided an idea for us to design a honeywords scheme.

Therefore, we design a honeywords generation scheme based on the zero-divisor graph and graphic labeling. We propose a ZDG generation algorithm, which is easy to deploy in the honeywords verification server. Through the analysis of security, flatness, storage overhead, and other aspects, our honeywords scheme has better advantages. The adversary can be detected by a honeywords verification system constructed by randomly selecting the ZDG. We summarize the key contributions of this paper as follows.

- In Section II, we give the definition of the ternary zero-divisor under the congruence relation, and based on the prime decomposition theorem, we give the calculation method of the ternary zero-divisor.
- In Section III, we construct a honeywords generation scheme based on the ZDG sequences. The generation of the ZDG sequences is combined with solving the 3-tuples congruence equation. To overcome the semantic statistical characteristics of natural language and improve the honeywords flatness, the ZDG sequences are used to construct the honeywords.
- In Section IV, we analyze the security of the proposed scheme for several attack scenarios. The generating space of the zero-divisor graph sequence ensures the diversity of generating honeywords and enhances the security of honeywords.
- In Section V, from the aspects of flatness, storage overhead, and DoS attack resistance, we analyze the comprehensive performance of the proposed scheme. According to different lengths of ASCII code corresponding to username, the availability and security of user password files can be enhanced by selecting different numbers of zero-divisor graph sequences.

II. Preliminary

With the development of computer technology, graph theory and algebra have become important theoretical tools to study computer science. More and more scholars are paying attention to the relationship between algebra and graph theory [22], [23], [24], [25], [26], [27], [28]. The abundant results of algebra are applied to the study of graph theory. One of its applications is to describe the properties of graph theory by using zero-divisor graphs over commutative rings. In 1988, I. Beck [29] illustrated the concept of zero-divisor graphs of commutative rings R. The relationships between zero-divisor graphs and ring have been deeply studied [30], [31], [32], [33]. In this paper, we study the properties of zero-divisor graphs satisfying congruence relation. Specifically, we use zero-divisor graphs to study the generation of honeywords.

First, we give some definitions of the zero-divisor graph and graphic labeling. A more detailed introduction can be found in Refs. [34], [35]. In Table I, we list some important symbols used in this paper.

Definition 1: Let Z_n be a commutative ring with identity. We define the ZDG of Z_n, to be a simple graph with vertex set being the set of non-zero zero-divisors of Z_n and with (x, y, z) a vertex-edge-vertex if and only if $xyz \equiv 0 \, mod \, N$, $N \in Z_n$, the non-zero elements x, y, and z are called zero-divisors.

Definition 2: Let G be a (p, q)-graph, the vertex set of a graph G is referred to as $V(G)$, its edge set as $E(G)$. If there exist a constant N and a mapping F: $V(G) \cup E(G) \rightarrow [1, 2q+1]$, such that $F(u) \cdot F(v) \cdot F(uv) \equiv 0 \, mod \, N$ for every edge $uv \in E(G)$, then the F is generalized edge-magic labeling of G, and N is a zero-divisor constant. If G is a bipartite graph with bipartition (V_1, V_2), $F(V(G)) = [1, q+1]$ and $F(V_1) < F(V_2)$ hold, F is called a generalized super set-ordered edge labeling. The symbol $[1, 2q+1]$ represents the set of $2q+1$ positive integers between 1 and $2q+1$.

Definition 3: ([35]) A topological coding matrix is defined as

$$T = \begin{pmatrix} x_1 & x_2 & \cdots & x_q \\ e_1 & e_2 & \cdots & e_q \\ y_1 & y_2 & \cdots & y_q \end{pmatrix} = \begin{pmatrix} X \\ E \\ Y \end{pmatrix}_{3\times q} = \begin{pmatrix} X & E & Y \end{pmatrix}^{T}_{3\times q}, \tag{1}$$

where $v - vector \;\; X = (x_1, x_2, \ldots, x_q)$, $e - vector \;\; E = (e_1, e_2, \ldots, e_q)$, and $v - vector \; Y = (y_1, y_2, \ldots, y_q)$ consist of integers e_i, x_i and y_i for $i \in [1, q]$. The Topcode-matrix T_{code} can be calculated if there exists a function F such that $e_i = F(x_i, y_i)$ for $i \in [1, q]$, and call x_i and y_i to be the ends of e_i.

Definition 4: ([36]) An graph isomorphism from a simple graph G to a simple graph H is a bijection $f : V(G) \rightarrow V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. G is isomorphic to H, written $G \cong H$, if there is an isomorphism from G to H.

Theorem 1: Suppose x, y, z are positive integers greater than 1 and less than N, where $x \neq y \neq z$, and satisfy the following equation

$$xyz \equiv 0 \, mod \, N, \; N \in N^{*}. \tag{2}$$

When N is taken of different values, the number of triples (x, y, z) formed by the solutions to (2) is given

TABLE I
SYMBOL ABBREVIATION COMMENT TABLE

Symbol	Description	Symbol	Description
ZDG	zero-divisor graph	T_{code}	topological coding matrix
N_{set}	the set of solutions for all the positive integers satisfying 3-tuples congruence equation	N	the modulus of the congruence equation
$\mathscr{P}(X)$	the set of subsets of the set X	C_{number}	the column number in the ZDG matrix
$\|A\|$	the size of a finite set A	$L_{U/P}$	the lengths of the username (password)
C_n^m	combinatorial number	H	the set of hash values
$G_{N,T}$	ZDG with module N and triples T	$\oplus$	symmetric difference
L_{ID}	the binary lengths of the username ID	L_H	the binary lengths of the hashed sequence value
$L_{T_{code}}$	the binary lengths of the ZDG sequences	L_M	the binary lengths of the number of the ZDG sequences
L_{S_I}	the binary lengths of the ZDG sequences index	L_j	the binary lengths of the index of correct passwords in hash sequences
L_{AUP_i}	the lengths of the ASCII code of the username	L_{TS}	the lengths of the topological sequence index
P_{ZDG}	the value of corresponding position of the ZDG sequence	P_{ASCII}	the value of corresponding position of the ASCII code

by

$$G_N = \begin{cases} 0 & \text{N is prime} \\ p^2q^2 - \frac{3}{2}p^2q - \frac{3}{2}pq^2 + \frac{1}{2}p^2 + \frac{1}{2}q^2 + \frac{3}{2}p + \frac{3}{2}q - 2 & N = pq \\ \frac{1}{2}p^4 - \frac{11}{6}p^3 + p^2 + \frac{7}{3}p - 2 & N = p^2 \\ 4p^2q^2r^2 - \frac{9}{2}p^2q^2r - \frac{9}{2}p^2qr^2 - \frac{9}{2}pq^2r^2 + \frac{3}{2}p^2q^2 + \frac{9}{2}p^2qr + \frac{3}{2}p^2r^2 + \frac{9}{2}pq^2r + \frac{9}{2}pqr^2 + \frac{3}{2}q^2r^2 - \frac{3}{2}p^2q - \frac{3}{2}p^2r - \frac{3}{2}pq^2 - 10pqr - \frac{3}{2}pr^2 - \frac{3}{2}q^2r - \frac{3}{2}qr^2 + \frac{1}{2}p^2 + \frac{11}{2}pq + \frac{11}{2}pr + \frac{1}{2}q^2 + \frac{11}{2}qr + \frac{1}{2}r^2 - \frac{5}{2}p - \frac{5}{2}q - \frac{5}{2}r & N = pqr \\ \frac{7}{6}p^6 - \frac{5}{2}p^5 + \frac{1}{2}p^3 + \frac{7}{3}p^2 + \frac{1}{2}p - 2 & N = p^3 \\ \frac{5}{2}p^4q^2 - 3p^4q - 4p^3q^2 + p^4 + 3p^3q + \frac{3}{2}p^2q^2 - \frac{5}{6}p^3 - 3p^2q + \frac{5}{2}p^2 + 3pq - \frac{2}{3}p - 2 & N = p^2q \end{cases}.$$

For the convenience of calculation, the symbol N_{set} represents the set of solutions for all the positive integers satisfying the 3-tuples congruence equation, G_N represents the number of elements in the set N_{set}, S represents the set that contains i, $1 < i < N$, $\complement_S A$ represents the set that contains i, $i \notin A$, $i \in S$, and A_p represents the set that contains multiple of p.

