博士論文

Doctoral Thesis

The Computational Complexity in Various Settings of Cryptographic Primitives
(暗号プリミティブにおける様々な状況下での計算困難性)

Ying Hwei Ming Jason

イン ホェイ ミン ジェイソン

Acknowledgements

I wish to express my heartfelt thanks to my academic supervisor, Professor Noboru Kunihiro for his support and advice throughout my doctoral course. I will also like to thank Professor Phong Nguyen for some insightful discussions. In addition, I wish to extend my appreciation to Professor Hirosuke Yamamoto, Professor Tsuyoshi Takagi, Professor Yasutoshi Makino and Professor Junya Honda for providing valuable comments on a preliminary version of this thesis. I am also grateful to Sakakibara-san for helping me in administrative matters. Finally, I will like to thank every past and present members of the laboratory whom I have had the pleasure of knowing and interacting with.

Abstract

Discrete logarithms arise in many aspects of cryptography. The hardness of the discrete logarithm problem is central in many cryptographic schemes; for instance in signatures, key exchange protocols and encryption schemes.

The first main contribution of this thesis examines the generic hardness of the generalized multiple discrete logarithm problem, where the solver has to solve k out of n instances for various settings of the discrete logarithm problem. For generic k and n, we introduce two techniques to establish the lower bounds for this computational complexity. One method can be shown to achieve asymptotically tight bounds for small inputs in the classical setting. The other method achieves bounds for larger inputs as well as being able to adapt for applications in other discrete logarithm settings. In the latter, we obtain the generalized lower bounds by applying partitions of n and furthermore show that our chosen method of partition achieves the best bounds. This work can be regarded as a generalization and extension on the hardness of the multiple discrete logarithm problem analyzed by Yun (EUROCRYPT '15). Some explicit bounds for various n with respect to k are also computed.

This second main contribution of this thesis describes methods of solving certain parameters of the discrete logarithm problem with low Hamming weight product exponents. Our approach is shown to be applicable for a concrete analysis of the GPS identification scheme. To achieve this, we introduce the notion of parameters dependent splitting system which served as tools to yield two improved results. The first attains a lower time complexity over the current state of the art without any compromise in memory. The second achieves the first known attack of the GPS scheme in a time complexity of under 2^{64} at the expense of some added memory requirements over the former. Furthermore, our analysis uncovers classes of parameters that are susceptible to such an improved attack. Overall, this work also serves as

a framework to identify parameters which are more vulnerable in the design of future cryptosystems based on the discrete logarithm problem with low Hamming weight exponents.

Time-memory trade-off methods provide means to invert one way functions. Such attacks offer a flexible trade-off between running time and memory cost in accordance to users' computational resources. In particular, they can be applied to hash values of passwords in order to recover the plaintext. They were introduced by Martin Hellman and later improved by Philippe Oechslin with the introduction of rainbow tables. The drawbacks of rainbow tables are that they do not always guarantee a successful inversion.

This third main contribution of this thesis address such issues described in the previous paragraph. In the context of passwords, it is pertinent that frequently used passwords are incorporated in the rainbow table. It has been known that up to four given passwords can be incorporated into a chain but it is an open problem if more than four passwords can be achieved. We solve this problem by showing that it is possible to incorporate more of such passwords along a chain. Furthermore, we prove that this results in faster recovery of such passwords during the online running phase as opposed to assigning them at the beginning of the chains. For large chain lengths, the average improvement translates to three times the speed increase during the online recovery time.

Contents

Chapter 1

Introduction

1.1 Overview and Motivations

The advent of the Information Age has brought about revolutions in how information is stored and transmitted. Since then, the world has seen massive developments progressed rapidly over the past decades. We are now living in an era where much of the data everywhere is handled and processed electronically. This has engendered a necessity for secure handling of confidential and private data. Such means are achieved through cryptography; which is the study of techniques for secure communications. In particular, modern cryptography utilizes a hybrid of tools in the fields of mathematics and computer science. Some of its essential applications include electronic commerce and computer passwords. The cores of modern cryptosystems are based upon cryptographic primitives. Such cryptographic primitives are the building blocks of secure cryptosystems. As such, the securities of cryptosystems largely depend on their underlying primitives. It is therefore imperative that the securities of these cryptographic primitives are well understood. In this thesis, we analyze two commonly deployed primitives; the discrete logarithm problem and cryptographic hash functions. Our analysis considers a variety of settings in which they occur and their respective computational complexities.

Public key cryptography is a cryptographic system consisting of pairs of keys: public keys and private keys. Public keys are accessible to everyone whereas private keys are secret and only known by the owner. In a public key encryption system, anyone can encrypt a message using the public key of the receiver, but such a message can only be decrypted with the receiver's private key. For practical usage, the pair of public and private keys needs to be computationally easily generated by the user. The hardness of solving certain computational problems is a fundamental aspect in the security of public key cryptosystems. An advantage of public key algorithms over symmetric key algorithms is that the former do not require a secure channel during the initial exchange of secret keys between the parties. The difficulty of the discrete logarithm problem is a basis for the constructions of various cryptographic systems. As such, an in depth understanding of its security is of paramount importance.

Denote the order of an algebraic group to be p, a sufficiently large prime. There are generic algorithms that are known to solve the discrete logarithm problem in $O(\sqrt{p})$. It was also shown by Shoup that any algorithm solving the generic discrete logarithm requires at least $\Omega(\sqrt{p})$ group operations. This provides some form of assurance that the discrete logarithm problem is computationally intractable as long as the underlying group is carefully chosen.

Since the emergence of elliptic curve cryptography, NIST has recommended a number of standard fixed curves for use in cryptographic schemes. As a result, there is an interest in understanding the security of solving multiple instances of the discrete logarithm problem arising from the same curve. This is referred to as the multiple discrete logarithm problem. A trivial method to resolve this is to simply solve each discrete logarithm instance separately, one after another which results in a trivial complexity of $O(k\sqrt{p})$. It was subsequently shown that the bound can be reduced to $O(\sqrt{kp})$. More recently, it was proven by Yun that any generic algorithm to solve the multiple discrete logarithm problem requires at least $\Omega(\sqrt{kp})$ group operations.

The first work presented in this thesis generalizes the results of Yun by investigating the complexity of solving the generalized multiple discrete logarithm problem, i.e. solving k out of n instances of the discrete logarithm problem. We obtain concrete rigorous lower bounds for any generic algorithm required to solve some subcollection of multiple discrete logarithms instances. Our analysis is also extended to other settings of the discrete logarithm problem.

There are other variations of the discrete logarithm problem having certain structures which arise in other cryptographic schemes. In particular, the GPS identification scheme utilizes the discrete logarithm problem with low Hamming weight product exponents as its security basis. Low Hamming weight products provide the advantage of speeding up the online computations of the identification scheme. On the other hand, it has to be evaluated to ensure that security is not compromised in this variation. Coron *et al.* proposed certain parameters of low Hamming weight products and analyzed the security of those set of parameters based on computational and storage requirements. Improvements to the computational complexity were subsequently discovered by Kim-Cheon.

Our second main contribution details a method of solving the discrete logarithm problem with low Hamming weight exponents. We introduce a notion of parameter dependent splitting system as a tool to solve it. We show that this splitting system provides an improvement over the parameterized splitting system introduced by Kim-Cheon in many instances. More pertinently, this can be applied to analyze the security of the GPS identification scheme utilizing the parameters proposed by Coron *et al.*. Our results provide a lower attack complexity over all current known methods.

A cryptographic hash function is an algorithm that maps data of arbitrary size to a bit string of a fixed size. It is designed as a one way function such that it is computationally infeasible to invert. The output of a hash function is commonly referred to as the hash digest.

An application of hash functions in the context of our work arises in user authentication and more specifically in password verification. The storage of all user passwords as clear plaintext can potentially result in a security breach if the password file is compromised. To resolve this issue, only the hash digests of passwords are stored. During the authentication process, the password provided by the user is hashed and a comparison with the stored hash is performed. A match will result in a successful authentication and a failure otherwise.

Passwords provide the most convenient and popular means of user authentication. Password based authentication systems are deployed to safeguard sensitive information against intruders and malicious adversaries. Two common forms of passwords are textual and graphical. For textual password systems, the authentication mechanism is based on a string of characters associated to each user. On the other hand, graphical password systems require the selection of images in a specific order as the primary form of authentication. In recent years, there has been some amount of interest in graphical password based systems. For instance, some research have been undertaken to explore the feasibility where the images concerned are maps. In this particular case, the graphical password associated to an individual user is a sequence of regions within the map. Nevertheless, textual password based systems remain the most predominant and most widespread means of user authentication.

Some methods of decrypting password hashes include exhaustive search, precomputations, targeted dictionary attacks, exploiting possible vulnerabilities of certain hashing algorithms and time-memory trade-off techniques. Exhaustive search is infeasible in most scenarios since all possible permutations of the permissible password characters in the password space are attempted resulting in an impractical amount of time during the online decryption phase. Precomputations refer to the method of storing plaintext hash pairs of all every string of characters in the passwords space. In reality, an unrealistic amount of storage space is typically required in this case. Targeted dictionary attacks aim at exploiting users' tendencies to use common popular passwords by specifically targeting such perceived ones via exhaustive search. The class of time-memory trade-off techniques represents a hybrid of exhaustive search and precomputations. This tool was first introduced by Hellman and subsequent refinements were developed by Oechslin which are referred to as rainbow tables. Rainbow tables provide a more efficient time-memory trade-off but certain drawbacks remain inherent. In particular, a successful hash inversion of a given password hash digest is not always guaranteed.

Our third main contribution attempts to address this issue to some extent by including additional features of rainbow tables during the generation phase. More specifically, we introduce an enhanced construction of rainbow tables which can incorporate given passwords along a rainbow chain. The advantages are two-fold. Firstly, this ensures that commonly used passwords can be recovered within a rainbow table. Secondly, incorporating them along a chain provides an improved efficiency over conventional means. An analysis of the improvements is also provided.

In a broad sense, cryptographic primitives are ubiquitous in the field of cryptography. They arise in many different settings and the work in this thesis evaluates the computational complexity when deploying cryptographic primitives in various scenarios relevant to cryptography and information security.

1.2 Overall Organization

This thesis is organized as follows. The remainder of Chapter 1 contains the definitions of several standard notations. Chapter 2 computes bounds in various generalized settings of the discrete logarithm problem. Chapter 3 presents methods for solving the discrete logarithm problem with low Hamming weight product exponents as well as improved attacks on the GPS identification scheme. Chapter 4 covers the decryption of frequent password hashes in rainbow tables. Within each individual Chapters 2, 3 and 4, a summary of our contributions is also included. We provide an overall conclusion in Chapter 5.

1.3 Notations

We present several standard notations that are used throughout this thesis.

$O(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c, n_0 > 0 \text{ s.t. } 0 \le f(n) \le cg(n)\ \forall n > n_0\}$;

$\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c, n_0 > 0 \text{ s.t. } cg(n) \le f(n)\ \forall n > n_0\}$;

$\Theta(g) = \{f : f = O(g) \text{ and } g = O(f)\}$;

$\tilde{O}(g) : O(g)$ with logarithmic factors ignored;

$f(n) = o(g(n)) \iff \forall k > 0\ \exists n_0 \text{ s.t. } \forall n > n_0, |f(n)| \le k|g(n)|$;

$\mathbb{Z}_p :$ the group of primitive residue classes modulo a prime p;

$\gamma :$ the Euler-Mascheroni constant;

$B(n,p) :$ the binomial distribution with the parameters n and p;

Chapter 2

Hardness Bounds of the Generalized Multiple Discrete Logarithm Problems

2.1 Introduction

Many variants of the discrete logarithm problem have evolved over the years. Some of these include the Bilinear Diffie-Hellman Exponent Problem [14], the Bilinear Diffie-Hellman Inversion Problem [12], the Weak Diffie-Hellman Problem [39] and the Strong Diffie-Hellman Problem [13]. We first describe the classical discrete logarithm problem and its generalizations. Formal descriptions of the statements are as follow.

Let G be a cyclic group such that $|G| = p$ where p a prime and denote g to be a generator of G so that $G = \langle g \rangle$.

The discrete logarithm problem (DLP) is defined as follows: Given G, p and any h selected uniformly at random from G, find $x \in \mathbb{Z}_p$ satisfying $g^x = h$.

The k-Multiple Discrete Logarithm (k-MDL) is defined as follows:

Definition 2.1.1 (MDL). *Given G, p and k elements h_1, h_2, …, h_k selected uniformly at random from G, find non-negative integers x_1, x_2, …, x_k satisfying $g^{x_i} = h_i$ $\forall i \in \mathbb{Z}^+$ such that $1 \leq i \leq k$.*

In particular when $k = 1$, the 1-MDL is equivalent to DLP.

For $k \leq n$, we define the (k,n)-Generalized Multiple Discrete Logarithm ((k,n)-GMDL) as follows:

Definition 2.1.2 (GMDL). *Given G, p and n elements h_1, h_2, $\dots$, h_n selected uniformly at random from G, find k pairs (i, x_i) satisfying $g^{x_i} = h_i$ where $i \in S$ and where S is a k-subset of $\{1, \dots, n\}$.*

As the definition suggests, (k,n)-GMDL can be viewed as a generalization of k-MDL. In particular when $n = k$, the (k,k)-GMDL is equivalent to the k-MDL.

Cryptographic constructions based on DLP are applied extensively. For instance, an early application of the DLP in cryptography came in the form of the Diffie-Hellman key exchange protocol [19] for which the security is dependent on the hardness of the DLP. Among some of the others include the ElGamal encryption and signature schemes [20] as well as Schnorr's signature scheme and identification protocol [48]. The multiple discrete logarithm problem mainly arises from elliptic curve cryptography. NIST recommended a small set of fixed (or standard) curves for use in cryptographic schemes [1] to eliminate the computational cost of generating random secure elliptic curves. The implications for the security of standard elliptic curves over random elliptic curves were analysed based on the efficiency of solving multiple discrete logarithm problems [28].

In a generic group, no special properties which are exhibited by any specific groups or their elements are assumed. Algorithms for a generic group are termed as generic algorithms. There are a number of results pertaining to generic algorithms for DLP and k-MDL.

Shoup showed that any generic algorithm for solving the DLP must perform $\Omega(\sqrt{p})$ group operations [49]. There are a few methods for computing discrete logarithm in approximately $\sqrt{p}$ operations. For example, Shanks Baby-Step-Giant-Step method computes the DLP in $\tilde{O}(\sqrt{p})$ operations. One other method is the Pollard's Rho Algorithm which can be achieved in $O(\sqrt{p})$ operations [42]. Since then, further practical improvements to the Pollard's Rho Algorithm have been proposed in [10, 15, 51] but the computational complexity remains the same. There exist index calculus methods which solve the DLP in subexponential time. Some of these more recent works include [7, 32] for finite fields of small characteristic (with the latter achieving a heuristic quasi-polynomial algorithm) and [8] for finite fields of medium to high characteristic. However, such index calculus methods are not relevant in our context since they are not applicable for a generic group. Hence, currently known generic algorithms are asymptotically optimal.