Proof: According to definition 1, if $1 < N < 6$, the set N_{set} does not exist. if N is an arbitrary prime number, then the set N_{set} does not exist.

In the following discussion, we assume that $N \geq 6$. Based on the fundamental theorem of arithmetic, the prime factorization formula of the N is

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_s^{\alpha_s}, \tag{3}$$

where $p_1 < p_2 < \cdots < p_s$, $\alpha_i \geq 1$, $i \in [1, s]$.

We discuss the calculation of G_N in the following decomposition cases of N, which is similar to that in other cases. Our basic idea is that, the multiples of prime factors form several disjoint sets. We take any three sets from these disjoint sets and take an element from these three sets. If the product of these three elements is a multiple of N, then this triple is the solution of the (2). The solution of G_N is closely related to the decomposition formula of N, and with the increase of the prime factor of N, the formula of G_N becomes more and more complex, and it is difficult to characterize it with a unified formula. We will discuss it in detail below.

i) $N = pq$

The number of elements $|\complement_S(A_p \cup A_q)|$ is $pq - p - q$, the number of elements $|A_p|$ is $q - 1$, and the number of elements $|A_q|$ is $p - 1$. By the permutation and combination formula, we have

$$\begin{aligned} G_N &= \mathrm{C}_{A_p}^1 \mathrm{C}_{A_q}^1 \mathrm{C}_{\complement_S(A_p \cup A_q)}^1 + \mathrm{C}_{A_p}^2 \mathrm{C}_{A_q}^1 + \mathrm{C}_{A_q}^2 \mathrm{C}_{A_p}^1 \\ &= p^2q^2 - \frac{3}{2}p^2q - \frac{3}{2}pq^2 + \frac{1}{2}p^2 + \frac{1}{2}q^2 + \frac{3}{2}p + \frac{3}{2}q - 2. \end{aligned} \tag{4}$$

In this case, G_N is calculated from (4).

ii) $N = p^2$

The number of elements $|A_p|$ is $p - 1$. Since $i \in S$, $2 \leq i < p^2$, the number of elements $|S|$ is $p^2 - 2$, and the number of elements $|\complement_S A_p|$ is $p^2 - p - 1$. By the permutation and combination formula, we can get

$$\begin{aligned} G_N &= \mathrm{C}_{A_p}^2 \mathrm{C}_{\complement_S A_p}^1 + \mathrm{C}_{A_p}^3 \\ &= \frac{1}{2}p^4 - \frac{11}{6}p^3 + p^2 + \frac{7}{3}p - 2. \end{aligned} \tag{5}$$

In this case, G_N is calculated from (5).

iii) $N = pqr$

The number of elements $|A_p|$ is $qr - 1$, the number of elements $|A_q|$ is $pr - 1$, the number of elements $|A_r|$ is $pq - 1$, the number of elements $|A_{pq}|$ is $r - 1$, the number of elements $|A_{pr}|$ is $q - 1$, the number of elements $|A_{qr}|$ is $p - 1$, the number of elements $|\complement_S(A_p \cup A_q \cup A_r)|$ is $pqr - pq - pr - qr + p + q + r - 2$, the number of elements $|A_p - A_{pq} - A_{pr}|$ is $qr - q - r - 1$, the number of elements $|A_q - A_{pq} - A_{qr}|$ is $pr - p - r - 1$, and the number of elements $|A_r - A_{pr} - A_{qr}|$ is $pq - p - q - 1$. $\mathscr{P}(A)_1$ represents the set $\{A_q, A_r, A_{pq}, A_{pr}\}$, $\mathscr{P}(A)_2$ represents the set $\{A_r, A_{pq}, A_{qr}\}$, $\mathscr{P}(A)_3$ represents the set $\{A_{pr}, A_{qr}\}$, $\mathscr{P}(A)_4$ represents the set $\{A_p, A_r, A_{pq}, A_{qr}\}$, and $\mathscr{P}(A)_5$ represents the set $\{A_p, A_q, A_{pr}, A_{qr}\}$. For the sake of writing conveniently, $A_p - A_{pq} - A_{pr}$ is abbreviated as A_p^*, $A_q - A_{pq} - A_{qr}$ is abbreviated as A_q^*, $A_r - A_{pr} - A_{qr}$ is abbreviated as A_r^*, and $\complement_S(A_p \cup A_q \cup$

A_r) is abbreviated as $A^*_{\cup}$. By the permutation and combination formula, we have

$$
\begin{aligned}
G_N &= \mathrm{C}^3_S - \mathrm{C}^3_{A^*_{\cup}} - \mathrm{C}^2_{A^*_{\cup}} \sum_{i\in\mathscr{P}(A)} \mathrm{C}^2_i - \mathrm{C}^1_{A^*_{\cup}}\mathrm{C}^1_{A^*_p} \sum_{i\in\mathscr{P}(A)_1} \mathrm{C}^1_i \\
&\quad - \mathrm{C}^1_{A^*_{\cup}}\mathrm{C}^1_{A^*_q} \sum_{i\in\mathscr{P}(A)_4} \mathrm{C}^1_i - \mathrm{C}^1_{A^*_{\cup}}\mathrm{C}^1_{A^*_r} \sum_{i\in\mathscr{P}(A)_3} \mathrm{C}^1_i \\
&\quad - \mathrm{C}^1_{A^*_p}\mathrm{C}^1_{A^*_q}\mathrm{C}^1_{A_{pq}} - \mathrm{C}^1_{A^*_p}\mathrm{C}^1_{A^*_r}\mathrm{C}^1_{A_{pr}} - \mathrm{C}^1_{A^*_q}\mathrm{C}^1_{A^*_r}\mathrm{C}^1_{A_{qr}} \\
&\quad - \mathrm{C}^1_{A^*_{\cup}} \sum_{i\in\mathscr{P}(A)} \mathrm{C}^2_i - \sum_{i\in\mathscr{P}(A)} \mathrm{C}^3_i - \mathrm{C}^2_{A^*_p} \sum_{i\in\mathscr{P}(A)} \mathrm{C}^1_i \\
&\quad - \mathrm{C}^2_{A^*_q} \sum_{i\in\mathscr{P}(A)_4} \mathrm{C}^1_i - \mathrm{C}^2_{A^*_r} \sum_{i\in\mathscr{P}(A)_5} \mathrm{C}^1_i \\
&\quad - \mathrm{C}^2_{A_{pq}}(\mathrm{C}^1_{A^*_p} + \mathrm{C}^1_{A^*_q}) - \mathrm{C}^2_{A_{pr}}(\mathrm{C}^1_{A^*_p} + \mathrm{C}^1_{A^*_r}) \\
&\quad - \mathrm{C}^2_{A_{qr}}(\mathrm{C}^1_{A^*_q} + \mathrm{C}^1_{A^*_r}) \\
&= 4p^2q^2r^2 - \frac{9}{2}p^2q^2r - \frac{9}{2}p^2qr^2 - \frac{9}{2}pq^2r^2 + \frac{3}{2}p^2q^2 \\
&\quad + \frac{9}{2}p^2qr + \frac{3}{2}p^2r^2 + \frac{9}{2}pq^2r + \frac{9}{2}pqr^2 + \frac{3}{2}q^2r^2 - \frac{3}{2}p^2q \\
&\quad - \frac{3}{2}p^2r - \frac{3}{2}pq^2 - 10pqr - \frac{3}{2}pr^2 - \frac{3}{2}q^2r - \frac{3}{2}qr^2 + \frac{1}{2}p^2 \\
&\quad + \frac{11}{2}pq + \frac{11}{2}pr + \frac{1}{2}q^2 + \frac{11}{2}qr + \frac{1}{2}r^2 - \frac{5}{2}p - \frac{5}{2}q - \frac{5}{2}r.
\end{aligned} \tag{6}
$$

In this case, G_N is calculated from (6).

iv) $N = p^3$

The number of elements $|A_p|$ is $p^2 - 1$, the number of elements $|A_{p^2}|$ is $p - 1$, and the number of elements $|\complement_S A_p|$ is $p^3 - p^2 - 1$. By the permutation and combination formula, we have

$$
\begin{aligned}
G_N &= \mathrm{C}^1_{A_p - A_{p^2}}\mathrm{C}^1_{A_{p^2}}\mathrm{C}^1_{\complement_S A_p} + \mathrm{C}^2_{A_p - A_{p^2}}\mathrm{C}^1_{A_{p^2}} \\
&\quad + \mathrm{C}^3_{A_p - A_{p^2}} + \mathrm{C}^2_{A_{p^2}}\mathrm{C}^1_{\complement_S A_p} + \mathrm{C}^3_{A_{p^2}} \\
&= \frac{7}{6}p^6 - \frac{5}{2}p^5 + \frac{1}{2}p^3 + \frac{7}{3}p^2 + \frac{1}{2}p - 2.
\end{aligned} \tag{7}
$$

In this case, G_N is calculated from (7).

v) $N = p^2q$

The number of elements $|A_p|$ is $pq - 1$, the number of elements $|A_q|$ is $p^2 - 1$, the number of elements $|A_{p^2}|$ is $q - 1$, the number of elements $|A_{pq}|$ is $p - 1$, the number of elements $|\complement_S(A_p \cup A_q)|$ is $p^2q - p^2 - pq + p - 1$, $\mathscr{P}(A)_6$ represents the set $\{A_p, A_q, A_{p^2}, \complement_S(A_p \cup A_q)\}$, $\mathscr{P}(A)_7$ represents the set $\{A_p, A_{p^2}, A_{pq}, A_q\}$, $\mathscr{P}(A)_8$ represents the set $\{A_p, A_{pq}, \complement_S(A_p \cup A_q)\}$, $A_p - A_{p^2} - A_{pq}$ is abbreviated as A^*_p, $A_q - A_{pq}$ is abbreviated as A^*_q, $\complement_S A_p \cup A_q$ is abbreviated as $A^*_{\cup}$. By the permutation and combination formula, we have

$$
\begin{aligned}
G_N &= \mathrm{C}^3_S - \mathrm{C}^2_{A^*_{p^2}}(\mathrm{C}^1_{A^*_{\cup}} + \mathrm{C}^1_{A^*_p}) - \mathrm{C}^2_{A^*_q} \sum_{i\in\mathscr{P}(A)_8} \mathrm{C}^1_i \\
&\quad - \sum_{i\in\mathscr{P}(A)_6} \mathrm{C}^3_i - \mathrm{C}^2_{A^*_{\cup}} \sum_{i\in\mathscr{P}(A)_7} \mathrm{C}^1_i - \mathrm{C}^2_{A^*_p}\mathrm{C}^1_{A^*_{\cup}} - \mathrm{C}^2_{A^*_p}\mathrm{C}^1_{A^*_{p^2}} \\
&= \frac{5}{2}p^4q^2 - 3p^4q - 4p^3q^2 + p^4 + 3p^3q + \frac{3}{2}p^2q^2 - \frac{5}{6}p^3 \\
&\quad - 3p^2q + \frac{5}{2}p^2 + 3pq - \frac{2}{3}p - 2.
\end{aligned} \tag{8}
$$

In this case, G_N is calculated from (8). This completes the proof.

Based on the above construction methods, as long as the value of N is determined, based on the prime factorization formula of N, we can obtain the value of G_N corresponding to any N. Through Algorithm 1, we can obtain the set of all triplets (x, y, z) that satisfy (2).

In Fig. 1(a), the dark blue line represents the distribution of the number of triples (x, y, z) satisfying (2) within 500. It can be seen that with the increase of N, the number of triples (x, y, z) shows an overall increasing trend, and soon decreases to zero after reaching a peak. This is the result of N being a series of prime numbers. In theorem 2, according to the prime factorization of N, we discuss the calculation formula of the number of triples (x, y, z) in five cases. The red line represents the solution of three tuples satisfying (2) in Ref. [20] within 500. It is not difficult to find that the number of triples represented by the dark blue line is much higher than that represented by the red line, and the generated zero-divisor graph matrix has better randomness, which ensures that the generated zero-divisor graph has enough key-space, to improve the security of the honeywords scheme.

There are many ways to transform a ZDG matrix into a ZDG sequence. Without loss of generality, in this paper, we define four ways to construct a ZDG sequence (see Fig. 2). i) The first row, from left to right, the second row, from right to left, the third row, from left to right; ii) The first row, from right to left, the third row, from left to right, the second row, from right to left; iii) The first row, from right to left, the second row, from left to right, the third row, from right to left; iv) The second row, from left to right, the first row, from right to left, the third row, from left to right.

According to the generative method of the ZDG matrix coding sequences above, we have

$T^1_{code}(G_{100,10}) = 20104642152570154081828554487525755580869298999895869890 90$

or others representation sequences

$T^2_{code}(G_{100,10}) = 40127025152446102086929899989586989090818285544875257555 80$.

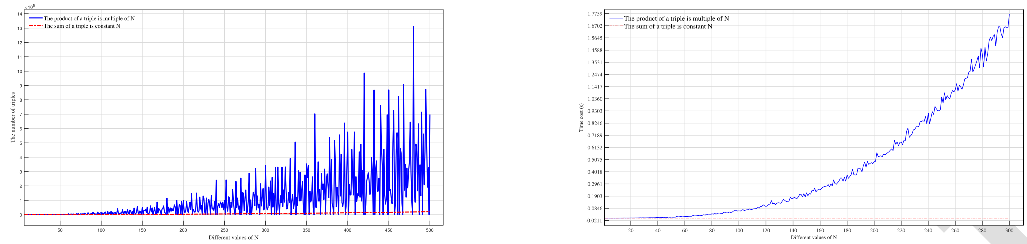
$T^3_{code}(G_{100,10}) = 40127025152446102080557525754854858281909098869598999892 86$.

$T^4_{code}(G_{100,10}) = 80557525754854858281401570251524461020869298999895869890 90$.

The ZDG matrix corresponding to the above the ZDG coding sequences is shown as follows

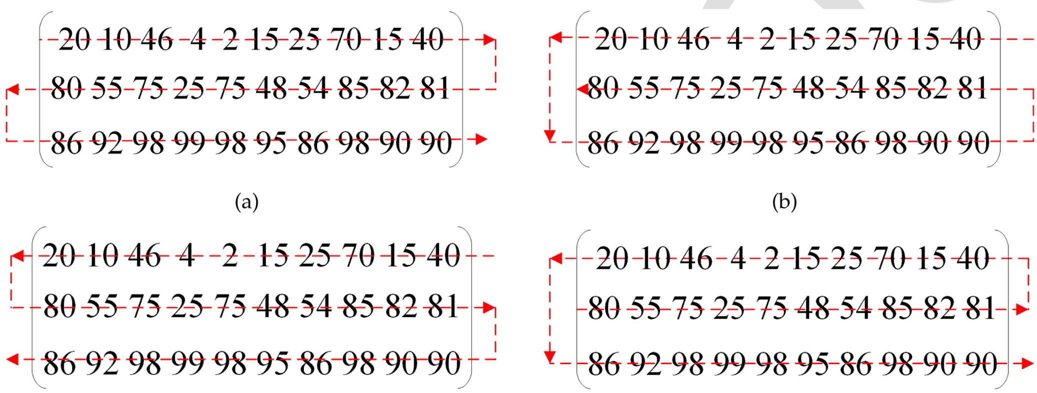
$$
T_{G_{100,10}} = \begin{pmatrix} 20 & 10 & 46 & 4 & 2 & 15 & 25 & 70 & 15 & 40 \\ 80 & 55 & 75 & 25 & 75 & 48 & 54 & 85 & 82 & 81 \\ 86 & 92 & 98 & 99 & 98 & 95 & 86 & 98 & 90 & 90 \end{pmatrix}. \tag{9}
$$

Through graph operations (intersection, union, difference, symmetric difference, etc.), zero-divisor graphs of different scales and structures can be obtained, since the triples generating



(a) (b)