An extension of Pollard's Rho algorithm was proposed in [36] which solves k-MDL in $O(\sqrt{kp})$ group operations if $k \leq O(p^{1/4})$. It was subsequently shown in [21] that $O(\sqrt{kp})$ can in fact be achieved without the imposed condition on k. The former's method of finding discrete logarithm is sequential in the sense that they are found one after another. However in the latter's method, all the discrete logarithms can only be obtained towards the end. Finally, it was presented in [58] that any generic algorithm solving k-MDL must require at least $\Omega(\sqrt{kp})$ group operations if $k = o(p)$. This shows that the algorithms of [36, 21] are asymptotically optimal, up to factors polynomial in $\log p$.

2.1.1 Our Contributions

In the context of our work, suppose an adversary has knowledge or access to many instances of the discrete logarithm problem either from a generic underlying algebraic group or from a standard curve recommended by NIST. Our work investigates how difficult it is for such an adversary to solve subcollections of those instances. One of our result outcomes in this work shows that an adversary gaining access to additional instances of the DLP provides no advantage in solving some subcollection of them when k is small and for corresponding small classes of n. Our techniques are also applicable to other standard non-NIST based curves. For instance, the results in this work are relevant to Curve25519 [9] which has garnered considerable interest in recent years. Furthermore, we also establish formal lower bounds for the generic hardness of solving the GMDL problem for larger k values. As a corollary, these results provide the lower bounds of solving the GMDL problem for the full possible range of inputs k. Part of this work can be viewed as a generalization of the results in [58].

More specifically, we introduce two techniques to solve such generalized multiple discrete logarithm problems. The first we refer to as the matrix method which is also shown to achieve asymptotically tight bounds when the inputs are small. From this, we obtain the result that the GMDL problem is as hard as the MDL problem for $k = o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$ and $kn^2 = o\left(\frac{p}{\log^2 p}\right)$. This strictly improves the result of [36] where the equivalence is achieved for a smaller range of inputs satisfying $k = o(p^{\frac{1}{4}})$ and $k^2n^2 = o(p)$. We also analyse its trade-off efficiency when the sizes of matrices involved are varied. The second technique is referred as the block method which can be applied for larger inputs. We also show that the block partitioning in this method is optimized. Moreover, when n is relatively small with respect to k, the bounds that are obtained in this way are also asymptotically tight. Furthermore, we demonstrate that the block method can be adapted and applied to generalized versions of other discrete logarithm settings introduced in [35] to also obtain generic hardness bounds for such problems. For instance, part of this work also shows that solving one out of n instances of the Discrete Logarithm Problem with Auxiliary Inputs is as hard as solving a single given instance when n is not too large. In addition, we also explain why the matrix method cannot be extended to solve these problems.

2.1.2 Chapter Organization

This chapter is organized as follows. The hardness bounds of the $(1, n)$-GMDL problem and the (k, n)-GMDL problem which were obtained in [36] will be covered in Chapter 2.2. Chapter 2.3 describes the matrix method used to solve the GMDL problem. An in depth analysis of the matrix method is carried out in Chapter 2.4. We present the block method as well as establish generalized bounds in Chapter 2.5. The bounds obtained for GMDL problem can be viewed as a generalization of the bounds for the MDL problem. Indeed for $n = k$, our bounds correspond to the results of [58]. The contents of Chapter 2.6 show the partition of n carried out in the methods of Chapter 2.5 achieves the fastest running time among all other possible partitions. Chapter 2.7 discusses the applicability of our techniques in other discrete logarithm settings. In Chapter 2.8, we provide explicit bounds for various values of n relative to k. We conclude in Chapter 2.9.

2.2 Preliminaries

For generic groups of large prime order p, denote T_k, $T_{k,n}$ to be the expected workload (in group operations) of an optimal algorithm solving the k-MDL problem and (k, n)-GMDL problem respectively.

Lemma 2.2.1 and Corollary 2.2.1 for a special case $k = 1$ are attributed to [36].

Lemma 2.2.1. $T_1 \leq T_{1,n} + 2n \log_2 p$

Proof. Given an arbitrary $h \in G = \langle g \rangle$, obtaining x such that $g^x = h$ can be achieved in time T_1. For all i, $1 \leq i \leq n$, select integers r_i uniformly at random from the set $\{0, \ldots, p-1\}$ and define $h_i := g^{r_i} h = g^{x+r_i}$. All the h_i are random since all the r_i and h are random. Apply a generic algorithm with inputs of $(h_1, h_2, \ldots, h_n)$ that solves the $(1, n)$-GMDL problem in time $T_{1,n}$. The resulting algorithm outputs (j, y) such that $h_j = g^y$, $1 \leq j \leq n$. Therefore, $x \equiv y - r_j \bmod p$, thus solving the 1-MDL problem within $T_{1,n} + 2n \log_2 p$ group multiplications. □

Corollary 2.2.1. *For all* $n = o\left(\frac{\sqrt{p}}{\log p}\right)$,

$$T_{1,n} = \Omega(\sqrt{p}).$$

Proof. Since $T_1 = \Omega(\sqrt{p})$ [49], when $n = o\left(\frac{\sqrt{p}}{\log p}\right)$, it follows directly from Lemma 3.2.1 that $T_{1,n} = \Omega(\sqrt{p})$. □

It was also obtained in [36] that the GMDL problem is as hard as the MDL problem if $kn \ll \sqrt{p}$. Since $k \leq n$, this equivalence is valid for $k = o(p^{\frac{1}{4}})$ and and $k^2 n^2 = o(p)$.

2.3 Generalized Bounds of $T_{k,n}$ for small k

The first method we introduce is to obtain an improved lower bound of $T_{k,n}$ for small k. We refer to this as the Matrix technique.

We seek to obtain an upper bound of T_k based on $T_{k,n}$. Given $g^{x_i} = h_i$ $\forall\ 1 \leq i \leq k$, T_k represents the time to solve all such x_i. For all $1 \leq i \leq n$, denote y_i by the following[1]:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}$$

[1]This representation of y_i came about from a suggestion by Phong Q. Nguyen.

Next, multiply each g^{y_i} with a corresponding random element g^{r_i} where $0 \leq r_i \leq p-1$. By considering these randomized $g^{y_i+r_i}$ as inputs to a (k, n)-GMDL solver, this solver outputs solutions to k out of n of such discrete logarithms. These solutions are of the form y_i+r_i. As such, a total of k values of y_i can be obtained by simply subtracting from their corresponding r_i. We claim that any k collections of y_i is sufficient to recover all of $x_1, x_2, \ldots, x_k$. Indeed, it suffices to show that any k-by-k submatrices of

$$V = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

has non zero determinant. This can be satisfied by simply letting $\alpha_i = i$ since V is a Vandermonde matrix.

In this case, recovering $x_1, x_2, \ldots, x_k$ from k number of y_i requires solving a k-by-k system of linear equations. This can be achieved in $O(k^3)$ arithmetic operations using Gaussian elimination. Crucially, this does not involve any group operations. On the other hand, group operations are incurred from the computations of all

$$g^{y_i} = g^{x_1+\alpha_i x_2+\cdots+\alpha_i^{k-1}x_k}. \tag{2.1}$$

Since $\alpha_i = i$, this process requires the computations of each $(g^{x_j})^{i^{j-1}}$ $\forall\, 1 \leq i \leq n$, $1 \leq j \leq k$. Denote $a_{i,j} = (g^{x_j})^{i^{j-1}}$. By noting that $a_{i+1,j} = a_{i,j}^{(\frac{i+1}{i})^{j-1}}$, it can be concluded that computing $a_{i+1,j}$ given $a_{i,j}$ requires at most $2(j-1)\log_2 \frac{i+1}{i}$ group multiplications. Moreover, $a_{1,j} = g^{x_j}$ is already known. Hence the total number of groups multiplications required to compute all $(g^{x_j})^{i^{j-1}}$ is at most

$$\sum_{j=1}^{k} \sum_{i=1}^{n-1} 2(j-1)\log_2\left(\frac{i+1}{i}\right) = k(k-1)\log_2 n. \tag{2.2}$$

Furthermore, each addition in the exponent of $g^{x_1+\alpha_i x_2+\cdots+\alpha_i^{k-1}x_k+r_i}$ constitutes a group multiplication. Therefore, kn group multiplications are necessary in this step. Thus, the total number of group multiplications required to compute all of $g^{y_i+r_i}$ is at most

$$kn + k(k-1)\log_2 n.$$

Since $k \leq n < p$, the above expression can be bounded from above by $2kn \log_2 p$ and so it follows that

$$T_k \leq T_{k,n} + 2kn \log_2 p. \tag{2.3}$$

Since $T_k = \Omega(\sqrt{kp})$ [58], from equation (2.3), $T_{k,n}$ is asymptotically as large as T_k if $nk \log p \ll \sqrt{kp}$. Hence, $T_{k,n} = \Omega(\sqrt{kp})$ if $k = o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$ and $n\sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)$. Moreover, this bound is asymptotically tight since there exists an algorithm which solves k-MDL in $O(\sqrt{kp})$.

2.4 Trade-off between Arithmetic Operations and GMDL calls

This section is mainly more of theoretical interests than of practical intents. The main computational bottleneck for arithmetic operations lies in solving the large $k \times k$ system of linear equations. It is thus natural to consider breaking the large matrix into multiple finer granularities and solving each of them separately. Therefore, instead of solving all x_i together in a sense, we recover them separately in smaller subsets, one subset at a time. One might expect this process allows the Gaussian elimination phase to be reduced considerably. However, there is a trade-off that more calls to the GMDL solver are required.

Denote $m(k)$ to be the number of such subsets and $f_1(k), f_2(k), \ldots, f_m(k)$ to be the size of each subset. In this case, the Matrix technique requires m calls to the GMDL solver. Furthermore, the complexity of the Gaussian elimination phase is given by $\Theta(\sum_{i=1}^{m} f_i^3)$. We obtain a feasible bound by proving the following claim.

Theorem 2.4.1. *Let m be a function $m(k) : \mathbb{Z}^+ \to \mathbb{Z}^+$ and $f_1(k), f_2(k), \ldots, f_m(k)$ be positive functions of k such that $\sum_{i=1}^{m} f_i = k$. Then,*

$$\sum_{i=1}^{m} f_i^3 \geq \frac{k^3}{m^2}.$$

Proof. Since

$$\sum_{1\le i<j\le m} (f_i - f_j)^2 \ge 0, \tag{2.4}$$

the following ensue:

$$(m-1)\sum_{i=1}^{m} f_i^2 \ge 2\sum_{1\le i<j\le m} f_i f_j \tag{2.5}$$

$$m\sum_{i=1}^{m} f_i^2 \ge 2\sum_{1\le i<j\le m} f_i f_j + \sum_{i=1}^{m} f_i^2 \implies \sum_{i=1}^{m} f_i^2 \ge \frac{k^2}{m} \tag{2.6}$$

Moreover, by noting that all f_i are positive,

$$\sum_{1\le i<j\le m} (f_i - f_j)^2(f_i + f_j) \ge 0, \tag{2.7}$$

we obtain

$$\sum_{1\le i<j\le m} (f_i^3 + f_j^3) \ge \sum_{1\le i<j\le m} (f_i^2 f_j + f_i f_j^2) \tag{2.8}$$

$$m\sum_{i=1}^{m} f_i^3 \ge \sum_{1\le i<j\le m} (f_i^2 f_j + f_i f_j^2) + \sum_{i=1}^{m} f_i^3 \tag{2.9}$$

$$m\sum_{i=1}^{m} f_i^3 \ge (\sum_{i=1}^{m} f_i^2)(\sum_{i=1}^{m} f_i) = k\sum_{i=1}^{m} f_i^2. \tag{2.10}$$

Combining inequality (2.10) with the previously obtained $\sum_{i=1}^{m} f_i^2 \ge \frac{k^2}{m}$ from inequality (2.6), it follows that

$$\sum_{i=1}^{m} f_i^3 \ge \frac{k^3}{m^2}. \tag{2.11}$$

□

Hence from inequality (2.11), the complexity of the Gaussian elimination phase is given by $\sum_{i=1}^{m} f_i^3 = \Omega(\frac{k^3}{m^2})$, where m is the trade-off parameter for the number of GMDL calls. In particular, if the number of calls is minimized to some constant, then the Gaussian elimination phase incurs at least cubic time of arithmetic operations with respect to k.

2.5 Generalized Bounds of $T_{k,n}$ for larger k

The matrix technique has been shown to provide asymptotically tight bounds required to solve k out of n of the multiple discrete logarithm in the classical setting when the inputs are small. One main limitation of this technique is that it is only applicable to the classical DLP and cannot be extended for other variants or other settings of the DLP. This will be shown in further details in the subsequent sections. Moreover, in light of the fact that the bound of T_k can be achieved for large inputs extending to $o(p)$, the matrix method is not sufficient to obtain analogous bounds for larger such k values. In this section, we address these issues by introducing the block method to evaluate lower bounds of $T_{k,n}$ for general k, including large k values. Moreover, the block technique can also be applied to other variants or other settings in the MDLP. This will be also described in the later sections.

Proposition 2.5.1. *Suppose that $n \geq k^2$. Then,*

$$T_k \leq kT_{k,n} + 2nH_k \log_2 p$$

where H_k denotes the k^{th} harmonic number, $H_k = \sum_{i=1}^{k} \frac{1}{i}$.

Proof. Given arbitraries $h_1, h_2, \ldots, h_k \in G = \langle g \rangle$, obtaining $x_1, x_2, \ldots, x_k$ such that $g^{x_i} = h_i \; \forall \; 1 \leq i \leq k$ can be achieved in time T_k. Consider n elements partitioned into k blocks each of size approximately s_k, where $s_k = \frac{n}{k}$. Each block is labelled i where i ranges from 1 to k. For each i, $1 \leq i \leq k$, select about s_k integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k,n)-GMDL problem, outputs k pseudo solutions. Computing each $h_{i,j}$ requires at most $2\log_2 p$ group multiplications. Since each block is about size $\frac{n}{k} \geq k$, these k pseudo solutions might be derived from the same block. In which case, the algorithm outputs k of $((i', j),\, y_{i',j})$ such that $h_{i',j} = g^{y_{i',j}}$ for some i'. As a result, one can obtain $x_{i'} \equiv y_{i',j} - r_{i',j} \bmod p$ but derive no other information of other values of x_i. This invokes at most $T_{k,n} + 2n\log_2 p$ group operations. Figure 3.1 illustrates an overview of the first phase.

The second phase proceeds as follows. Since 1 out of k discrete logarithms has been obtained, discard the block for which that determined discrete logarithm is contained previously. For each of $i \in \{1, \ldots, k\} \setminus \{i'\}$, select about $\frac{s_k}{k-1}$ integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define

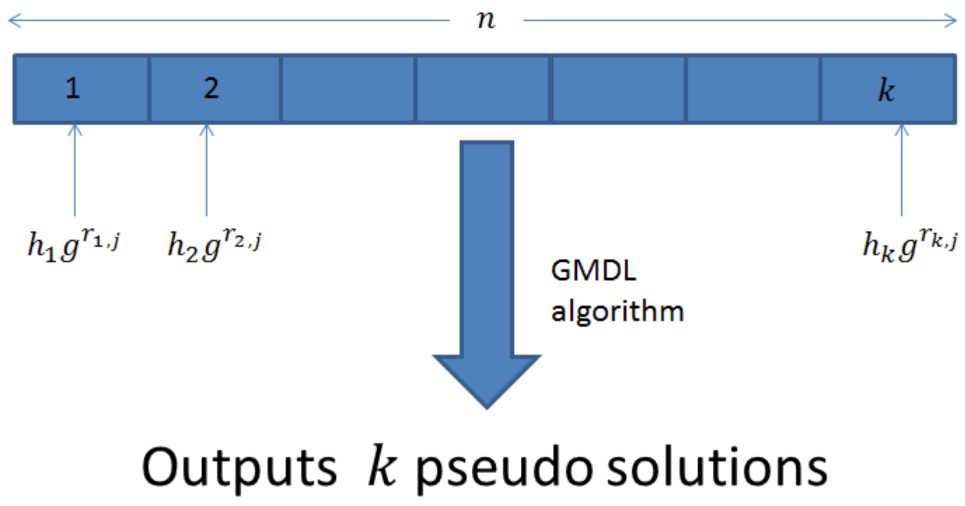

Figure 2.1: Overview of the first phase

$h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $k-1$ blocks. Hence, each of the remaining $k-1$ unsolved discrete logarithms are contained separately in $k-1$ blocks each of size approximately $\frac{n}{k-1}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Since each block is about size $\frac{n}{k-1} \geq k$, these k pseudo solutions might once again be derived from the same block. In which case, the algorithm outputs k of $((i'', j),\ y_{i'',j})$ such that $h_{i'',j} = g^{y_{i'',j}}$ for some i''. As a result, one can obtain $x_{i''} \equiv y_{i'',j} - r_{i'',j} \mod p$ but derive no other information for the other remaining values of x_i. This second phase incurs at most $T_{k,n} + 2s_k \log_2 p$ group operations.

The third phase executes in a similar manner to the second phase as follows. For each of $i \in \{1, \ldots, k\} \setminus \{i',\ i''\}$, select about $\frac{s_{k-1}}{k-2}$ integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $k-2$ blocks. Hence, each of the remaining $k-2$ unsolved discrete logarithms are contained separately in $k-2$ blocks each of size approximately $\frac{n}{k-2}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Since each block is about size $\frac{n}{k-2} \geq k$, these k pseudo solutions might be derived from the same block. In which case, the algorithm outputs k of $((i^{(3)}, j),\ y_{i^{(3)},j})$ such that $h_{i^{(3)},j} = g^{y_{i^{(3)},j}}$ for some $i^{(3)}$. As a result, one can obtain $x_{i^{(3)}} \equiv y_{i^{(3)},j} - r_{i^{(3)},j} \mod p$ but derive no other information for the other remaining values of x_i. This second phase incurs at most $T_{k,n} + 2s_{k-1}\log_2 p$ group operations.

During each phase of the process, a generic algorithm for the (k,n)-GMDL problem can never guarantee outputs deriving from different blocks since $n \geq k^2$ implies $\frac{n}{k-i+1} \geq k$, $\forall 1 \leq i \leq k$. In general for $i \geq 2$, the maximum number of group operations required in the i^{th} phase is given by

$$T_{k,n} + 2s_{k+2-i}\log_2 p.$$

The process terminates when all k discrete logarithms have been obtained. Since each phase outputs exactly 1 out of k discrete logarithms, all k discrete logarithms can be determined after k phases. Therefore, the following inequality can be obtained.

$$T_k \leq T_{k,n} + 2n\log_2 p + \sum_{i=2}^{k}(T_{k,n} + 2s_{k+2-i}\log_2 p) \tag{2.12}$$

Since $s_k = \frac{n}{k}$,

$$\sum_{i=2}^{k}(T_{k,n} + 2s_{k+2-i}\log_2 p) = (k-1)T_{k,n} + (2\log_2 p)\sum_{i=2}^{k} s_i$$

$$= (k-1)T_{k,n} + (2\log_2 p)\sum_{i=2}^{k}\frac{n}{i}.$$

Hence, it follows that

$$T_k \leq kT_{k,n} + 2n(1 + \sum_{i=2}^{k}\frac{1}{i})\log_2 p = kT_{k,n} + 2nH_k\log_2 p. \tag{2.13}$$

This completes the proof. □

Remark. When $k = 1$, Proposition 2.5.1 corresponds to Lemma 2.2.1.

By regarding $k = k(p)$ as a function of p such that $\lim_{p\to+\infty} k(p) = +\infty$, the asymptotic bounds of $T_{k,n}$ can be obtained.

Theorem 2.5.1. *Suppose* $k^3\log^2 k = o\left(\frac{p}{\log^2 p}\right)$, *then*

$$T_{k,n} = \Omega\left(\sqrt{\frac{p}{k}}\right)$$

for all n satisfying $n = k^2 + \Omega(1)$ and $n = o\left(\frac{\sqrt{kp}}{(\log k)(\log p)}\right)$.

Proof. Clearly $k^3 \log^2 k = o\left(\frac{p}{\log^2 p}\right)$ implies that $k = o(p)$. Hence from [58], $T_k = \Omega(\sqrt{kp})$. It follows from Proposition 2.5.1 that if $nH_k \log_2 p \ll T_k$, then

$$T_{k,n} = \Omega(\frac{1}{k}\sqrt{kp}) = \Omega\left(\sqrt{\frac{p}{k}}\right). \tag{2.14}$$

Moreover, since

$$\lim_{k\to+\infty}(H_k - \log k) = \lim_{k\to+\infty}[(\sum_{i=1}^{k}\frac{1}{i}) - \log k] = \gamma \tag{2.15}$$

where γ is the Euler-Mascheroni constant, the condition $nH_k \log_2 p \ll T_k$ implies that $n = o\left(\frac{\sqrt{kp}}{(\log k)(\log p)}\right)$. The lower bound $n = k^2 + \Omega(1)$ is obtained by noting that n has to be of size at least k^2 from the condition of Proposition 2.5.1. Finally, since the lower bound for n cannot be asymptotically greater than its upper bound, $k(p)$ has to satisfy $k^3 \log^2 k = o\left(\frac{p}{\log^2 p}\right)$. This completes the proof of Theorem 2.5.1. □

Remark. Although Theorem 2.5.1 holds for a wide asymptotic range of n as given, the $T_{k,n}$ bound becomes sharper as n approaches $\frac{\sqrt{kp}}{(\log k)(\log p)}$. In essence, Theorem 2.5.1 does not yield interesting bounds but is a prelude to the more essential Theorem 2.5.2 which requires Proposition 2.5.1 and is hence included.

Proposition 2.5.2. *Suppose that $k < n < k^2$. Then,*

$$T_k \leq (r + \frac{n}{k})T_{k,n} + 2rk\log_2 p + 2nH_{\lceil \frac{n}{k} \rceil}\log_2 p$$

where

$$r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil.$$

Proof. The proof comprises two main phases. The former consists of the initializing phase followed by subsequent subphases. It utilizes the extended pigeon hole principle to obtain more than one solution during each of the initial subphases. The latter phase takes place after some point where the number of remaining unknown discrete logarithms is small enough such that each subphase can only recover one discrete logarithm. After this point, the method of determining all other discrete logarithms essentially mirrors that of the method described in the proof of Proposition 2.5.1. The formal proof and details are given as follows.

The initializing phase proceeds as follows. Given arbitraries $h_1, h_2, \ldots, h_k \in G = \langle g \rangle$, obtaining $x_1, x_2, \ldots, x_k$ such that $g^{x_i} = h_i \ \forall \ 1 \le i \le k$ can be achieved in time T_k. Consider n elements partitioned into k blocks each of size approximately s_k, where $s_k = \frac{n}{k}$. Each block is labelled i where i ranges from 1 to k. For each i, $1 \le i \le k$, select about s_k integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Computing each $h_{i,j}$ requires at most $2 \log_2 p$ group multiplications. Since each block is about size $\frac{n}{k} < k$, by the extended pigeon hole principle, at least $\frac{k^2}{n}$ out of these k solutions must be derived from distinct blocks. In other words, at least $\frac{k^2}{n}$ correspond to distinct i values and as a result, $\frac{k^2}{n}$ discrete logarithms out of k discrete logarithms can be obtained during this initializing phase. This invokes at most $T_{k,n} + 2n \log_2 p$ group operations.

The first subphase proceeds as follows. Since $\frac{k^2}{n}$ out of k discrete logarithms have been obtained, discard all the blocks for which those determined discrete logarithms are contained previously. Thus, about $k - \frac{k^2}{n} = \frac{k(n-k)}{n}$ blocks remain, each of size approximately $\frac{n}{k}$. For each of the remaining blocks i, select about k integers $r_{i,j}$ uniformly distributed across each i and uniformly at random from $\mathbb{Z}_p$. Define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $\frac{k(n-k)}{n}$ blocks. Hence, each of the remaining $\frac{k(n-k)}{n}$ unsolved discrete logarithms are contained separately in $\frac{k(n-k)}{n}$ blocks each of size approximately $\frac{n^2}{k(n-k)}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. This incurs a maximum of $T_{k,n} + 2k \log_2 p$ group operations.

If $k \le \frac{n^2}{k(n-k)}$, these k pseudo solutions might be derived from the same block and hence phase two begins in which the method described in Proposition 1 can then be applied on this new set of blocks.

If $k > \frac{n^2}{k(n-k)}$, the extended pigeon hole principle ensures that at least

$\frac{k^2(n-k)}{n^2}$ of the k pseudo solutions correspond to distinct i values and a result, about $\frac{k^2(n-k)}{n^2}$ discrete logarithms can be obtained in the first subphase.

Subsequent subphases are similar to the first subphase. In general for the r^{th} subphase, since about $\frac{k^2(n-k)^{r-1}}{n^r}$ solutions will have been obtained in the $(r-1)^{th}$ subphase, discard all the blocks for which those determined discrete logarithms are contained previously. By a simple induction, it can be shown that the number of remaining blocks in the r^{th} subphase is about $\frac{k(n-k)^r}{n^r}$. The induction proceeds as follows. The base case has already been verified in the first subphase. Suppose the result holds for $r = m-1$ for some $m \geq 2$. By the inductive hypothesis, the number of blocks remaining in the $(m-1)^{th}$ subphase is about $\frac{k(n-k)^{m-1}}{n^{m-1}}$. During the m^{th} subphase, since about $\frac{k^2(n-k)^{m-1}}{n^m}$ solutions have already been obtained previously and are thus discarded, the number of remaining blocks is given by

$$\frac{k(n-k)^{m-1}}{n^{m-1}} - \frac{k^2(n-k)^{m-1}}{n^m} = \frac{k(n-k)^m}{n^m}. \tag{2.16}$$

This completes the induction. For each of the remaining blocks i, select about k integers $r_{i,j}$ uniformly distributed across each i and uniformly at random from $\mathbb{Z}_p$. Define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $\frac{k(n-k)^r}{n^r}$ blocks. Hence, each of the remaining $\frac{k(n-k)^r}{n^r}$ unsolved discrete logarithms are contained separately in $\frac{k(n-k)^r}{n^r}$ blocks each of size approximately $\frac{n^{r+1}}{k(n-k)^r}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k,n)-GMDL problem, outputs k pseudo solutions. Each of the r^{th} subphase requires at most $T_{k,n} + 2k\log_2 p$ group operations.

When $k \leq \frac{n^{r+1}}{k(n-k)^r}$, the k outputs can only guarantee one solution. Hence, as soon as r satisfies the above inequality, the first main phase terminates at the end of the r^{th} subphase and the second main phase commences. That is

$$k \leq \frac{n^{r+1}}{k(n-k)^r} \implies r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil. \tag{2.17}$$

At the beginning of second main phase, there are a total of about $\frac{k(n-k)^r}{n^r} - 1$ unresolved discrete logarithms. The rest of the procedure follows starting from the second phase of Proposition 2.5.1 until the end. Hence, it can immediately be derived from the proof of Proposition 2.5.1 that the number of group operations required to solve them all is at most $2[(\frac{k(n-k)^r}{n^r} - 1)T_{k,n}$

$+ nH_{\lceil \frac{k(n-k)^r}{n^r} \rceil} - n] \log_2 p$. Since

$$k \leq \frac{n^{r+1}}{k(n-k)^r} \implies \frac{k(n-k)^r}{n^r} \leq \frac{n}{k},$$

the maximum number of group operations required in the second main phase is given by

$$(\frac{n}{k} - 1)T_{k,n} + 2(nH_{\lceil \frac{n}{k} \rceil} - n) \log_2 p.$$

The number of group operations required during the first main phase is the sum of the number required for the initializing phase and the number required for all the subphases. Therefore, the maximum number of group operations required in the first main phase is given by

$$T_{k,n} + 2n \log_2 p + r(T_{k,n} + 2k \log_2 p) = (r+1)T_{k,n} + 2n \log_2 p + 2rk \log_2 p.$$

Thus the maximum number of group operations required to execute both the first main phase and the second main phase is given by

$$(r+1)T_{k,n} + 2(n+rk) \log_2 p + (\frac{n}{k} - 1)T_{k,n} + 2(nH_{\lceil \frac{n}{k} \rceil} - n) \log_2 p.$$

Hence,

$$T_k \leq (r + \frac{n}{k})T_{k,n} + 2rk \log_2 p + 2nH_{\lceil \frac{n}{k} \rceil} \log_2 p. \tag{2.18}$$

□

Remark. One other approach is to replace the (k, n)-GMDL solver with the $(1, n)$-GMDL solver during the second main phase. Both yield identical asymptotic results given in Chapter 2.8 as the computational bottleneck arises from the first main phase.

By regarding $k = k(p)$ as a function of p such that $\lim_{p \to +\infty} k(p) = +\infty$, the asymptotic bounds of $T_{k,n}$ can be obtained.

Theorem 2.5.2. *Suppose k, n satisfy the following conditions:*

1. $k = o(p)$
2. $n = k^2 - \Omega(1)$
3. $\frac{n}{\sqrt{k}} \log(\frac{n}{k}) = o\left(\frac{\sqrt{p}}{\log p}\right)$

4. $\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)$

Then,

$$T_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{p}\right)$$

where $r = r(k,n) \geq 1$ *is any function of* k *and* n *satisfying* $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$.

Proof. Condition 1 is necessary to utilize results in [58] for a lower bound of T_k. Condition 2 is required in order to apply the result of Proposition 2.5.2. Conditions 3 and 4 can be obtained by requiring $nH_{\lceil \frac{n}{k} \rceil}\log p \ll T_k$ and $rk\log p \ll T_k$ respectively and noting that $T_k = \Omega(\sqrt{kp})$. Hence from Proposition 2.5.2 and under these conditions, $T_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{p}\right)$. $\square$

It should be mentioned that $r(k,n)$ can be taken to be any function satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$. However, it is clear that $T_{k,n}$ achieves sharper bounds for asymptotically smaller choices of r. One other point of note is that when $n = k$ where $k = o(p)$, all the 4 conditions are satisfied and $r(k,n)$ can be taken to be 1. In this case,

$$T_k = T_{k,k} = \Omega(\sqrt{kp}) \tag{2.19}$$

which indeed corresponds to the bound obtained in [58].