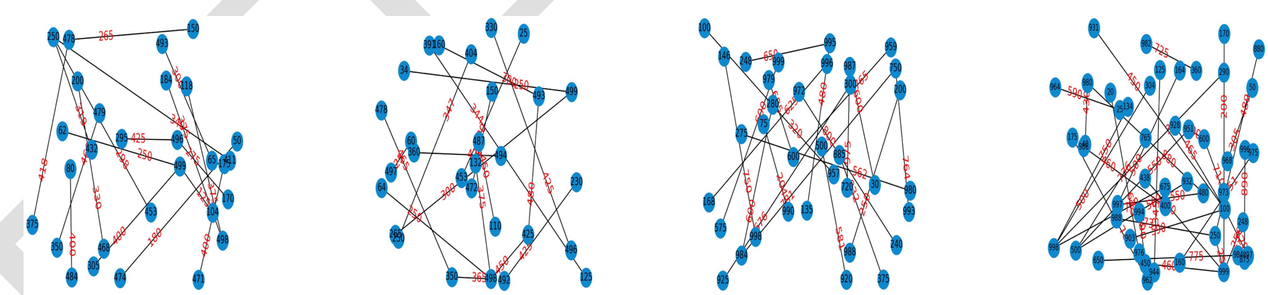
Fig. 1. (a) The distribution curve corresponding to the solution of $xyz \equiv 0 \; mod \, N$ and $x + y + z = N$, (b) The time cost comparison of the two algorithms in computing triples



(a) (b)

(c) (d)

Fig. 2. The rules for converting zero-divisor graph matrices into zero-divisor graph sequences



(a) (b) (c) (d)

Fig. 3. Zero-divisor graphs of different scales are obtained by set operation, (a) Zero-divisor graph $G_{500,18}$, (b) zero-divisor graph $G_{500,18}$ obtained by zero-divisor graph $G_{500,10} \oplus G_{500,8}$, (c) Zero-divisor graph $G_{1000,20}$, (d) zero-divisor graph $G_{1000,20} - G_{1000,15}$

the zero-divisor graph are randomly selected, even if the same integer N and the number $C_{numbers}$ of the triples are selected, the zero-divisor graph generated at different times is different. This undoubtedly increases the solution space of the zero-divisor graph (as shown in Fig. 4).

In Table II, the comparison of editing distances corresponding to the four rules shown in Fig. 2 is summarized. The four topological graph matrices contain 58 characters, and the editing distance between rule (a) and rule (b) in Fig. 2 is 41. Levenshtein distance is a string measure that calculates the degree of difference between two strings. This is the minimum number of times it takes to edit a single character (such as modify, insert, delete) when modifying from one string to another. Comparison of edit distances based on the four generation rules in Fig. 2. Four methods for transforming zero-divisor graph sequences are given in this paper, and the conversion methods are not limited to these four in the actual deployment of honeywords. The general evaluation principle is to ensure that the correlation between the elements in the converted zero-divisor graph matrix is the lowest, which increases the computational overhead of the adversary's cracking. These four rules show good independence.

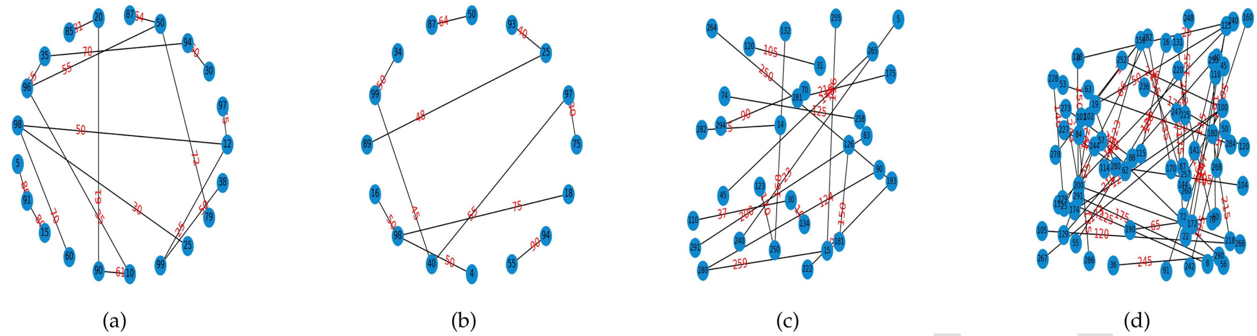


Fig. 4. Zero-divisor graphs of different scales are obtained by set operation, (a)zero-divisor graph $G_{100,18}$ obtained by zero-divisor graph $G_{100,20} - G_{100,10}$, (b) zero-divisor graph $G_{100,11}$ obtained by zero-divisor graph $G_{100,50} \cap G_{100,40}$, (c) Zero-divisor graph $G_{300,18}$, (d) zero-divisor graph $G_{300,50}$

TABLE II
EDITING DISTANCES COMPARISON OF FOUR RULES FOR GENERATING ZERO-DIVISOR GRAPH SEQUENCES

Edit distance	Rule (a)	Rule (b)	Rule (c)	Rule (d)
Rule (a)	0	71%	60%	47%
Rule (b)	71%	0	55%	69%
Rule (c)	60%	55%	0	60%
Rule (d)	47%	69%	60%	0

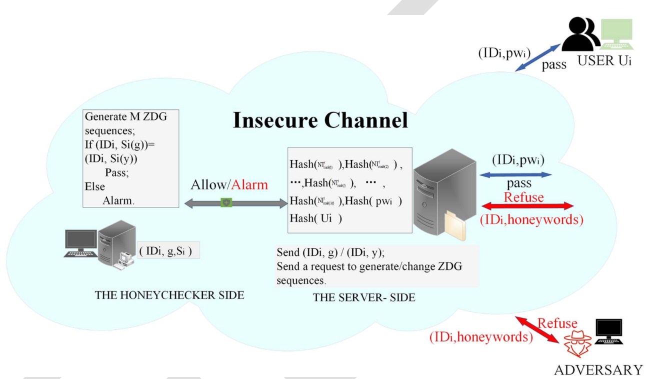


Fig. 5. Authentication system with honeywords

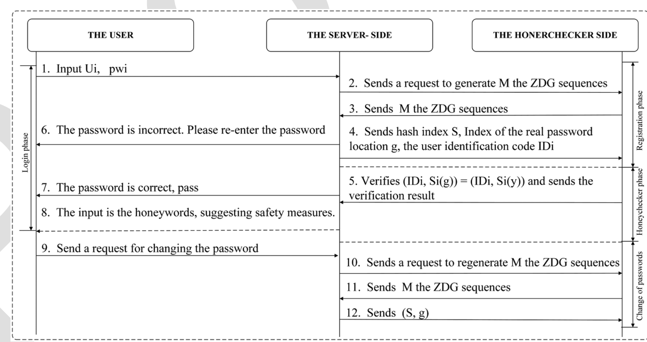


Fig. 6. Identity authentication protocol based on zero-divisor graph sequences

III. METHODOLOGY

Our password leakage detection is based on the honeywords model of the ZDG sequences, we illustrate the division of roles among the participants(see Fig. 5). We use the ZDG matrix to construct the ZDG sequences, and design a combination strategy of the ZDG sequences and usernames (password) to enhance the honeywords flatness. The combination rule of the ZDG sequences and username (password) is as follows. When the user enters the username and password, based on the Levenshtein distance algorithm, if the similarity of the string composed of the username and password is less than 50%, the username and the ZDG sequences are selected for combination. Otherwise, select the user password and ZDG sequences for combination. This method is called adaptive combination, which aims to increase the confusion of passwords.

During the process of generating the honeywords. For the sake of discussion, we specify that the ZDG sequence is aligned with the first digit of the ASCII code of the username (password). According to the length of the ASCII code of each symbol, the ZDG sequence is divided, and the value of the corresponding position of the two sequences is compared. If $P_{ZDG} > P_{ASCII}$, the value of the corresponding position of the ASCII code is increased by 1. If $P_{ZDG} < P_{ASCII}$, then the value of the ASCII position is reduced by 1. Otherwise, the value of the ASCII position remains unchanged. Until the value of the corresponding position of the ASCII code is compared. Finally, a new sequence is formed. Without loss of generality, we stipulate that the length of the zero-divisor graph sequence is greater than the length of the ASCII code. We recommend that the difference in length between the two sequences be no less than 10. There are at least 10 such alignments position. (see Table III). The role division of each system is shown in Fig. 6, the dotted line represents the registration phase, login phase, honeychecker phase, and change of passwords respectively. Dashed lines with arrows indicate that security measures are taken, when the system detects that the attacker (user) is logging in by the honeywords, but the attacker (user) cannot feel this change in the system. The specific scheme of detecting password leakage based on honeywords is given as follows.