2.6 Optimizing the Partition of n

From the methods described in the proofs of Propositions 2.5.1 and 2.5.2, n is partitioned into blocks of approximately equal size at each phase. There are many ways to perform such partitions of n. The running time of each phase is partially determined by the number of uniformly randomly chosen $r_{i,j}$, which invariably depends on the partition of n. In this section, we show that the method of partition described in the proofs of the earlier Propositions minimizes the expected number of chosen $r_{i,j}$ required and hence results in the fastest running time among all other possible partitions. We first consider the case where a (k,n)-GMDL solver output solutions derived from the same block so only one discrete logarithm can be determined at each phase. We follow this up by considering general scenarios where a (k, n)-GMDL solver outputs solutions derived from multiple blocks.

Chebyshev's Sum Inequality
We first provide a statement and proof of Chebyshev's sum inequality which is utilized to establish results in the subsequent parts of this section.

Chebyshev's inequality. *If $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$, the following inequality holds.*

$$n \left(\sum_{i=1}^{n} x_i y_i \right) \geq \left(\sum_{i=1}^{n} x_i \right) \left(\sum_{i=1}^{n} y_i \right)$$

Proof. From the rearrangement inequality, the following system of inequalities can be obtained.

$$\sum_{i=1}^{n} x_i y_i \geq x_1 y_1 + x_2 y_2 + \ldots x_n y_n \tag{2.20}$$

$$\sum_{i=1}^{n} x_i y_i \geq x_1 y_2 + x_2 y_3 + \ldots x_n y_1 \tag{2.21}$$

$$\ldots$$

$$\sum_{i=1}^{n} x_i y_i \geq x_1 y_n + x_2 y_1 + \ldots x_n y_{n-1} \tag{2.22}$$

The result follows by summing all the above inequalities. $\square$

Pseudo Solutions Deriving from the Same Block
Denote s_i to be the size of block i, $k \leq s_i$, $1 \leq i \leq k$. so that $\sum_{i=1}^{k} s_i = n$. Let $p_{i,k}$ be the conditional probability that the k output solutions derive from block i given that the k output solutions derive from the same block. Then, $p_{i,k}$ can be expressed by the following.

$$p_{i,k} = \frac{\binom{s_i}{k}}{\sum_{j=1}^{k} \binom{s_j}{k}} \tag{2.23}$$

Suppose a solution is derived from block i. Upon discarding block i, s_i of $r_{i,j}$ have to be randomly chosen to fill the remaining blocks so that they sum back up to n. Let $E_k^{(1)}$ be the expected number of randomly chosen $r_{i,j}$ required. Then, $E_k^{(1)}$ can be expressed by the following.

$$E_k^{(1)} = \sum_{i=1}^{k} p_{i,k} s_i = \frac{\sum_{i=1}^{k} \binom{s_i}{k} s_i}{\sum_{i=1}^{k} \binom{s_i}{k}} \tag{2.24}$$

Our objective is therefore to minimize $E_k^{(1)}$ given $\sum_{i=1}^{k} s_i = n$. We expand the admissible values of s_i to the set of positive real numbers so that $s_i \in \mathbb{R}^+$. In this way, $\binom{s_i}{k}$ is defined as $\binom{s_i}{k} = \frac{s_i(s_i-1)\ldots(s_i-k+1)}{k!}$. We prove the following result.

Theorem 2.6.1 *Given that* $\sum_{i=1}^{k} s_i = n$,

$$\frac{\sum_{i=1}^{k} \binom{s_i}{k} s_i}{\sum_{i=1}^{k} \binom{s_i}{k}} \geq \frac{n}{k}.$$

Proof. Without loss of generality, assume that $s_1 \leq s_2 \leq \cdots \leq s_k$. Thus, $\binom{s_1}{k} \leq \binom{s_2}{k} \leq \cdots \leq \binom{s_i}{k}$. By Chebyshev's sum inequality,

$$k \sum_{i=1}^{k} \binom{s_i}{k} s_i \geq \left(\sum_{i=1}^{k} s_i \right) \left(\sum_{i=1}^{k} \binom{s_i}{k} \right). \tag{2.25}$$

The result follows by replacing $\sum_{i=1}^{k} s_i$ with n in the above inequality (2.25). □

Hence, $E_k^{(1)} \geq \frac{n}{k}$ and it is straightforward to verify that equality holds if $s_1 = s_2 = \cdots = s_k$. Therefore, the method of partitioning n into blocks of equal sizes at each phase as described in the proof of the Proposition 2.5.1 indeed minimizes the running time.

Pseudo Solutions Deriving from Multiple Blocks

Denote s_i to be the size of block i, $1 \leq i \leq k$. so that $\sum_{i=1}^{k} s_i = n$. Let $p_{i_1,i_2,\ldots,i_m,k}$ be the conditional probability that the k output solutions derive from blocks $i_1, i_2, \ldots, i_m$ given that the k output solutions derive from m distinct blocks. Then, $p_{i_1,i_2,\ldots,i_m,k}$ satisfies the following.

$$p_{i_1,i_2,\ldots,i_m,k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \sum_{k_1+\cdots+k_m=k} \binom{s_{i_1}}{k_1}\cdots\binom{s_{i_m}}{k_m} = \sum_{k_1+\cdots+k_m=k} \binom{s_{i_1}}{k_1}\cdots\binom{s_{i_m}}{k_m}$$

A more concise representation can be expressed as follows.

$$p_{i_1,i_2,\ldots,i_m,k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \sum_{k_1+\cdots+k_m=k} \prod_{t=1}^{m} \binom{s_{i_t}}{k_t} = \sum_{k_1+\cdots+k_m=k} \prod_{t=1}^{m} \binom{s_{i_t}}{k_t} \tag{2.26}$$

We can further simplify the above expression by the following lemma.

Lemma 2.6.1

$$\sum_{k_1+\cdots+k_m=k} \prod_{t=1}^{m} \binom{s_{i_t}}{k_t} = \binom{s_{i_1}+\cdots+s_{i_m}}{k}$$

Proof. For brevity, denote $s = s_{i_1} + \cdots + s_{i_m}$.

Consider the polynomial $(1+x)^s$. By the binomial theorem,

$$(1+x)^s = \sum_{r=0}^{s} \binom{s_{i_1}+\cdots+s_{i_m}}{r} x^r. \tag{2.27}$$

On the other hand,

$$(1+x)^s = (1+x)^{s_{i_1}} \ldots (1+x)^{s_{i_m}} = \prod_{t=1}^{m} \sum_{r_t=0}^{s_{i_t}} \binom{s_{i_t}}{r_t} x^{r_t}. \tag{2.28}$$

In this instance, the coefficient of x^r is the sum of all products of binomial coefficients of the form $\binom{s_{i_t}}{r_t}$ where the r_t sum to r. Therefore,

$$\prod_{t=1}^{m} \sum_{r_t=0}^{s_{i_t}} \binom{s_{i_t}}{r_t} x^{r_t} = \sum_{r=0}^{s} \sum_{r_1+\ldots r_m=r} \prod_{t=1}^{m} \binom{s_{i_t}}{r_t} x^r. \tag{2.29}$$

Hence,

$$\sum_{r=0}^{s} \binom{s_{i_1}+\cdots+s_{i_m}}{r} x^r = \sum_{r=0}^{s} \sum_{r_1+\ldots r_m=r} \prod_{t=1}^{m} \binom{s_{i_t}}{r_t} x^r. \tag{2.30}$$

The result follows by equating the coefficients of x^r on both sides of the above equation (2.30). □

Corollary 2.6.1

$$p_{i_1,i_2,\ldots,i_m,k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k} = \binom{s_{i_1}+\cdots+s_{i_m}}{k}$$

Proof. Follows directly from Equation (2.26) and Lemma 2.6.1 □

Suppose m solutions are derived from blocks $i_1,\ldots,i_m$. Upon discarding blocks $i_1,\ldots,i_m$, $s_{i_1}+\cdots+s_{i_m}$ of $r_{i,j}$ have to be randomly chosen to fill the remaining blocks so that they sum back up to n. Denote $E_k^{(m)}$ to be the expected number of randomly chosen $r_{i,j}$ required. Then, a generalized form of $E_k^{(m)}$ can be expressed as follows.

$$E_k^{(m)} = \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} p_{i_1,i_2,\ldots,i_m,k}(s_{i_1}+\cdots+s_{i_m}) \tag{2.31}$$

From the result of Corollary 2.6.1, this implies that $E_k^{(m)}$ satisfies

$$E_k^{(m)} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k} = \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k}(s_{i_1}+\cdots+s_{i_m}).$$

Once again, we seek to maximize $E_k^{(m)}$ given $\sum_{i=1}^k s_i = n$. As before, we expand the admissible values of s_i to the set of positive real numbers so that $s_i \in \mathbb{R}^+$. In this way, $\binom{s_i}{k}$ is defined as $\binom{s_i}{k} = \frac{s_i(s_i-1)\ldots(s_i-k+1)}{k!}$. We prove the following result.

Theorem 2.6.2 *Given that* $\sum_{i=1}^k s_i = n$,

$$\sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k}(s_{i_1}+\cdots+s_{i_m}) \geq \frac{mn}{k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k}.$$

Proof. By Chebyshev's sum inequality,

$$\binom{k}{m} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k}(s_{i_1}+\cdots+s_{i_m})$$

$$\geq \left(\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} \right) \left(\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} (s_{i_1} + \cdots + s_{i_m}) \right).$$

Since $\sum_{i=1}^{k} s_i = n$,

$$\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} (s_{i_1} + \cdots + s_{i_m}) = \binom{k-1}{m-1} n. \tag{2.32}$$

Hence, we obtain

$$\binom{k}{m} \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} (s_{i_1} + \cdots + s_{i_m})$$
$$\geq \binom{k-1}{m-1} n \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k}.$$

By elementary algebraic operations, it is straightforward to verify that

$$\frac{\binom{k-1}{m-1}}{\binom{k}{m}} = \frac{m}{k} \tag{2.33}$$

from which the result follows. □

Hence, $E_k^{(m)} \geq \frac{mn}{k}$ and it is straightforward to verify that equality holds if $s_1 = s_2 = \cdots = s_k$. Therefore, the method of partitioning n into blocks of equal sizes at each phase as described in the proof of the Proposition 2.5.2 indeed minimizes the running time.

2.7 Applications in Other MDLP Settings

We demonstrate how the block method can be adapted to obtain bounds in other generalized multiple discrete logarithm settings. We consider applications to the $(e_1, \dots, e_d)$-Multiple Discrete Logarithm Problem with Auxiliary Inputs (MDLPwAI) as well as the $\mathbb{F}_p$-Multiple Discrete Logarithm Problem in the Exponent ($\mathbb{F}_p$-MDLPX). Let $G = \langle g \rangle$ be a cyclic group of large prime order p. Their formal definitions are as follow.

Definition 2.7.1 (MDLPwAI). *Given G, g, p, e_i and $g^{{x_i}^{e_1}}, g^{{x_i}^{e_2}}, \dots g^{{x_i}^{e_d}}$ $\forall\ 1 \le i \le k$, find non-negative integers $x_1, x_2, \dots, x_k \in \mathbb{Z}_p$.*

Definition 2.7.2 ($\mathbb{F}_p$-MDLPX). *Let $\chi \in \mathbb{F}_p$ be an element of multiplicative order N. Given G, g, p, χ and $g^{\chi^{x_i}}$ $\forall i \in \mathbb{Z}^+$ such that $1 \le i \le k$, find non-negative integers $x_1, x_2, \dots, x_k \in \mathbb{Z}_N$.*

The computational complexity of MDLPwAI was analysed in [35]. In the same paper, the authors introduced the $\mathbb{F}_p$-MDLPX and also analysed its complexity.

Here, we define the Generalized Multiple Discrete Logarithm Problem (GMDLPwAI) with Auxiliary Inputs and the Generalized $\mathbb{F}_p$-Multiple Discrete Logarithm Problem in the Exponent ($\mathbb{F}_p$-GMDLPX) to be solving k out of n instances of the MDLPwAI and $\mathbb{F}_p$-MDLPX respectively. We provide the formal definitions below.

Definition 2.7.3 (GMDLPwAI). *Given G, g, p, e_i and $g^{{x_i}^{e_1}}, g^{{x_i}^{e_2}}, \dots g^{{x_i}^{e_d}}$ $\forall i \in \mathbb{Z}^+$ such that $1 \le i \le n$, find k pairs (i, x_i), $x_i \in \mathbb{Z}_p$, where $i \in S$ such that S is a k-subset of $\{1, \dots, n\}$.*

Definition 2.7.4 ($\mathbb{F}_p$-GMDLPX). *Let $\chi \in \mathbb{F}_p$ be an element of multiplicative order N. Given G, g, p, χ and $g^{\chi^{x_i}}$ $\forall i \in \mathbb{Z}^+$ such that $1 \le i \le n$, find k pairs (i, x_i), $x_i \in \mathbb{Z}_N$, where $i \in S$ such that S is a k-subset of $\{1, \dots, n\}$.*

2.7.1 Block Based GMDLPwAI

The block method can be adapted to obtain bounds for the GMDLPwAI by randomizing the input elements in the following way. Given $g^{{x_i}^{e_1}}$, select random integers $r_{i,j} \in \mathbb{Z}_p^*$ and compute values of

$$(g^{{x_i}^{e_1}})^{{r_{i,j}}^{e_1}} = g^{(r_{i,j}x_i)^{e_1}} \tag{2.34}$$

as inputs into the GMDLPwAI solver. For each $r_{i,j}$, reduce $r_{i,j}^{e_1}$ modulo p and then $g^{(r_{i,j}x_i)^{e_1}}$ can be computed within $2\log_2 p$ group operations. Repeat this procedure for all $e_2, \dots, e_d$. We show how the x_i can be recovered. For instance, suppose the solver outputs solution (l, y) corresponding to some particular input $g^{(r_{i,j}x_i)^{e_1}}$. In which case, x_i can thus be obtained by solving $r_{i,j}x_i \equiv y \bmod p$. Such congruence equations are efficiently solvable since $\gcd(r_{i,j}, p) = 1$.

Let T'_k and $T'_{k,n}$ denote the time taken in group operations for an optimal algorithm to solve the MDLPwAI and GMDLPwAI problems respectively.

Suppose $k < n < k^2$. Then by adapting the block technique applied to the GMDL problem before, it can be shown that

$$T'_k \leq (r + \frac{n}{k})T'_{k,n} + 2d(rk + nH_{\lceil \frac{n}{k} \rceil}) \log_2 p \tag{2.35}$$

where

$$r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil .$$

It has been conjectured in [35] that $T'_k = \Omega(\sqrt{kp/d})$ for values of $e_i = i$. Assuming this conjecture, we can conclude from our results that for all polynomially bounded inputs with $d = O(p^{1/3-\epsilon})$, $\epsilon > 0$, $T'_{k,n}$ is bounded by

$$T'_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k} + r} \sqrt{\frac{p}{d}} \right) \tag{2.36}$$

where $r = r(k, n) \geq 1$ is any function of k and n satisfying $r(k, n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right)$.