A. Initialization Phase

Honeywords authentication system uses dual server structure, the server-side and honeychecker-side, and they only

TABLE III
HONEYWORDS GENERATION RULES BASED ON THE ZDG SEQUENCES

ASCII code of the username (password)	The ZDG sequences	New sequences	Hash value
AUP_i	$T^i_{code(1)}$	$NT^i_{code(1)}$	Hash($NT^i_{code(1)}$):H_{i1}
AUP_i	$T^i_{code(2)}$	$NT^i_{code(2)}$	Hash($NT^i_{code(2)}$):H_{i2}
...	...	...	...
AUP_i	$T^i_{code(t)}$	$NT^i_{code(t)}$	Hash($NT^i_{code(t)}$):H_{it}
...	...	...	...
AUP_i	$T^i_{code(M)}$	$NT^i_{code(M)}$	Hash($NT^i_{code(M)}$):H_{iM}

*t is the length of ASCII code, and M is the number of the ZDG sequences, where $1/3t < M$.

do simple communication, the user-side and the server-side communication process is the same as the previous design, the user does not have additional overhead, so the user will not perceive the existence of such a server.

The server-side should store U_i and the identifier ID_i of U_i, the hash value $H(pw_i)$ of the user's real password, the hash value H_i corresponding to the new zero-divisor graph sequences NT^i_{code} ($i \in [1, n]$), which are the combination of the username's ASCII codes and zero-divisor graph sequences. The honeychecker-side contains the index value k of the user‘s real password location, the identifier ID_i of U_i, the sequence index value set S_i of hash files H_{i1}, H_{i2}, ..., H_{iM}, $H(pw_i)$.

In this paper, to increase the set space of the ZDG, we get the ZDG and the ZDG matrices of different scales and forms by graph operations (intersection, union, difference, symmetric difference, etc. See Fig. 4). The set B of triples (x, y, z) is given by Algorithm 1, the G_N is given by (2), and the ZDG $G_{N,T}$ is given by Algorithm 2.

The honeychecker is an auxiliary server, which is responsible for generating the ZDG and the ZDG sequences. The generation methods are given as follows.

- Select randomly the values of N and T.
- Obtain the ZDG matrices by using the steps of Algorithm 2.
- Select M $(=1/3L_{AUP_i} + 2)$ from the generated ZDG matrices.
- Construct the ZDG sequences, according to the generation rules in Fig. 2.

Now, we introduce how to generate the ZDG and the ZDG matrices by graph computation, and obtain the unique sequence of the ZDG according to the four generation rules in Fig. 3, without loss of generality, we use the first generation rule (a) in subsequent discussions. According to Theorem 1, the adjacency matrix corresponding to the isomorphic graph is the same, so it is difficult for the adversary to get the unique ZDG through the adjacency matrix. By observing the ZDG matrix (10), we notice that when the C_{number} is very small ($C_{number} = 11$), the sequences $T_{code}(G_{500,11})$ of the ZDG matrix corresponding to the ZDG reaches 88 characters, which can be combined with the ASCII code of the username to hide the statistical characteristics of the username. The following is a new matrix for different matrices through splicing operations.

Algorithm 1: Generate All Triples That Satisfy the Equation.

Input: The target values N, $(x, y, z) \in N_{set}$.
  ▷ The conditions set N_{set} is the input sequences.
Output: B. ▷ B is the target set.

Initialization:
candidates ← list(range(2, N))
if $length <= 3$ **then**
  return$B$ ▷ In this case, B is an empty set.
end if
def backtrack(i, array, $list_{array}$) ▷ The i is the number of elements traversed into the candidates, the array is the product of currently traversing array elements, the $list_{array}$ is the array currently traversed.
if array% N==0 and len($list_{array}$)==3 **then**
  $B \leftarrow B \cup list_{array}$
else if len($list_{array}$)==3 **then**
  return
end if
  for $j = i$ to len(candiates) **do**
    $j \leftarrow j + 1$, array← array*candidates[j], $list_{array} \leftarrow list_{array} \cup$ candidates[j]
  backtrack(0, 1, [])
  return
end for
return B.

$$T_{G_{500,11}} = \begin{pmatrix} 10 & 57 & 70 & 350 & 25 & 14 & 120 & 75 & 255 & 4 & 177 \\ 246 & 150 & 75 & 425 & 135 & 250 & 151 & 372 & 400 & 6 & 224 \\ 420 & 200 & 246 & 438 & 188 & 403 & 275 & 455 & 442 & 125 & 375 \end{pmatrix},$$

$$T_{G_{500,6}} = \begin{pmatrix} 10 & 57 & 70 & 350 & 25 & 14 \\ 246 & 150 & 75 & 425 & 135 & 250 \\ 420 & 200 & 246 & 438 & 188 & 403 \end{pmatrix},$$

$$T_{G_{500,5}} = \begin{pmatrix} 120 & 75 & 255 & 4 & 177 \\ 151 & 3720 & 400 & 6 & 224 \\ 275 & 455 & 442 & 125 & 375 \end{pmatrix} \quad (10)$$

According to the ZDG sequences generation method defined earlier, we get

Algorithm 2: Generating the Zero-Divisor Graph $G_{N,T}$.

Input: Select two subsets B_1 and B_2 from the set B generated by Algorithm 1

Output: Zero-divisor graph $G_{N,T}$.

1: Initialization:
Set the operations (intersection, union and difference, symmetric difference, etc.) on sets B_1 and B_2 to generate the vertex set $V = \{v_1, v_2, \ldots, v_n\}$.
2: **for** $i = 1$ to $|V|$ **do**
3: $v_i \leftarrow V_{\min}$
4: **for** $j = 1$ to $|V - V_i|$ **do**
5: **if** weight$(v_i v_{j_k}) \neq$ weight$(v_i v_{j_s})$ **then**
6: Generate the edge set:
$E_i \leftarrow \{e_i e_{j_1}, \ldots, e_i e_{j_m}\}$ $\triangleright$ m is the number of vertices connected to v_i, V_i is the set of vertices with the smallest labeling.
7: **else**
8: $v_j \leftarrow \min\{labeling(v_{j_k}), labeling(v_{j_s})\}$,
9: **return** Step 6.
10: **end if**
11: **end for**
12: **end for**
13: **return** $G_{N,T}$.

$T_{code}(G_{500,11})$=1057703502514120752554177246150754251352501513724006224420200246438188403275455442125375.

B. Registration Phase

In this phase, the server prepares the registration service for the user. The user sends the username U_i and password pw_i to the server-side. The server requires the following operation. First, the server-side generates the index ID_i for the username U_i, second, the server-side converts the username (password) string into ASCII code, third, the server-side sends the requests of the ZDG sequences to the honeychecker-side, finally, the server-side decomposes the ASCII code with ZDG sequence into a new sequence, to hash the sequence, and to store the hash file. Generate an index g of the sequences code that contains the user's real password. Send the index ID_i of the username U_i and the index g of the user password pw_i to the honeychecker side.

During the registration phase, the communication between the server and the honeychecker side only should achieve the following goals. i) The server-side sends the request of the ZDG sequences; ii) The server-side sends the index value ID_i of the username and the sequence index value g of the real password; iii) The honeychecker side sends the ZDG sequences. There is only simple communication between the server and the honeychecker side.

C. Login phase

When the server-side receives the username and password submitted by the user. First, the server-side determines whether the username exists, second, the server-side judges whether the user's password matches the username stored. To achieve these, the server-side checks the username index file. the server-side and the honeychecker side performs the following operations. The server-side hashes the password submitted by the user and compares it with the hash values $H^i = \{H_{i1}, H_{i2}, \ldots, H_{iM}, H(pw_i)\}$ stored in the system.

- If it is inconsistent with the hash value stored in the system, the user will be prompted that the password is wrong and needs to be re-entered.
- If it is consistent with the hash value H^i stored in the system, the server-side sends the index value y where the hash value is located to the honeychecker side.

D. Honeychecker Phase

The honeychecker side is an auxiliary server, which only communicates with the server of the service provider. The honeychecker side stores the ZDG sequences, the user index ID_i, and the index k corresponding to the user's correct password, the index value set $S(=\{S_1, S_2, \ldots, S_n\})$ of hash files $H = \{H^1, H^2, \ldots, H^j, \ldots, H^n\}$. The honeychecker side communicates with the server-side through a secure channel. The role of the honeychecker is consistent with that described in Ref. [7]. The following information is exchanged between the honeychecker side and the server-side. In our scheme, T^i_{code} represents the selected ith ZDG sequences of the ZDG, and S_I represents the ZDG sequences index. The following are the operations to be run by the honeychecker side.

- Send the ZDG sequences $\left(T^i_{code(1)}, T^i_{code(2)}, \ldots, T^i_{code(M)}\right)$ to the service-side over a secure channel.
- Check: ID_i, g, S_i, y
  Verifying whether $(ID_i, S_i(g))$ and $(ID_i, S_i(y))$ are the same, $S_i(g)$ represents the index value of the user's real password. If the verification results are inconsistent, the honeychecker side will remind the server-side that what the user just entered is honeywords, and the server will take corresponding security measures.

The verification side works intermittently. Only when the server-side sends the request, the verification side can carry out the necessary communication. The verification side only knows the index of the username, but does not know the user's password or the hash file of the user's password.

E. Change of Passwords

When the server-side receives the user's request to modify the password. First, the server should confirm the legitimacy of the user, second, the server sends the request to modify the ZDG sequences to the honeychecker side. The information interaction between the honeychecker side and the server-side is given as follows:

- Regenerate M zero-divisor graph matrices and the corresponding zero-divisor graph sequences by Algorithm 2.
- Send the ZDG sequences $\left(T^i_{code(1)}, T^i_{code(2)}, \ldots, T^i_{code(M)}\right)$ to the service-side over a secure channel.

TABLE IV
PROBABILITY OF OBTAINING THE CORRECT ZDG SEQUENCES FOR DIFFERENT N, C_{number} AND M

N	C_{number}	M	Pr	Exponent
30	8	10	$0.17069 \times 10^{-2091}$	2^{-6949}
40	8	10	$0.12720 \times 10^{-4708}$	2^{-15643}
50	6	12	$0.71766 \times 10^{-5266}$	2^{-17494}
60	8	20	$0.49801 \times 10^{-21745}$	2^{-72237}
60	10	8	$0.44821 \times 10^{-21743}$	2^{-72230}
80	4	12	$0.24718 \times 10^{-38484}$	$2^{-127844}$
100	6	10	$0.33863 \times 10^{-56076}$	$2^{-186283}$
100	8	6	$0.18963 \times 10^{-56074}$	$2^{-186277}$

- Update the hash files $H_{i1}, \ldots, H_{it}, \ldots, H_{iM}, H(pw_i)$ of the user U_i stored on the server-side.
- Update information (ID_i, g, S_i) stored on the honey-checker side.

IV. Security Analysis

For some possible attack scenarios, we analyze the rationality and security of the proposed scheme. We assume that the adversary can crack many hash files stored on the server-side. When the times that an adversary login through honeywords exceeds the threshold allowed by the system, the verification side will give an alarm, and the server-side will take security measures. To reduce the threshold of triggering the alarm in this case, the corresponding security policy against the DoS attack is designed.

A. Brute-Force Attack

We assume that the adversary can reverse the hash file stored on the server-side. If N, C_{number}, M, and construction rules of the ZDG sequences are leaked, the adversary will analyze all possible ZDG sequences by the brute-force attack. If M zero-divisor graph sequences can be obtained, the adversary will get easily the user's password. A detailed analysis is given as follows.

According to the generating method of the ZDG sequences, select C_{number} from triple set A to form the ZDG matrix. With the different arrangement of C_{number} triples, form $C_{number}!$ different ZDG matrices. Select M from these matrices to generate the ZDG sequences. We can analyze the success probability of the adversary obtaining the ZDG sequences. The probability of the adversary acquiring M zero-divisor graph sequences is

$$Pr = \frac{1}{\mathrm{A}_{G_N}^{C_{number}} - M + 1}. \tag{11}$$

When $N = 60$, $C_{number} = 9$, $M = 10$, the probability of getting correct ZDG sequences is $Pr = 0.44821 \times 10^{-21744}$, which is approximately equal to 2^{-72234}. Therefore, it is reasonable for our zero-divisor graph sequences generation method, which can provide a suitable key-space for the subsequent generation of the honeywords (see Table IV).

B. Dictionary Attack

In order to improve the success rate of cracking the ZDG sequences, the adversary may combine the selected ciphers into a specific dictionary, construct the set of the ZDG sequences according to the rules, and generate honeywords through the combination of the elements in the two sets. The following is a detailed analysis.

The security of our scheme is related to the difficulty of calculating the ZDG and the ZDG sequences. Therefore, the adversary should construct an appropriate scale ZDG sequence set. First, according to the generation rules of the ZDG, the adversary should select C_{number} elements from the triples set. The C_{number} elements have $C_{number}!$ permutations, that is to say, $C_{number}!$ matrices can be formed. Second, the adversary selects M matrices from these matrices to construct the ZDG sequences. Finally, in the operation stage of M ZDG sequences and ASCII code, with the different of M ZDG sequences, the combination results of AUP_i and M ZDG sequences are also different, and the hash value is also different. The set size of the ZDG sequences that the adversary should construct is

$$Size = \mathrm{C}_{\mathrm{A}_{G_N}^{C_{number}}}^{M} = \frac{\left(\frac{G_N!}{C_{number}!}\right)!}{M!\left(\frac{G_N!}{C_{number}!} - M\right)!}$$

From the above analysis, we can see that our scheme can provide better security. The number of the ZDG sequences to be selected varies with the length of the AUP_i, which will undoubtedly increase the computational overhead. In the practical application process, according to the needs of the system, we need to make a trade-off between limiting the length of the username and reducing the computational overhead.

C. Denial-of-Service Attack

In this case, the adversary does not crack the password file, but through a certain way to get the ZDG sequences generation method, he can generate all the possible honeywords of the user password, and he can trigger the early warning of the honeychecker side through the honeywords login. We assume that the adversary obtained the username information. As long as he finds one of the M zero-divisor graph sequences, and can construct a honeyword to launch the DoS attack. The success rate that the opponent obtaining the effective honeywords is

$$P_h = \frac{N_{re} \times M}{\mathrm{A}_{G_N}^{C_{number}}}, \tag{12}$$

N_{re} represents the number of registered users, M represents the number of honeywords assigned for each user. Without

TABLE V
HONEYWORD GENERATION RULES BASED ON THE ZDG SEQUENCES

ASCII code AUP_i of the username (password)	Zero-divisor graph sequence T^i_{code}	New sequences NT^i_{code}	Honeyword
	26101062425351540404541284028	7311010311156505249	Ingo8241
	...	...	...
John9352 **74111104** **11057515350**	91215231725302023253225354044	7511210511156505249	Kpio8241
	...	...	...
	16412152630222520354535482545	7311210511156505249	Ipio8241
	40121524047362030434839235554	11710611011012249505163	ujnnz123?
	...	...	...
timmy234@ **116105109** **10912150515264**	28445402225444835355751515454	11710610811012249505363	ujlnz125?
	...	...	...
	27101231621241912555631452555	11710411011012249525363	uhnnz145?

losing generality, we consider that the adversary has $N_{re} = 10^6$ username-password pairs, he (she) may use these accounts to carry out DoS attacks. Assuming the threshold for unsuccessful login is T_l, the probability for an attacker guesses v honeywords after guessing r times is $C^v_r p^v (1-p)^{r-v}$. For example, when $T_l = 5, v = 500, N = 15, C_{number} = 20, M = 14, P_h = 0.55623 \times 10^{-78}$. Since the adversary has 10^6 accounts, he can try 5×10^6 times. The probability that an attacker guesses 1000 honeywords after guessing 5×10^6 times is 0.96×10^{-19349}. In this case, the ability of the adversary to trigger the honeychecker to issue an early warning is effectively reduced.

V. COMPARISON OF HONEYWORDS METHODS

In this section, we will discuss the performance of the following schemes from three aspects, i.e., the honeywords flatness, DoS attack resistance, and the cost of memory.