When $k = 1$, the above bound given in (2.36) is not applicable since $n \geq k^2$ in this situation. Nevertheless, we show how an unconditional bound for $T'_{1,n}$ can still be obtained in this specific case without the assumption of any conjecture. Similar to the generalized case, randomize input elements of the form $g^{(r_{i,j}x_i)^{e_1}}$. One of the x_i can then be computed by solving $r_{i,j}x_i \equiv y \bmod p$ where y is a given known output. The process terminates here since one of the x_i has already been obtained. Hence, we have the following inequality:

$$T'_1 \leq T'_{1,n} + 2nd \log_2 p. \tag{2.37}$$

From the results of [13], $T'_1 = \Omega(\sqrt{p/d})$. It follows that for $n = o\left(\frac{\sqrt{p}}{d^{3/2} \log p} \right)$,

$$T'_{1,n} = \Omega(\sqrt{p/d}). \tag{2.38}$$

2.7.2 Matrix Based GMDLPwAI

In this section, our objective is to find values of e_i for which the matrix method can applied to solve the GMDLPwAI. We recall from Chapter 2.3

that the validity of the method for the GMDL problem necessitates $g^{x_1+x_2+\cdots+x_k}$ to be efficiently computable from given values of $g^{x_1}, g^{x_2}, \ldots, g^{x_k}$. Moreover, the GMDLPwAI can be viewed as a generalization of the GMDL problem (i.e. the GMDLPwAI reduces to the GMDL problem when $d = 1$ and $e_1 = 1$). In a similar vein, it is required that $g^{f_i(x_1+x_2+\cdots+x_k)}$ to be efficiently computable, from the given known values of the GMDLPwAI. In the instance of GMDLPwAI, $f_i(x) = x^{e_i}$. In that regard, suppose $g^{f_{i0}(x_1+x_2+\cdots+x_k)} = g^{a_1 f_{i1}(x_1)} g^{a_2 f_{i2}(x_2)} \ldots g^{a_k f_{ik}(x_k)}\ \forall\ x_i$ and for some integer constants a_i so that it can be efficiently computed. We prove the following result.

Theorem 2.7.1 *Let $f_i(x) = x^{e_i}$, where $e_i \in \mathbb{Z}$ are not necessarily distinct. If for some integer constants a_i such that*

$$f_0(x_1 + x_2 + \cdots + x_k) \equiv a_1 f_1(x_1) + a_2 f_2(x_2) + \ldots a_k f_k(x_k) \mod p$$

for all odd primes p and for all $x_1, x_2, \ldots, x_k \in \mathbb{Z}_p$, then the only solutions are of the form $f_i = x^{c_i(p-1)+1}$ for some integer constants c_i.

Proof. For each $1 \leq i \leq k$, substitute $x_i = 1$ and $x_{i'} = 0$ for $i \neq i'$. We obtain $f_0(1) = a_i f_i(1)\ \forall\ i$. Hence, $a_i = 1\ \forall\ i$. Upon establishing that all a_i values have to be 1, we proceed as follows.

Let $x_2 = x_3 = \cdots = x_k = 0$. This implies that $f_0(x_1) \equiv f_1(x_1) \bmod p$ for all x_1.

Similarly, let $x_1 = x_3 = \cdots = x_k = 0$. This implies that $f_0(x_2) \equiv f_2(x_2) \bmod p$ for all x_2.

Continuing in this fashion, it can be deduced that $f_0(x) \equiv f_1(x) \equiv \ldots f_k(x) \bmod p\ \forall\ x$. Next, let $x_3 = x_4 = \cdots = x_k = 0$. This implies $f_0(x_1 + x_2) \equiv f_0(x_1) + f_0(x_2) \bmod p$. We claim that $f_0(x) \equiv x f_0(1) \bmod p\ \forall\ x$. It clear that the result holds true for $x = 0, 1$. By applying an inductive argument,

$$f_0(x+1) \equiv f_0(x) + f_0(1) \equiv x f_0(1) + f_0(1) \equiv (x+1) f_0(1) \mod p$$

and the claim follows. Hence, p divides $x^{e_0} - x$ for all $x \in \mathbb{Z}_p$. Since $(\mathbb{Z}/p\mathbb{Z})^\times \cong C_{p-1}$, there exists a generator of the cyclic group $x_0 \in \mathbb{Z}_p$ such that if p divides $x_0^{e_0} - x_0$, then $e_0 \equiv 1 \bmod p-1$. Moreover, since we have earlier established that $f_0(x) \equiv f_1(x) \equiv \ldots f_k(x) \bmod p$, it can be concluded that $e_i \equiv 1 \bmod p-1$ for all i. Hence, $f_i = x^{c_i(p-1)+1}$ and it is straightforward to verify that these are indeed solutions to the original congruence equation.

□

From the result of Theorem 2.7.1, if e_i is of the form $e_i = c_i(p-1)+1$, then $g^{f_i(x_1+x_2+\cdots+x_k)}$ be can efficiently computed. However,

$$g^{x^{c_i(p-1)+1}} = g^x$$

so such e_i values reduces to the classical multiple discrete logarithm problem. Therefore, the matrix method is not applicable to solve the GMDLPwAI.

2.7.3 Block Based $\mathbb{F}_p$-GMDLPX

The block method can be also adapted to obtain bounds for the $\mathbb{F}_p$-GMDLPX by randomizing the input elements with the computations

$$(g^{\chi^{x_i}})^{\chi^{r_{i,j}}} = g^{\chi^{x_i+r_{i,j}}}$$

where $r_{i,j} \in \mathbb{Z}_p$ are selected randomly. For each $r_{i,j}$, reduce $\chi^{r_{i,j}}$ modulo p and then $g^{\chi^{x_i+r_{i,j}}}$ can be computed within $2\log_2 p$ group operations.

Suppose the solver outputs solution (l, y) corresponding to some particular input $g^{\chi^{x_i+r_{i,j}}}$. In which case, x_i can thus be obtained by solving $r_{i,j} + x_i \equiv y \bmod p$. The analysis and obtained bounds in this case is similar to the classical GMDL problem which has already been discussed so we will omit the details here.

Let T''_k and $T''_{k,n}$ denote the time taken in group operations for an optimal algorithm to solve the $\mathbb{F}_p$-MDLPX and $\mathbb{F}_p$-GMDLPX problems respectively. It has been shown in [35] that T''_k can be achieved in $O(\sqrt{kN})$. If this is optimal, then our results show that

$$T''_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{N}\right)$$

where $r = r(k,n) \geq 1$ is any function of k and n satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$, subject to the conditions given in Theorem 2.5.2.

2.7.4 Matrix Based $\mathbb{F}_p$-GMDLPX

We recall that the computational Diffie-Hellman assumption (CDH assumption) states that a certain computational problem within a large cyclic group is hard. More specifically, it states the following.

CDH assumption. *Suppose G is a large cyclic group of order p. Let g be a generator element of the underlying group G and x, $y \in \{0,1,\ldots,p-1\}$ be randomly selected. Given g, g^x and g^y, it is computationally intractable to compute g^{xy}.*

In this case, the matrix method does not apply here since there is no known efficient method to compute $g^{\chi^{x_i+x_j}}$ given $g^{\chi^{x_i}}$ and $g^{\chi^{x_j}}$ if the Diffie-Hellman assumption holds.

2.8 Some Explicit Bounds of $T_{k,n}$

The conditions imposed in Theorem 2.5.2 might initially seem restrictive but in fact they are satisfied by large classes of k and n. In this section, we present some interesting explicit bounds of $T_{k,n}$ by varying n relative to k. For the remainder of this section, k can be taken to be any function satisfying $k = O(p^{1-\epsilon})$, for some $0 < \epsilon < 1$.

Suppose $n = k + c$ for some constant c, $c > 0$. It is straightforward to verify that all the conditions of Theorem 2.5.2 are satisfied. We show that $r(k, n)$ be can be taken to be a constant.

Lemma 2.8.1 *Let $r(k, n) = 2$, where $n = k + c$, $c > 0$. Then*

$$r(k, n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right).$$

Proof. The proof is straightforward. For all $k > c^2$,

$$2 > \frac{\log k}{\log k - \log c} = \frac{\log k}{\log \frac{k}{c}}. \tag{2.39}$$

Since $\log k \geq \log(\frac{k^2}{k+c})$ and $\log(\frac{k}{c}) \leq \log(\frac{k+c}{c})$, we have

$$\frac{\log k}{\log \frac{k}{c}} \geq \frac{\log(\frac{k^2}{k+c})}{\log(\frac{k+c}{c})}. \tag{2.40}$$

Thus for all $k > c^2$,

$$r(k,n) = 2 > \frac{\log(\frac{k^2}{k+c})}{\log(\frac{k+c}{c})}$$

and the result follows. □

Corollary 2.8.1

$$T_{k,k+c} = \Omega(\sqrt{kp})$$

where c is a positive constant.

Proof. It follows directly from Lemma 2.8.1 and Theorem 2.5.2 that

$$T_{k,k+c} = \Omega\left(\frac{\sqrt{k}}{\frac{k+c}{k}+2}\sqrt{p}\right) = \Omega(\sqrt{kp}).$$

□

Next, we consider $n = ck$ for some constant c, $c > 1$. Once again, it is straightforward to verify that all the conditions of Theorem 2.5.2 are satisfied. We show that $r(k,n)$ can be taken to be $\log k$.

Lemma 2.8.2 *Let $r(k,n) = \log k$, where $n = ck$, $c > 1$. Then*

$$r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right).$$

Proof. Clearly when $n = ck$,

$$\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} = (\log\frac{c}{c-1})^{-1}(\log k - \log c) \tag{2.41}$$

and the result follows. □

Corollary 2.8.2

$$T_{k,ck} = \Omega(\frac{\sqrt{k}}{\log k}\sqrt{p})$$

where c is a constant, $c > 1$.

Proof. By a direct application of Lemma 2.8.2 and Theorem 2.5.2,

$$T_{k,ck} = \Omega(\frac{\sqrt{k}}{c + \log k}\sqrt{p}) = \Omega(\frac{\sqrt{k}}{\log k}\sqrt{p}).$$

□

Let $n = k \log^c k$ for some constant c, $c > 0$. It is again straightforward to verify that all the conditions of Theorem 2.5.2 are satisfied. We show that $r(k, n)$ can be taken to be $\log^{1+c} k$. The proof here is only slightly more involved than the previous two. We first recall the Taylor series.

Taylor series. *The Taylor series of a real function f(x) that is infinitely differentiable at a real number a is the power series*

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x - a)^n.$$

When $a = 0$, the series is also known as a Maclaurin series.

Lemma 2.8.3 *Let $r(k, n) = \log^{1+c} k$, where $n = k \log^c k$, $c > 0$. Then*

$$r(k, n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right).$$

Proof.

$$\log \frac{n}{n-k} = \log \frac{\log^c k}{\log^c k - 1}.$$

Next, consider the function $\log \frac{\log^c k}{\log^c k - 1}$, for $k > e$ where e is the base of the natural logarithm. For brevity, denote $x = \log^c k$ so $x > 1$. By the Maclaurin series expansion, for $x > 1$,

$$\log \frac{x}{x-1} = -\log(1 - \frac{1}{x}) = \sum_{i=1}^{\infty} \frac{1}{ix^i} > \frac{1}{x}. \qquad (2.42)$$

In particular, this proves that when $k > e$,

$$\log \frac{\log^c k}{\log^c k - 1} > \frac{1}{\log^c k}. \tag{2.43}$$

Hence for $k > e$,

$$\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} < (\log^c k) \log(\frac{k^2}{n}) = (\log^c k) \log(\frac{k}{\log^c k}) < \log^{1+c} k. \tag{2.44}$$

Thus $r(k,n)$ can be taken to be $\log^{1+c} k$ when $n = k \log^c k$. □

Corollary 2.8.3

$$T_{k,k\log^c k} = \Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p})$$

where c is a positive constant.

Proof. From Lemma 2.8.3, we can take $r(k,n) = \log^{1+c} k$ and so

$$\frac{n}{k} + r = \log^c k + \log^{1+c} k.$$

From Theorem 2.5.2, we obtain

$$T_{k,k\log^c k} = \Omega(\frac{\sqrt{k}}{\log^c k + \log^{1+c} k}\sqrt{p}) = \Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p}).$$

□

Table 2.1 provides a summary of the results for the lower bounds of $T_{k,n}$ with different n relative to k.

2.9 Conclusion

In this chapter, we established rigorous bounds for the generic hardness of the generalized multiple discrete logarithm problem which can be regarded a generalization of the multiple discrete logarithm problem. Some explicit bounds are also computed using both the matrix method and block method. Many instances of $T_{k,n}$ are shown to be in fact asymptotically optimal. The

Table 2.1: Some bounds of $T_{k,n}$

k	n	r	$T_{k,n}$
$o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$\{n : n\sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)\}$	N.A.	$\Omega(\sqrt{kp})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$k + c,\ c \geq 0$	constant	$\Omega(\sqrt{kp})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$ck,\ c > 1$	$\log\ k$	$\Omega(\frac{\sqrt{k}}{\log k}\sqrt{p})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$k \log^c k,\ c > 0$	$\log^{1+c} k$	$\Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p})$

overall best bounds obtained here require the union of results from both of these techniques. Furthermore, we show that the block method can also be adapted to handle generalizations arising in other discrete logarithm problems. We similarly obtain bounds for these generalizations. For instance, a consequence of our result highlights that solving an instance of the MDLPwAI problem is as hard as solving the DLPwAI problem under certain conditions. We also demonstrated why the matrix method is not applicable to these and other variants of DLP.

Chapter 3

Splitting Systems for Solving the DLP with Low Hamming Weight Product Exponents

3.1 Introduction

The hardness of the discrete logarithm problem (DLP) is central in many cryptographic schemes; for instance those arising in identification schemes [2, 23, 48]. The security of the discrete logarithm problem has been understood to a large extent over generic groups of prime order p. In particular, Shoup showed that any generic algorithm for solving the DLP must perform $\Omega(\sqrt{p})$ group operations [49]. There are a few methods for computing discrete logarithm in approximately $\sqrt{p}$ operations. For example, Shanks Baby-Step-Giant-Step method computes the DLP in $\tilde{O}(\sqrt{p})$ operations. One other method is the Pollard's Rho Algorithm which can be achieved in $O(\sqrt{p})$ operations [42]. Since then, further practical improvements to the Pollard's Rho Algorithm have been proposed in [10, 15, 51] but the computational complexity remains the same.

One aspect of practical cryptosystems lies in the implementation efficiency in which exponentiations can be carried out. Agnew *et al.* proposed the usage of low Hamming weight integers as secret exponents in order to achieve faster implementations [3]. This speeds up computations since the number of multiplications for exponentiations depends on the Hamming weight of the exponent. The security of the low Hamming weight discrete logarithm problem has also been analyzed. For example, Heiman and Odlyzko first provided a method of attack [26]. Further improvements were subsequently demonstrated in [37, 50] by applying the Coppersmith's splitting system.

The GPS identification scheme is an interactive protocol between a prover and a verifier. It was introduced by Girault in [23] and later shown to be secure in [43]. This protocol is applicable for usage in low cost chips as the computational cost required by the prover is relatively low. Nevertheless, every operation incurred is still significant for low cost chips, like RFID tags. As such, Girault and Lefranc proposed in [24] for the private key exponent to be the product of two integers with low Hamming weight, thereby reducing the online computational cost. More specifically the private key was chosen such that it is a product of a 142-bit number with 16 random bits equal to 1 chosen among the 138 least significant ones and a 19-bit number with 5 random bits equal to 1 chosen among the 16 least significant ones (from which we henceforth refer to as GL parameters). In the same paper, it was computed that these parameters are not susceptible to a routine attack by exhaustive search.

Subsequent work by Coron, Lefranc and Poupard [18] demonstrated a method of attack via Coppersmith's splitting system of the GL parameters with lower complexity than routine exhaustive search. As a result, they instead proposed a different set of parameters; namely that the private key be a product of a 30-bit number with 12 nonzero bits and a 130-bit number with 26 nonzero bits (from which we henceforth refer to as CLP parameters). Moreover, they show that their line of attack is not effective against the CLP parameters.

Parameterized splitting system was first introduced by Kim and Cheon in [33]. Parameterized splitting system can be regarded as a generalization of Coppersmith's splitting system. Using this tool, they demonstrated an improved attack (with regards to speed) on the CLP parameters. They later further improve this attack over the previous work by applying a refinement [34]. Thus far, this is the current fastest known attack of the GPS identification scheme with CLP parameters.

3.1.1 Our Contributions

This work highlights general methods of solving various DLP with low Hamming weight product (LHWP) exponents by providing improved results for certain settings of the parameterized splitting system. We introduce the concept of parameters dependent splitting system which served as tools to solve such problems more efficiently. Moreover, we show that the GPS identification scheme utilizing CLP parameters satisfy such settings. There are two significant results that arise from this work. The first provides an improved attack on the GPS scheme with lower time over the most recent state of the art without any increment in memory. The second result shows for the first time that the GPS scheme can be attacked in time complexity of under 2^{64} with a slight increase in memory requirement over the former. Furthermore, we also prove that the settings required to apply our improved parameterized system are not restrictive but in fact that the set of admissible applicable values increases when the splitting sizes are further apart. More details of this property will be described in a later section. Overall, this work also serves to provide a general framework on the type of parameters suitable for consideration in future design of cryptosystems based on DLP with LHWP exponents.

3.1.2 Chapter Organization

This chapter is organized as follows. Preliminaries involving related tools required to solve the DLP with LHWP exponents will be covered in Chapter 3.2. Chapter 3.3 describes the GPS scheme and its current known attacks. We present the results of two parameters dependent splitting systems in Chapter 3.4. Chapter 3.5 describes methods to solve the DLP with LHWP exponents and implications of our results. The contents of Chapter 3.6 demonstrate improved attacks to the GPS scheme. Chapter 3.7 discusses the classes of parameters which are relevant to our analysis. A summary of results is covered in Chapter3.8. We conclude in Chapter 3.9.

3.2 Preliminaries

Let G be a group, $g \in G$ be a generator of the group and z be an integer. Denote $\mathrm{wt}(z)$ and $\mathrm{ord}(g)$ to be the Hamming weight of z and the order of g respectively.

The low Hamming weight DLP seeks solution z such that $g^z = h$ for given known G, g, $h \in G$, *ord g* and $\mathrm{wt}(z)$. The computational complexity of solving the low Hamming weight DLP has been well understood and a good exposition of known methods can be found in [50]. We denote CSS and PSS to mean Coppersmith's splitting system and parameterized splitting system respectively.

In our work, we are interested to solve the DLP with LHWP exponents which has applications to the security of the GPS identification scheme. A definition of the DLP with LHWP exponents is given as follows.

Definition 3.2.1 (DLP with LHWP exponents). *Let $z = xy$, where x, $y \in \mathbb{Z}^+$. Given G, g, ord g, $wt(x)$, $wt(y)$ and $h \in G$, find z satisfying $g^z = h$.*

The Coppersmith's splitting system was initially introduced in [37] which came about from the work of [17]. It is described as follows.

Definition 3.2.2 (Coppersmith's splitting system (CSS)). *Let n and t be even integers such that $0 < t < n$. A (N,n,t)-splitting system is a pair of (X,$\mathcal{B}$) satisfying*
1. $|X|= n$.
2. $\mathcal{B}$ is a set of $\frac{n}{2}$-subsets of X called blocks and $|\mathcal{B}|= N$.
3. For every $Y \subseteq X$ such that $|Y|= t$, $\exists$ a block $B \in \mathcal{B}$ such that $|B \cap Y|= \frac{t}{2}$.

It was shown in [50] among others that N can be taken to be $\frac{n}{2}$. This result was applied in [18] to obtain improved attacks of the GPS scheme with GL parameters.

The parameterized splitting system was first introduced in [33]. It can be also be regarded as a generalized version of Coppersmith's splitting system and is given as follows.

Definition 3.2.3 (Parameterized splitting system (PSS)). *Let n and t be integers such that $0 < t < n$. For any t_s such that $1 \leq t_s \leq t$. A (N,n,t,t_s)-parameterized splitting system is a pair of (X,$\mathcal{B}$) satisfying*
1. $|X|= n$.
2. $\mathcal{B} = \{B \subset X : |B|= \lfloor \frac{t_s n}{t} \rfloor\}$ is a set of $\lfloor \frac{t_s n}{t} \rfloor$-subsets of X called blocks and $|\mathcal{B}|= N$.
3. For every $Y \subseteq X$ such that $|Y|= t$, $\exists$ a block $B \in \mathcal{B}$ such that $|B \cap Y|= t_s$.

In particular, when $t_s = \frac{t}{2}$, the parameterized splitting system corresponds to the Coppersmith's splitting system.

3.3 The GPS Scheme and Related Work

In this section, we outline the GPS scheme and briefly describe related work pertaining to solving the LHWP DLP.

3.3.1 The GPS identification scheme

The GPS identification scheme was submitted to the NESSIE project [25] and is based on the DLP modulo a large composite. The details of the scheme are as follows:

- N is the product of two sufficiently large prime integers so that factoring N is computationally infeasible,

- $g \in \mathbb{Z}_N^*$ (the multiplicative group of invertible elements in $\mathbb{Z}_N$) is of maximal order m,

- z is the private key of the prover and the associated public keys are $(N, g, h = g^{-z})$.

The security parameters are given by:

- S refers to the binary size of the private key z; $S = 160$ typically.

- k refers to the binary size of the challenges sent to the provers. It also corresponds to the security level of the scheme.

- R refers to the binary size of the exponents in the commitment computation and typically $R = S + k + 80$.

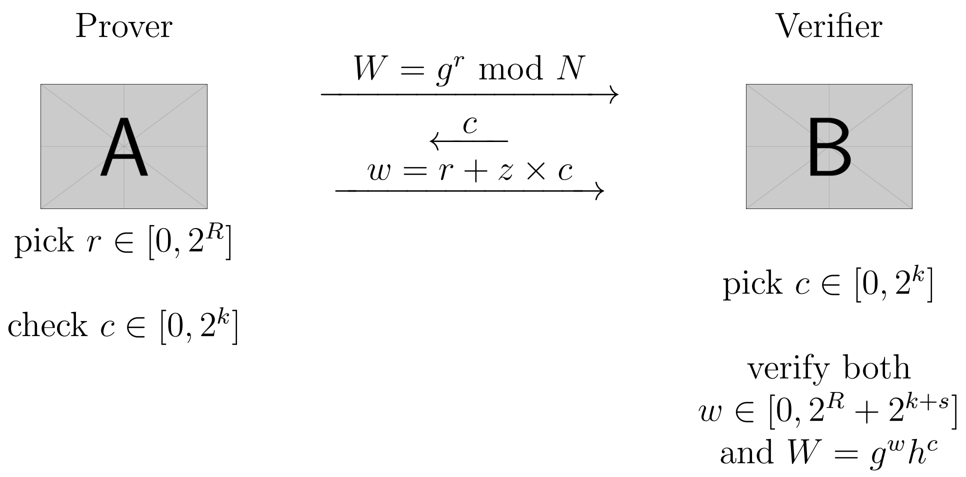

Figure 3.1*: GPS Scheme

Figure 3.1* provides an overview of the interactive protocol between a prover and verifier of the GPS identification scheme. It was subsequently proposed in [24] for the private key to be a product of two low Hamming weight integers to speed up the online computation.

3.3.2 Related Work

We briefly cover the known methods to solve the DLP with LHWP exponents. Let z be a product of two elements $x \in X$ and $y \in Y$, where $X, Y \in \mathbb{Z}_m$.

One straightforward method proceeds in the following manner. Recall that $h = g^z = g^{xy}$, for some given $\mathrm{wt}(x)$ and $\mathrm{wt}(y)$. This can be rewritten as $g^x = h^{y^{-1}}$. The meet-in-the-middle technique can then applied to this equation by computing all possible values of g^x and $h^{y^{-1}}$ respectively until equality between the two is achieved. The complexity is thus

$$O(|X| + |Y|)$$

and the memory requirement is

$$O(\min(|X|, |Y|)).$$

Method of Coron *et al.*

A method in [18] considers the equation $h = g^z = g^{xy}$ to be expressed as

$$h = g^{xy} = r^y$$

where $r = g^x$. Without loss of generality, suppose that $|X| > |Y|$. For every y_i such that $\mathrm{wt}(y_i) = \mathrm{wt}(y)$, apply the method of Coppersmith splitting system. Hence, there must exist y_i such that the x value obtained via this splitting system will result equality in the equation. Let l and t denote the binary size and Hamming weight of x respectively. Thus, the computational complexity of solving for z is given by

$$O\left(|Y| \times l \binom{l/2}{t/2}\right)$$

and the space requirement is given by

$$O\left(\binom{l/2}{t/2}\right).$$

Method of Kim-Cheon

The results in [33] and [34] utilize the parameterized splitting system in solving the DLP with LHWP exponents. Without loss of generality, suppose that $|X| > |Y|$. Then split $x = u + v$ for $u \in U$, $v \in V$ and where U and V are disjoint subsets of $\mathbb{Z}_m$ such that $X \subset U + V = \{u + v : u \in U, v \in V\}$. By considering the following equation

$$h^{y^{-1}} g^{-u} = g^{v}$$

the parameterized splitting system from Definition 3.2.3 can be applied. Let $\mathrm{wt}(x) = t$, $\mathrm{wt}(u) = t_s$ and $\mathrm{wt}(v) = t - t_s$, where $t_s \leq \lceil \frac{t}{2} \rceil$. Since, a result from [33] shows that n blocks are required, the resulting time complexity is given by

$$O\left(n\left(|Y|\binom{n_s}{t_s} + \binom{n-n_s}{t-t_s}\right)\right)$$

and memory space requirement of

$$O\left(\min\left\{|Y|\binom{n_s}{t_s}, \binom{n-n_s}{t-t_s}\right\}\right).$$

A refinement from [12] shows that time and space complexity can be further reduced to

$$O\left(n\left(|Y|\binom{n_s-1}{t_s-1} + \binom{n-n_s}{t-t_s}\right)\right)$$

and

$$O\left(\min\left\{|Y|\binom{n_s-1}{t_s-1}, \binom{n-n_s}{t-t_s}\right\}\right)$$

respectively.

3.4 Improved Results on the Parameterized Splitting System

It was shown in [33] and [34] that the parameterized splitting system requires n blocks. We prove that for numerous classes of parameters, the number of blocks required in the parameterized splitting system is less than n. In particular, we show that the GPS scheme with CLP parameters is among those arising in such situations. As a result, we obtain a lower complexity attack on the GPS scheme.

We first provide a slight reformulation of the parameterized splitting system in order to distinguish and make a comparison of the number of blocks required of the splitting system. Let t_1, t_2, n_1, $n_2 \in \mathbb{Z}^+$ such that $t_1 \leq t_2$, $n_1 \leq n_2$, $t = t_1 + t_2$, $n = n_1 + n_2$ and $n_i = \frac{nt_i}{t}$ for $i = \{1, 2\}$. With these added notations, we introduce the parameters dependent splitting system or PDSS as follows.

Definition 3.2.3* (Parameters dependent splitting system (PDSS)). *A (N, n_1, n_2, t_1, t_2)-parameterized splitting system is a pair of $(X, \mathcal{B})$ satisfying*
1. $|X| = n = n_1 + n_2$.
2. $\mathcal{B} = \{B \subset X : |B| = \frac{t_2 n}{t} = n_2\}$ *is a set of* n_2*-subsets of* X *called blocks and* $|\mathcal{B}| = N$.
3. For every $Y \subseteq X$ *such that* $|Y| = t$, $\exists$ *a block* $B \in \mathcal{B}$ *such that* $|B \cap Y| = t_2$.

Remark. t_2, n_2 can essentially be swapped with corresponding t_1, n_1 by considering the complement. For example, let $X \subseteq \mathbb{Z}_m$ so that $X = \{x = \sum_{i=0}^{n-1} x_j 2^j : x_j = 0 \text{ or } 1, \mathrm{wt}(x) = t\}$. If $|B \cap Y| = t_2$ then $|(\mathbb{Z}_m \backslash B) \cap Y| = t_1$.

Theorem 3.4.1. *Let* $n_1 > \frac{n_2}{2}$. *For any* n_1, n_2, *let* $k \in \mathbb{Z}^+ \backslash \{1\}$ *be the integer satisfying*

$$\frac{k+1}{2k+1} n_2 \leq n_1 < \frac{k}{2k-1} n_2.$$

Suppose t_1 *and* t_2 *satisfy the following:*

$$\frac{2k+1}{k+1} t_1 - \frac{3k+1}{k+1} \leq t_2 \leq \frac{2k-1}{k} t_1 + \frac{3k-2}{k},$$

$$\frac{2k-1}{k} t_1 < t_2 \leq \frac{2k+1}{k+1} t_1.$$

Then $N = 2n_2 - n_1 + 1$ *suffices to generate a parameterized splitting system.*

Proof. Let $X = \mathbb{Z}_n$ and define

$$A_i = \{i + j \bmod n : 0 \leq j < n_1\}$$

$$B_i = \{i + j \bmod n : 0 \leq j < n_2\}$$

$$C_{i,j} = \{i \bmod n, i + 1 \bmod n \ldots, j - 1 \bmod n\}$$

Fix any subset $Y \subseteq X$ such that $|Y|= t$. From the construction of B_i, it is clear that for all i,

$$|B_i \cap Y|-|B_{i+1} \cap Y|= \{-1, 0, 1\}. \tag{3.1}$$

From the above (3.1), we say that $|B_i \cap Y|$ exhibits discrete continuity and refer to it as Property 1. Let

$$\mathcal{B} = \{B_i : 0 \leq i \leq 2n_2 - n_1\}.$$

For brevity we denote $[a, b)$ to mean the set of integers $\{a, a + 1, \ldots, b - 1\}$.