A. Honeywords Flatness

According to the analysis in Ref. [9], in Juels et al.'s scheme, the success rate that the adversary guessing is $29\% \sim 33\%$, which is higher than $1/k$. In Erguler's method [8], the flatness of the adversary guessing success is $1/k$ for registered users. In the method of Akshima et al. [37], Evolving-Password Model (EPM) and Append-Secret Model (ASM) schemes have a good performance in flatness, up to $1/k$, but User-Profile Model (UPM) involves the user's personal information, and the flatness is higher than $1/k$. In the scheme proposed by Guo et al. [13], since there is no direct correspondence between username and password, the flatness of the adversary guessing success is $1/k \sim 1/N$. In our scheme, in the registration phase, the process of handling username is given as follows. i) The username (password) is transformed into the corresponding ASCII code. ii) The zero-divisor graph sequence is aligned with the first digit of the ASCII code of the username (password). According to the length of the ASCII code of each symbol, the zero-divisor graph sequence is divided, and the value of the corresponding position of the two sequences is compared. iii) If $P_{ZDG} > P_{ASCII}$, the value of the corresponding position of the ASCII code is increased by 1. If $P_{ZDG} < P_{ASCII}$, then the value of the ASCII position is reduced by 1. Otherwise, the value of the ASCII position remains unchanged. Until the value of the corresponding position of the ASCII code is compared. iv) A new sequence is formed. From (11), the probability that the adversary obtaining the correct zero-divisor graph sequences through off-line guessing is very small. Since the triples are randomly selected from the set N_{set}, it can be regarded as equal probability.

Next, let us illustrate with an example. The username is *John9352 (smith1024)*, the U_i's password is *john8342 (timmy234@)*, ASCII code of the username *John9352* is *74111104110575153 50*, and ASCII code of the password *timmy234@* is *116105109109121505 15264*. Without losing generality, we choose $1/3L_{AUP_i} + 34$ zero-divisor graph sequences for combining with ASCII code (see shown in Table V), and the length of the chosen ZDG sequences is the same, which is 29, please see Table III for specific operation rules. Then the success rate for the opponent guessing the password is approximately $1/2^{L_{U/P}} = 0.0039$.

B. DoS Resistance

In terms of responding to DoS attacks, we evaluated the performance of the following scheme. In the strategy method proposed in Ref. [7], the chatting-with-tweaking-model performs poorly when it suffers DoS attacks. Because the honeywords have a small generating space, the password is easy to guess out. For example, when $t = 2$ (t is the number of the password tails to be modified), the success rate that the opponent guessing an effective honeywords is $(k-1)/99$. Whereas, the chatting-with a-password-model performs well when it suffers DoS attacks, since it constructs the honeywords based on the probability model. In the scheme proposed in Ref. [8], it generates the honeywords based on the passwords of other $k-1$ users, which performs well against the opponent guessing attacks. In Guo et al.'s method [13], the honeypot mechanism is based on a mix of real and fake accounts. The adversary should build fake accounts set. When he (she) logins with a certain number of

TABLE VI
COMPARISON OF THE HONEYWORDS GENERATOR MODELS

Method	DoS Resistance	Flatness	Storage Cost
Juels [7]	it depends	more than $1/k$	$2AL_U + kAL_H + AL_k$
Erguler [8]	strong	$1/k$	$2(A+T)L_U + (k(A+T)+N)L_I + NL_H + (A+T)L_k$
EPM [11]	strong	$1/k$	$2AL_U + kAL_H + AL_k$
UPM [11]	moderate	$\cong 1/k$	$2AL_U + kAL_H + AL_k$
APM [11]	strong	$1/k$	$2AL_U + kAL_H + AL_k$
Superword [13]	strong	$1/N$	$(A+T)L_U + 2(A+T+N)L_I + NL_H$
Tian [20]	strong	$1/M$	$A(L_U + 2L_{ID} + ML_H + ML_{TS} + L_{S_I} + L_j)$
Our model	strong	$1/M$	$A(L_U + 2L_{ID} + ML_H + L_S + 2L_g)$

fake user names, the honeychecker will detect and send out an alert. In the user-profile-model scheme [37], since the prediction of honeywords, the system is vulnerable to DoS attacks. In our scheme, according to the combination method of ASCII code AUP_i and ZDG sequences T^i_{code} in Table III, we can see that the combination of the sequences of ASCII code and the ZDG are different, and different results are obtained. When the adversary logs in with the honeywords, the server should set the threshold of triggering security measures. Combined with the analysis in Section IV-C, our scheme provides resistance against the DoS attack.

C. Storage Overhead

In this section, in terms of storage overhead in the secondary server, we evaluate the performance of the following scheme. Suppose there are A registered real user accounts stored in the system, and let L_U, L_H, L_k represent the binary lengths of username, hash sequence value and k. In the schemes proposed by Juels et al. [7] and Akshima et al. [11], they occupy the cost of $AL_U + kAL_H$ in the server-side, and the cost of $AL_U + AL_k$ in the honeychecker side. In Erguler's scheme [8], it requires the cost of $(A+T)L_U + k(A+T)L_I + NL_H + NL_I$ in the server-side and $(A+T)L_U + (A+T)L_k$ in the honeychecker. In the scheme proposed by Guo et al. [13], it occupies the cost of $(A+T)L_U + (A+T)L_I + NL_H + NL_I$ in the server-side and $(A+T)L_I + NL_I$ in the honeychecker, where L_I represents the lengths of an index, N represents the number of listed hashed passwords, and T represents the honeypots in bytes. In the scheme of Ref. [20], it requires the cost of $A(L_U + L_{ID} + ML_H)$ in the server-side and $A(L_{TS} + L_{ID} + L_{SI} + L_j)$ in the honeychecker. In our honeywords scheme, it occupies the storage cost $A(L_U + L_{ID} + L_g + (M+1)L_H)$ in the server-side and $A(L_{ID} + L_g + L_S)$ in the honeychecker, L_g represents the binary lengths of the index of correct passwords in hash sequences, L_S represents the binary lengths of the hash sequences index. Table VI summarizes the comparison of flatness, DoS resistance and storage overhead of different schemes. The storage overhead of our scheme is smaller than that of Tian et al.'s scheme [20], the main adjustment is that the honeychecker side does not need to store the zero-divisor graph sequences. When the server-side sends the request, the honeychecker side can regenerate the zero-divisor graph sequences by algorithm 1 and algorithm 2. This part of the computational overhead is acceptable. The main difference between these two schemes is in computing overhead. For example, the user size is 10^6, the username is 12 characters, the username ID_i is 12 characters, the SHA256 hash sequence takes 32 bytes, the user password is 8 characters, the hash sequence index is S ($S \in [1, M+1]$ digits), the index of the user's real password is g, it takes 2 characters. and the storage cost of the system is approximately 3.65GB. We use Intel (R) core (TM) i7-6700 processor, and the memory is 8192MB. When we run Algorithms 1 and 2 to generate the ZDG, the time is about 0.1547336s for generating the zero-divisor graph matrix and zero-divisor graph $G_{100,200}$, at this time, the scale of the triples satisfying (2) is 14988. Comparing our algorithm with the algorithm in Ref. [20], it is not difficult to find that the difference between them lies in the cost of computing triples. Fig. 1(b) shows the time comparison of these two algorithms in calculating the triples within 300. Our scheme generates more triples, although the traversal time of triples becomes longer, the total generation time is measured in seconds. Considering that our scheme gives more ZDG, it provides more space for constructing zero-divisor graph sequences. In other words, our scheme can provide higher security.

VI. CONCLUSION

It is the most effective way to solve the current user information leakage for password leakage detection technology, the honeywords scheme is one of the most potential solutions. The honeywords scheme only needs to make appropriate modifications to the existing server-side, and there is no additional authentication overhead for registered users. However, there are two core problems in the design of the honeywords scheme: one is the honeywords flatness, which is the premise to ensure the effectiveness of the honeywords scheme; the other is to resist DoS attack. If the adversary has a high success rate in guessing the honeywords, the server-side may perform a full password reset in response to the honeywords attack. This can seriously affect the normal access of the server-side. In this paper, we propose the concept of the zero-divisor graph and give an algorithm to solve the 3-tuples congruence equation. We use graph operation rules to generate numerous zero-divisor graphs. To ensure that the zero-divisor graph sequences have enough generated space, for example, $N = 400$, $G_N = 576008$, $C_{number} = 10$, and the scale of the zero-divisor graph is set to be $10^{3067892}$, we propose a honeywords generation scheme based on the zero-divisor graph. It should be noted that, since the username (password) length selected by each user is different, the corresponding ASCII code length is different, the number of the ZDG sequences selected is also different, and the final number of honeywords owned by each user is also different. For adequate security, the username should not be too short. Through security analysis and comparison with other honeywords schemes, our scheme has better performance.

REFERENCES

[1] C. Osborne, "Yahoo data breach victims have less than a week to join million-dollar class action settlement," [Online]. Available: https://portswigger.net/daily-swig/yahoo-data-breach-victims-have-less-than-a-week-to-join-million-dollar-class-action-settlement

[2] K. W. Alistair Barr and Bloomberg, "Facebook data on 533 million users reemerges online for free," [Online]. Available: https://fortune.com/2021/04/03/facebook-data-users-reemerges-online-for-free/

[3] K. Singh, "Leaked Android 12 privacy dashboard shows Google is getting serious about protecting your data," [Online]. Available: https://www.androidpolice.com/2021/05/18/android-12-privacy-dashboard-leak-shows-google-is-getting-serious-about-protecting-your-data/

[4] W. Sheng, "Bilibili source code containing user names and passwords leaked on GitHub," [Online]. Available: https://technode.com/2019/04/23/bilibili-source-code-leaked-on-github-containing-usernames-and-passwords/ Q4

[5] M. Dürmuth and T. Kranz, "On password guessing with (GPU)s and FPGAs," in *Proc. Int. Conf. Passwords*, 2015, pp. 19–38.

[6] M. Adeptus, "Hashdumps and passwords," May 2014. [Online]. Available: http://www.adeptus-mechanicus.com/codex/hashpass/hashpass.php

[7] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 145–160.

[8] I. Erguler, "Achieving flatness: Selecting the honeywords from existing user passwords," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 284–295, Mar./Apr. 2016.

[9] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018. Q5

[10] N. Chakraborty and S. Mondal, "On designing a modified-UI based honeyword generation approach for overcoming the existing limitations," *Comput. Secur.*, vol. 66, pp. 155–168, 2017.

[11] Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, "Generation of secure and reliable honeywords, preventing false detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 5, pp. 757–769, Sep./Oct. 2019. Q6

[12] K. C. Wang and M. K. Reiter, "How to end password reuse on the web," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2019.

[13] Y. Guo, Z. Zhang, and Y. Guo, "Superword: A honeyword system for achieving higher security goals," *Comput. Secur.*, vol. 103, 2019, Art. no. 101689.

[14] J. Camenisch, A. Lehmann, and G. Neven, "Optimal distributed password verification," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 182–194.

[15] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE Symp. Secur. Privacy*, Los Alamitos, CA, USA, 2022, pp. 489–506.

[16] A. Dionysiou, V. Vassiliades, and E. Athanasopoulos, "HoneyGen: Generating honeywords using representation learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2021, pp. 265–279.

[17] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2014, pp. 293–310.

[18] 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.

[19] F. Harary and E. M. Palmer,*Graphical Enumerations*. New York, NY, USA: Academic, 1973.

[20] Y. Tian, L. Li, H. Peng, and Y. Yang, "Achieving flatness: Graph labeling can generate graphical honeywords," *Comput. Secur.*, vol. 104, 2021, Art. no. 102212.

[21] H. Wang, J. Xu, M. Ma, and H. Zhang, "A new type of graphical passwords based on odd-elegant labelled graphs," *Secur. Commun. Netw.*, vol. 2018, 2018, Art. no. 9482345.

[22] L.-H. Lim, "Hodge Laplacians on graphs," *SIAM Rev.*, vol. 62, no. 3, pp. 685–715, 2020.

[23] L. Vaš, "Graded cancellation properties of graded rings and graded unit-regular Leavitt path algebras," *Algebras Representation Theory*, vol. 24, pp. 625–649, 2020.

[24] S. Dahlberg and S. van Willigenburg, "Chromatic symmetric functions in noncommuting variables revisited," *Adv. Appl. Math.*, vol. 112, 2020, Art. no. 101942.

[25] T. Nam and N. Phuc, "The structure of Leavitt path algebras and the invariant basis number property," *J. Pure Appl. Algebra*, vol. 223, no. 11, pp. 4827–4856, 2019.

[26] J. F. Alm and D. A. Andrews, "A reduced upper bound for an edge-coloring problem from relation algebra," *Algebra Universalis*, vol. 80, no. 2, pp. 1–11, 2019.

[27] A. Conca and V. Welker, "Lovász–Saks–Schrijver ideals and coordinate sections of determinantal varieties," *Algebra Number Theory*, vol. 13, no. 2, pp. 455–484, 2019.

[28] S. Akbari, D. Kiani, and F. Ramezani, "Commuting graphs of group algebras," *Commun. Algebra*, vol. 38, no. 9, pp. 3532–3538, 2010.

[29] I. Beck, "Coloring of commutative rings," *J. Algebra*, vol. 116, no. 1, pp. 208–226, 1988.

[30] P. S. Livingston, "Structure in zero-divisor graphs of commutative rings," 1997. Q7

[31] D. F. Anderson, R. Levy, and J. Shapiro, "Zero-divisor graphs, von Neumann regular rings, and Boolean algebras," *J. Pure Appl. Algebra*, vol. 180, no. 3, pp. 221–241, 2003.

[32] S. P. Redmond, "An ideal-based zero-divisor graph of a commutative ring," *Commun. Algebra*, vol. 31, no. 9, pp. 4425–4443, 2003.

[33] G. Aalipour and S. Akbari, "On the Cayley graph of a commutative ring with respect to its zero-divisors," *Commun. Algebra*, vol. 44, no. 4, pp. 1443–1459, 2016.

[34] G. Chartrand, *Introduction to Graph Theory*. New York, NY, USA: Tata McGraw-Hill Education, 2006.

[35] B. Yao et al., "Topological coding and topological matrices toward network overall security," 2019, *arXiv: 1909.01587*.

[36] D. B. West et al., *Introduction to Graph Theory*, vol. 2. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[37] K. Akshaya and S. Dhanabal, "Achieving flatness from non-realistic honeywords," in *Proc. Int. Conf. Innovations Inf., Embedded Commun. Syst.*, 2017, pp. 1–3.

Yanzhao Tian received the MS degree in applied mathematics from the Chengdu University of Information Technology in 2012, and the PhD degree in cyberspace security from the Beijing University of Posts and Telecommunications, Beijing, China, in 2022. He is currently a lecturer with the School of Cyberspace Security, Hainan University. His research interests include cryptography, graph theory and graphical passwords.

Lixiang Li received the MS degree in circuit and system from Yanshan University, Qinhuangdao, China, in 2003, and the PhD degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 2006. She is the National Excellent Doctoral Theses winner, the New Century Excellent Talents in University, Hong Kong Scholar Award winner, Beijing Higher Education Program for Young Talents winner. She visiting Potsdam Institute for Climate Impact Research, Germany, in 2011. She engaged in research of compressive sensing, swarm intelligence and network security. She is currently a professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, China. She is the co-author of more than 100 papers.

Haipeng Peng received the MS degree in system engineering from the Shenyang University of Technology, Shenyang, China, in 2006, and the PhD degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010. He is currently a professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, China. His research interests include compressive sensing, information security, network security and complex networks. He is the co-author of 60 papers.

Ding Wang received the PhD degree in information security with Peking University, in 2017. He is currently supported by the "Boya Postdoctoral Fellowship" in Peking University, China. As the first author, he has published more than 40 papers at venues like ACM CCS, Usenix Security, NDSS, IEEE DSN, ESORICS, ACM ASIACCS, *ACM Transactions on Cyber-Physical Systems*, *IEEE Transactions on Dependable and Secure Computing* and *IEEE Transactions on Information Forensics and Security*. Seven of them are recognized as "ESI highly cited papers". His PhD thesis receives the "ACM China Doctoral Dissertation Award" and "China Computer Federation (CCF) Outstanding Doctoral Dissertation Award". He has been involved in the community as a TPC member for more than 40 international conferences. His research interests focus on password, authentication and provable security.

Yixian Yang received the MS degree in applied mathematics and the PhD degree in electronics and communication systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 1986 and 1988, respectively. He is the managing director of Information Security Center, Beijing University of Posts and Telecommunications, Beijing, China. TheYangtze River Scholar Program professor, National Outstanding Youth Fund winner, the National Teaching Masters. Major in coding and cryptography, information and network security, signal and information processing; having published more than 300 high-level papers and 20 monographs.