Now, suppose that $|B_i \cap Y|\geq t_2+1, \forall i \in [0, 2n_2-n_1+1)$. In particular, this implies that $|B_0 \cap Y|\geq t_2+1$ and $|B_{n_2} \cap Y|\geq t_2+1$. Therefore correspondingly, $|A_{n_2} \cap Y|\leq t_1-1$ and $|A_{n_2-n_1} \cap Y|\leq t_1-1$. Thus, we obtain $|(A_{n_2} \cup A_{n_2-n_1}) \cap Y|\leq 2t_1 - 2$. This in turn implies that $|C_{0,n_2-n_1} \cap Y|\geq t_2 - t_1 + 2$. Also, $|B_{n_1} \cap Y|\geq t_2 + 1$ and hence $|A_0 \cap Y|\leq t_1 - 1$. Combining these results, we obtain

$$|C_{n_2-n_1,n_1} \cap Y|\leq 2t_1 - t_2 - 3. \tag{3.2}$$

Moreover, $|B_{n_2-n_1} \cap Y|\geq t_2 + 1$ and so $|(C_{0,n_2-n_1} \cup C_{2n_2-n_1,n}) \cap Y|\leq t_1 - 1$. Hence,

$$|(C_{0,n_1} \cup C_{2n_2-n_1,n}) \cap Y|\leq 3t_1 - t_2 - 4. \tag{3.3}$$

We proceed to build up the length of blocks as follows. For $3n_1 < 2n_2$, $|B_{2n_1} \cap Y|\geq t_2 + 1$ and $|B_{2n_1-n_2} \cap Y|\geq t_2 + 1$. Therefore correspondingly, $|A_{n_1} \cap Y|\leq t_1 - 1$ and $|A_{2n_1} \cap Y|\leq t_1 - 1$. Thus, we obtain

$$|(C_{0,2n_1-n_2} \cup A_{n_1,n}) \cap Y|\leq 2t_1 - 2. \tag{3.4}$$

This implies that

$$|C_{2n_1-n_2,n_1} \cap Y|\geq t_2 - t_1 + 2. \tag{3.5}$$

Also, $|B_{3n_1-n_2} \cap Y|\geq t_2 + 1$ and so $|A_{2n_1-n_2} \cap Y|\leq t_1 - 1$. Combining this with the result of (3.5), we obtain

$$|C_{n_1,3n_1-n_2} \cap Y|\leq 2t_1 - t_2 - 3. \tag{3.6}$$

Together with the result of (3.3), it follows that

$$|(C_{0,3n_1-n_2} \cup C_{2n_2-n_1,n}) \cap Y|\leq 5t_1 - 2t_2 - 7. \tag{3.7}$$

Moreover, if $|(C_{0,n_1} \cup C_{2n_2-n_1,n})| \geq n_2$ i.e. $5n_1 \geq 3n_2$ and $5t_1 - 2t_2 - 7 \leq t_2$ i.e. $5t_1 \leq 3t_2 + 7$, then from Property 1, it can be concluded that there exists some $i' \in [0, 2n_2 - n_1 + 1)$ such that

$$|B_{i'} \cap Y| = t_2.$$

Next, suppose that $|B_i \cap Y| \leq t_2 - 1, \forall i \in [0, 2n_2 - n_1 + 1)$. In particular, this implies that $|B_0 \cap Y| \leq t_2 - 1$ and $|B_{n_2} \cap Y| \leq t_2 - 1$. Therefore correspondingly, $|A_{n_2} \cap Y| \geq t_1 + 1$ and $|A_{n_2-n_1} \cap Y| \geq t_1 + 1$. Thus, we obtain $|(A_{n_2} \cup A_{n_2-n_1}) \cap Y| \geq 2t_1 + 2$. This in turn implies that

$$|C_{0,n_2-n_1} \cap Y| \leq t_2 - t_1 - 2. \tag{3.8}$$

Also, $|B_{n_1} \cap Y| \leq t_2 - 1$ and hence $|A_0 \cap Y| \geq t_1 + 1$. Combining this with the result of (3.8), we obtain

$$|C_{n_2-n_1,n_1} \cap Y| \geq 2t_1 - t_2 + 3. \tag{3.9}$$

Moreover, $|B_{n_2-n_1} \cap Y| \leq t_2 - 1$ and so $|(C_{0,n_2-n_1} \cup C_{2n_2-n_1,n}) \cap Y| \geq t_1 + 1$. Hence,

$$|(C_{0,n_1} \cup C_{2n_2-n_1,n}) \cap Y| \geq 3t_1 - t_2 + 4. \tag{3.10}$$

If $3t_1 - t_2 + 4 \geq t_2$ i.e. $3t_1 \geq 2t_2 - 4$ and $|(C_{0,n_1} \cup C_{2n_2-n_1,n})| < n_2$ i.e. $3n_1 < 2n_2$, then from property 1, it can be concluded that there exists some $i' \in [0, 2n_2 - n_1 + 1)$ such that

$$|B_{i'} \cap Y| = t_2.$$

Combining all of the above results, it follows that if $3n_1 < 2n_2$, $5n_1 \geq 3n_2$, $5t_1 \leq 3t_2 + 7$ and $3t_1 \geq 2t_2 - 4$ then there exists some $i' \in [0, 2n_2 - n_1 + 1)$ such that $|B_{i'} \cap Y| = t_2$. Recall that $n_i = \frac{nt_i}{t}$ for $i = \{1, 2\}$. Therefore, the bounds involving n_1 and n_2 impose additional conditions involving t_1 and t_2. More specifically, $3n_1 < 2n_2 \implies 3t_1 < 2t_2$ and $5n_1 \geq 3n_2 \implies 5t_1 \geq 3t_2$.

As a result, it can be concluded that if

$$\frac{3}{5}n_2 \leq n_1 < \frac{2}{3}n_2$$

and that t_1, t_2 satisfy

$$\frac{5}{3}t_1 - \frac{7}{3} \leq t_2 \leq \frac{3}{2}t_1 + 2,$$

$$\frac{3}{2}t_1 < t_2 \leq \frac{5}{3}t_1,$$

then there exists a $i' \in [0, 2n_2 - n_1 + 1)$ such that

$$|B_{i'} \cap Y| = t_2$$

This implies that $\mathcal{B}$ is the required set to form the splitting system and that $N = |\mathcal{B}| = 2n_2 - n_1 + 1$. This proves the case $k = 2$. Higher orders of k can be similarly derived by an inductive argument. □

Recall that $N = n = n_1 + n_2$ was obtained in [14, 12]. Since $n_1 > \frac{n_2}{2}$, $2n_2 - n_1 + 1 \leq n = n_1 + n_2$, we derive a parameterized splitting system which requires a fewer number of blocks.

In situations where $t_2 \approx 2t_1$, we show that Theorem 3.4.1 can be further improved. In particular, we present the following result.

Theorem 3.4.2. *Let $\frac{1}{2}n_2 \leq n_1 \leq \frac{2}{3}n_2$. Suppose $t_2 = 2t_1, 2t_1 - 1$ or $2t_1 - 2$. Then*

$$N = \begin{cases} 2n_1\lceil\frac{1}{3}(t_1 - 1)\rceil - n_2(\lceil\frac{1}{3}(t_1 - 1)\rceil - 1) + 1, & \textit{if } t_2 = 2t_1 \\ 2n_1\lceil\frac{1}{4}(t_1 - 2)\rceil - n_2(\lceil\frac{1}{4}(t_1 - 2)\rceil - 1) + 1, & \textit{if } t_2 = 2t_1 - 1 \\ 2n_1\lceil\frac{1}{5}(t_1 - 3\rceil - n_2(\lceil\frac{1}{5}(t_1 - 3)\rceil - 1) + 1, & \textit{if } t_2 = 2t_1 - 2 \end{cases}$$

suffices to generate a parameterized splitting system.

Proof. Let $X = \mathbb{Z}_n$ and define

$$B_i = \{i + j \bmod n : 0 \leq j < n_2\}$$

$$C_{i,j} = \{i \bmod n, i + 1 \bmod n \ldots, j - 1 \bmod n\}$$

Fix any subset $Y \subseteq X$ such that $|Y| = t$. Let

$$\mathcal{B}_k = \{B_i : 0 \leq i \leq 2kn_1 - (k-1)n_2\}.$$

For brevity, denote $[a, b)$ to mean the set of integers $\{a, a + 1, \ldots, b - 1\}$.

Suppose that $|B_i \cap Y| \geq t_2 + 1$, $\forall i \in [0, 2kn_1 - (k - 1)n_2 + 1)$. In particular, this implies that $|B_0 \cap Y| \geq t_2 + 1$ and $|B_{n_2} \cap Y| \geq t_2 + 1$. Therefore correspondingly, $|C_{n_2,n} \cap Y| \leq t_1 - 1$ and $|C_{n_2-n_1,n_2} \cap Y| \leq t_1 - 1$. Thus, we obtain

$$|C_{n_2-n_1,n} \cap Y| \leq 2t_1 - 2. \tag{3.11}$$

Furthermore, $|B_{n_1} \cap Y| \geq t_2 + 1$. Since $t_2 \geq 2t_1 - 2$, it follows that

$$|B_{n_1} \cap Y| > |C_{n_2-n_1,n} \cap Y|. \tag{3.12}$$

However, this implies that $n_1 > n_2 - n_1$, a contradiction.

On the other hand, suppose that $|B_i \cap Y| \leq t_2 - 1$, $\forall i \in [0, 2kn_1 - (k-1)n_2+1)$. In particular, this implies that $|B_0 \cap Y| \leq t_2 - 1$ and $|B_{n_2} \cap Y| \leq t_2 - 1$. Therefore correspondingly, $|C_{n_2,n} \cap Y| \geq t_1 + 1$ and $|C_{n_2-n_1,n_2} \cap Y| \geq t_1 + 1$. Thus, we obtain

$$|C_{n_2-n_1,n} \cap Y| \geq 2t_1 + 2. \tag{3.13}$$

Furthermore, $|B_{n_1} \cap Y| \leq t_2 - 1$. Hence,

$$|C_{n_2-n_1,n_1} \cap Y| \geq (2t_1 + 2) - (t_2 - 1) = 2t_1 - t_2 + 3. \tag{3.14}$$

We proceed to build the length of blocks in following way. We have that $|C_{2n_1,n_1} \cap Y| \leq t_2 - 1$. Thus, $|C_{n_1,2n_1} \cap Y| \geq t_1 + 1$. Hence,

$$|C_{n_2-n_1,2n_1} \cap Y| \geq t_1 + 1 + (2t_1 - t_2 + 3) = 3t_1 - t_2 + 4. \tag{3.15}$$

If $3t_1 - t_2 + 4 \geq t_2$ and $2n_1 - (n_2 - n_1) \leq n_2$, then there exists some $i' \in [0, 2n_1 + 1)$ such that $|B_{i'} \cap Y| = t_2$. Otherwise, continue with the building of blocks as follows. $|B_{2n_1} \cap Y| \leq t_2 - 1$ and $|B_{2n_1-n_2} \cap Y| \leq t_2 - 1$. Therefore correspondingly, $|C_{n_1,2n_1} \cap Y| \geq t_1 + 1$ and $|C_{2n_1,2n_1-n_2} \cap Y| \geq t_1 + 1$. Thus, we obtain

$$|C_{n_1,2n_1-n_2} \cap Y| \geq 2t_1 + 2. \tag{3.16}$$

Furthermore, $|B_{3n_1-n_2} \cap Y| \leq t_2 - 1$. Hence,

$$|C_{n_1,3n_1-n_2} \cap Y| \geq (2t_1 + 2) - (t_2 - 1) = 2t_1 - t_2 + 3. \tag{3.17}$$

Therefore, it follows that

$$|C_{n_2-n_1,3n_1-n_2} \cap Y| \geq 2(2t_1 - t_2 + 3). \tag{3.18}$$

Also, since $|B_{4n_1-n_2} \cap Y| \leq t_2 - 1$, it follows that $|C_{3n_1-n_2,4n_1-n_2} \cap Y| \geq t_1 + 1$. Combining this with the result of (3.18), we obtain

$$|C_{n_2-n_1,4n_1-n_2} \cap Y| \geq 5t_1 - 2t_2 + 7. \tag{3.19}$$

If $5t_1 - 2t_2 + 7 \geq t_2$ and $(4n_1 - n_2) - (n_2 - n_1) \leq n_2$, then there exists some $i' \in [0, 4n_1 - n_2 + 1)$ such that $|B_{i'} \cap Y| = t_2$. By proceeding in an inductive fashion we have shown that if t_1, t_2 satisfy $t_1(2k+1) \leq t_2(k+1)$ and $(t_1 + 1) + k(2t_1 - t_2 + 3) \geq t_2$ for some $k \in \mathbb{Z}^+$, then

$$N = 2kn_1 - n_2(k-1) + 1$$

suffices to generate a splitting system.

The final part of the proof involves evaluating the bounds for k under the above conditions for each of $t_2 = 2t_1, 2t_1 - 1$ and $2t_1 - 2$. Since $2kn_1 - n_2(k-1)$ is an increasing function of k, ideal results are obtained when k is as low as possible. When $t_2 = 2t_1$, $k \geq \frac{t_1 - 1}{3}$ so we can take $k = \lceil \frac{1}{3}(t_1 - 1) \rceil$. When $t_2 = 2t_1 - 1$, $k \geq \frac{t_1 - 2}{4}$ so we can take $k = \lceil \frac{1}{4}(t_1 - 2) \rceil$. When $t_2 = 2t_1 - 2$, $k \geq \frac{t_1 - 3}{5}$ so we can take $k = \lceil \frac{1}{5}(t_1 - 3) \rceil$. This completes the proof of Theorem 3.4.2. $\square$

When $t_2 \approx 2t_1$, the results of Theorem 3.4.2 show that fewer blocks than those obtained in Theorem 3.4.1 suffice.

3.5 Solving the DLP with LHWP Exponents and Applications

Upon obtaining improvements to the parameterized splitting system, the subsequent approach of solving the DLP with LHWP exponents is simply a meet-in the-middle technique. We provide an outline of it below.

The DLP with LHWP exponents seeks the solution z satisfying $h = g^z = g^{xy}$ given $\mathrm{wt}(x)$ and $\mathrm{wt}(y)$. Without loss of generality, suppose that $|X| > |Y|$. Then split $x = u + v$ for $u \in U$, $v \in V$ and where U and V are disjoint subsets of $\mathbb{Z}_m$ such that $X \subset U + V = \{u + v : u \in U, v \in V\}$. Denote n to be the maximum binary size of an element X. Hence

$$X = \{x = \sum_{j=0}^{n-1} x_j 2^{n-1-j} : x_j = 0 \textit{ or } 1, \mathrm{wt}(x) = t\}.$$

Now, by considering elements of X in their binary representations, say of the form $x_0 x_1 \ldots x_{n-1}$ where each $x_i = 0$ or 1, express them in a more concise form based off their indices via A_i and B_i where $A_i = \{i + j \bmod n : 0 \leq j < n_1\}$

and $B_i = \{i + j \bmod n : 0 \leq j < n_2\}$. For example $A_0 = \{0, 1, \dots, n_1 - 1\}$ and this represents $x_0 x_1 \dots x_{n_1-1} 0 \dots 0$.

Express $h = g^{xy}$ as

$$h^{y^{-1}} g^{-u} = g^v.$$

The method proceeds by computing and storing all the values of the left-hand side followed by computing each value of the right-hand side and check if it is in the list from the first part.

Let $t' \in \{1, 2, \dots, \lceil \frac{t}{2} \rceil \ : \frac{nt'}{t} \in \mathbb{Z}\}$ be the value of t_1 that minimizes

$$|Y| \binom{n_1}{t_1} + \binom{n - n_1}{t - t_1}.$$

If t' and the corresponding $t - t'$ satisfy the conditions stated in Theorem 4.4.1 for the given k, then the computational complexity of solving the problem is given by

$$O\left((2n_2 - n_1 + 1) \left(|Y| \binom{n'}{t'} + \binom{n - n'}{t - t'} \right) \right)$$

where $n' = \frac{nt'}{t}$. The corresponding memory requirement is given by

$$O\left(\min \left\{ |Y| \binom{n'}{t'}, \binom{n - n'}{t - t'} \right\} \right).$$

This already provides a lower computational complexity over the results in [14] without any additional memory requirements via PDSS.

Improved results over [33] were subsequently obtained in [34]. Fix any subset $T \subseteq X$ such that $|T| = t$. The main improvement derives by noting that among all blocks B_i such that $|B_i \cap T| = t_2$, there exists block $B_{i'}$ such that $x_{i'} = 1$ and block $B_{i''}$ such that $x_{i'' + n_2 - 1 \bmod n} = 1$. A similar property holds for A_i. However, this is not true in our PDSS. In the case of PDSS, there either exists block $B_{i'}$ such that $x_{i'} = 1$ and block $B_{i''}$ such that $x_{i'' + n_2 - 1 \bmod n} = 1$. From the symmetry of t_1 and t_2 in PDSS, a similar property also holds for A_i. Crucially, there is no way to determine which of the two stated block properties will occur (or both). As such, a direct application of the PDSS as shown above while it yields improvements to [33], does not provide better results over [34]. Nevertheless, with some delicate refinements, we show how the results and properties of PDSS can be utilized to obtain two improved results over the existing state of the art. First, we present the following lemma.

Lemma 3.5.1. *Suppose for inputs t_i and n_i there exists some $s < n-1$ such that for all $0 \leq i \leq s$ satisfying $|B_i \cap T|= t_2$, we have that $x_{i+n_2-1 \ mod\ n} = 0$. Then there exists some m satisfying $s < m \leq n-1$ such that $x_m = 0$ and $x_{m+n_2-1 \ mod\ n} = 1$.*

Proof. From Kim-Cheon's results of the PSS property that there must exist some i, $0 \leq i \leq n-1$ satisfying $|B_i \cap T|= t_2$, $x_{i+n_2-1 \text{ mod } n} = 1$, it can be concluded from the assumption of the lemma that there must exist some m satisfying $s < m \leq n-1$ such that $x_{m+n_2-1 \text{ mod } n} = 1$. Thus it suffices to show that at least one such exist m which must also satisfy the condition that $x_m = 0$. Let M be the set of all possible values of m satisfying $s < m \leq n-1$ such that $x_{m+n_2-1 \text{ mod } n} = 1$. We know that M is non empty from the above argument. Let m' be the largest element of M. If $x_{m'} = 0$, we are done. Otherwise, suppose (for a contradiction) that $x_{m'} = 1$. Since m' is the largest element in M, $|B_j \cap Y|< t_2$ for all j such that $m' < j \leq n-1$. Let $r \leq k$ be the minimum value satisfying $|B_r \cap Y|= t_2$. Since $|B_i \cap T|-|B_{i+1} \cap T|= \{-1, 0, 1\}$, it follows that $x_{r+n_2-1 \text{ mod } n} = 1$, a contradiction from the assumption of the lemma. $\square$

With the results of Theorem 3.4.1, Theorem 3.4.2 and Lemma 3.5.1, we provide two possible ways via PDSS (dependent on parameters) to solve the DLP with low Hamming weight exponents more efficiently. We will also show in the subsequent section on how they can be applied to analyze or attack the GPS identification scheme utilizing CLP parameters.

From the result of Lemma 3.5.1, we can refine the earlier procedure of solving the DLP with LHWP exponent by computing g^v (or g^{-u}) where during the computation of all possible elements with hamming weight of t_2 in B_i, some bits are redundant and can be removed from the computations. More specifically for $i \leq s$, we first proceed with the assumption that one of the bits at the end edge of a block is 1. Hence, this requires a total of

$$(s+1)\binom{n_2-1}{t_2-1}$$

computations. If the above does not yield a valid solution, then for $i > s$, we can deduce from Lemma 3.5.1 that one of the bits at the initial edge of a block can be taken to be 0 and one bit at the end edge within this same block can be taken to be 1. This requires

$$(n-s)\binom{n_2-2}{t_2-1}$$

computations. Lemma 3.5.1 ensures that ignoring these redundant computations will nevertheless still ensure the solution can be obtained. Without loss of generality, a similar result holds for A_i.

The roles of Theorems 3.4.1 and 3.4.2 arise in determining the values of s.

Our first improved result utilizing Theorem 3.4.1 and Lemma 3.5.1 is given as follows. Let

$$t' \in \left\{1, 2, \ldots, \left\lceil \frac{t}{2} \right\rceil \ : \frac{nt'}{t} \in \mathbb{Z}\right\}$$

be the value of t_1 that minimizes

$$|Y|\binom{n_1-1}{t_1-1} + \binom{n-n_1}{t-t_1}.$$

If t' and the corresponding $t-t'$ satisfy the conditions stated in Theorem 3.4.1 then s can be taken to be $2n_2-n_1$ and the computational complexity of solving the problem is given by

$$O\left((2n_2-n_1+1)|Y|\binom{n'-1}{t'-1} + n\binom{n-n'}{t-t'} + (2n_1-n_2-1)|Y|\binom{n'-2}{t'-1}\right)$$

where $n' = \frac{nt'}{t}$. The corresponding memory requirement is given by

$$O\left(\min\left\{|Y|\binom{n'-1}{t'-1}, \binom{n-n'}{t-t'}\right\}\right).$$

This provides a strict improvement over the current best known result of [12] with regards to time complexity while maintaining an equal amount of storage required.

We can further improve on this result if $t_2 \approx 2t_1$ using Theorem 3.4.2. Let $t' \in \{1, 2, \ldots, \lceil \frac{t}{2} \rceil \ : \frac{nt'}{t} \in \mathbb{Z}\}$ be the value of t_1 that minimizes

$$|Y|\binom{n_1}{t_1} + \binom{n-n_1-1}{t-t_1-1}.$$

If t' and the corresponding $t - t'$ satisfy the conditions stated in Theorem 3.4.2, then s can be taken to be $2kn_1 - (k-1)n_2$, where the value of k depends on the relation between t_1 and t_2 as highlighted in Theorem 3.4.2. For brevity, let $k' = 2kn_1 - (k-1)n_2$. The computational complexity of solving the problem is then given by

$$O\left((k'+1)\binom{n-n'-1}{t-t'-1} + n|Y|\binom{n'}{t'} + (n-k'-1)\binom{n-n'-2}{t-t'-1}\right)$$

where $n' = \frac{nt'}{t}$. The corresponding memory requirement is given by

$$O\left(\min\left\{|Y|\binom{n'}{t'}, \binom{n-n'-1}{t-t'-1}\right\}\right).$$

3.6 Application to the GPS Scheme

In this section, we provide 2 improved approaches of attacking the GPS identification scheme utilizing CLP parameters. The CLP parameters for the GPS scheme based on the DLP with LHWP exponents were proposed in [18]. We show that these parameters satisfy the conditions to utilize the PDSS and provide concrete security evaluations. Furthermore, we demonstrate additional enhancements which can be applied to provide further improvements for these specific parameters. One caveat is that the $\mathrm{ord}(g)$ is kept secret in the GPS. However, it is already known (for instance in [18]) that this can be circumvented without any significant increase in computational resources.

The CLP parameters are that the private key be a product of a 30-bit number with 12 nonzero bits and a 130-bit number with 26 nonzero bits. As such, $|Y| = \binom{30}{12}$, $n = 130$ and $t = 26$. Moreover, some simple computational checking reveals that $t' = 10$ is the value of t_1 which minimizes $|Y|\binom{n_1-1}{t_1-1} + \binom{n-n_1}{t-t_1}$. Hence $t_1 = 10$, $t_2 = 16$, $n_1 = 50$ and $n_2 = 80$. It is easily verified that $k = 2$ in Theorem 4.4.1 and the conditions for t_i are also satisfied. As a result, $2n_2 - n_1 + 1 = 111$ and so only 111 blocks are required as opposed to 130 required in [33, 34].

The additional enhancement to further reduce from 111 blocks to 106 blocks proceeds as follows. We refer back to the proof of Theorem 4.4.1, at the two separate parts where $|(C_{0,3n_1-n_2} \cup C_{2n_2-n_1,n}) \cap Y| \leq 5t_1 - 2t_2 - 7$ and $|(C_{0,n_1} \cup C_{2n_2-n_1,n}) \cap Y| \geq 3t_1 - t_2 + 4$. These bounds correspond to

$|(C_{0,70} \cup C_{110,130}) \cap Y| \leq 11$ and $|(C_{0,50} \cup C_{110,130}) \cap Y| \geq 18$ upon applying the CLP parameters. Hence if $|B_i \cap Y| \geq 17$, $\forall i \in [0, 110)$, then

$$|(C_{0,70} \cup C_{105,130}) \cap Y| \leq 16$$

and if $|B_i \cap Y| \geq 9$, $\forall i \in [0, 110)$

$$|(C_{0,50} \cup C_{105,130}) \cap Y| \geq 18.$$

By property 1, this proves that there exists some $i' \in [0, 106)$ such that

$$|B_{i'} \cap Y| = 16.$$

This additional enhancement shows that only 106 blocks are required. The computational complexity of this attack can now easily be computed to be $2^{64.49}$ exponentiations with memory requirement of $2^{54.58}$ which is an improvement in time complexity over the results of [34] without any added storage requirements.

Our second improvement is achieved by minimizing $|Y|\binom{n_1}{t_1} + \binom{n-n_1-1}{t-t_1-1}$. The minimum is attained when $t_1 = 9$, $t_2 = 17$, $n_1 = 45$ and $n_2 = 85$. In this case, we have that $t_2 = 2t_1 - 1$ and thus Theorem 3.4.2 can be applied. From the result of Theorem 3.4.2, we obtain $N = 96$. We can actually further reduce to $N = 95$ by following the proof in Theorem 3.4.2 until the portion where we have $|B_{4n_1-n_2} \cap Y| \leq t_2 - 1$. This corresponds to $|B_{95} \cap Y| \leq 16$. Instead in this particular case, we can be a little more precise and consider $|B_{94} \cap Y| \leq 16$. Hence, it follows that $|C_{49,94} \cap Y| \geq 10$. This implies $|C_{50,94} \cap Y| \geq 9$. Combining with earlier results in the proof of Theorem 3.4.2, we obtain $|C_{40,94} \cap Y| \geq 17$. Hence, only 95 blocks are required. The computational complexity of this attack can now easily be computed to be $2^{63.95}$ exponentiations with memory requirement of $2^{55.83}$. It is of note that this second approach yields the first known result that achieves a time complexity of under 2^{64} at a slight additional expense of memory space.

3.7 Admissible values of t_i

In this section, we investigate the type of parameters which are susceptible to the results of our work. In general, such problems can solved using the theory of Ehrhart polynomials. However in both of our cases, since we only deal with some particular forms involving systems of bivariate inequalities, the analysis can be simplified via Pick's theorem.

Let P be a simple polygon on the x-y plane such that all of x and y coordinates of it vertices are integers. Let A denote the area of the polygon P, i denote the number of (standard) lattice points strictly in the interior of P and b denote the number of (standard) lattice points on the boundary of P. Pick's theorem asserts the following.

Pick's theorem.

$$A = i + \frac{b}{2} - 1$$

An elementary proof of Pick's theorem can be found in [57].

Definition 3.7.1. *For given $k \in \mathbb{Z}^+ \backslash \{1\}$, t_1 and t_2 are admissible if they satisfy*

$$\frac{2k+1}{k+1}t_1 - \frac{3k+1}{k+1} \leq t_2 \leq \frac{2k-1}{k}t_1 + \frac{3k-2}{k},$$
$$\frac{2k-1}{k}t_1 < t_2 \leq \frac{2k+1}{k+1}t_1.$$

In the case of the GPS scheme with CLP parameters of $k = 2, t_1 = 10, t_2 = 16$, the collection of admissible t_i values for the same k are given in Figure 3.1. Figure 3.2 highlights the point where these CLP parameters are located.

The set of admissible values of t_i might seem restrictive from Figure 3.1. However, we show that the set of admissible values increase at a quadratic rate with respect to k. In particular, we prove that the number of admissible values of t_i for any integer $k \geq 2$ is given by $9k^2 - 4$.

Theorem 3.7.1 *Let $k \in \mathbb{Z}^+ \backslash \{1\}$. The number of integer pairs (t_1, t_2) satisfying*

$$\frac{2k+1}{k+1}t_1 - \frac{3k+1}{k+1} \leq t_2 \leq \frac{2k-1}{k}t_1 + \frac{3k-2}{k},$$
$$\frac{2k-1}{k}t_1 < t_2 \leq \frac{2k+1}{k+1}t_1$$

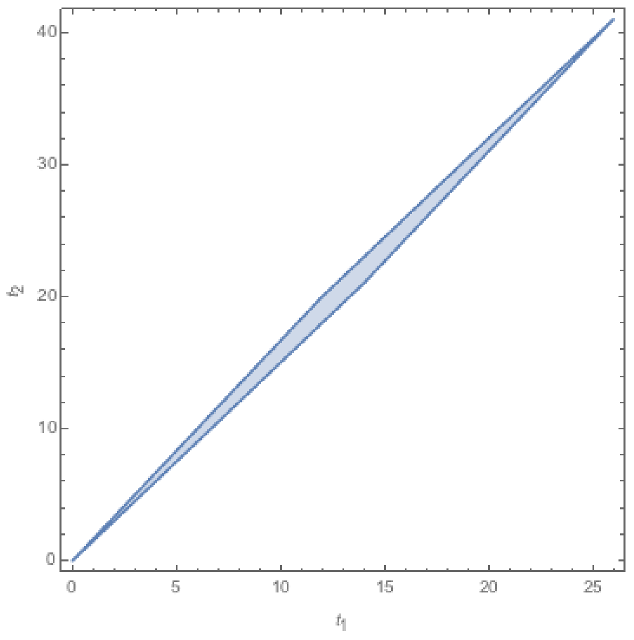

Figure 3.1: Region of admissible t_i values ($t_1 = 10$, $t_2 = 16$, $k = 2$)

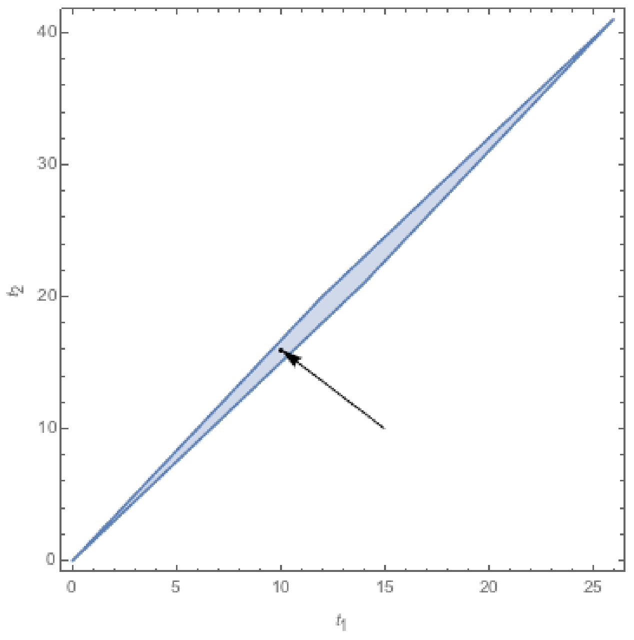

Figure 3.2: Location of CLP parameters ($t_1 = 10$, $t_2 = 16$, $k = 2$)

is $9k^2 - 4$.

Proof. The admissible values of t_i can be viewed to lie on the x-y plane. By solving the linear equations at equality, it straightforward to obtain the 4 vertices to be

$$(0,0),\ \ (3k^2+k, 6k^2-k-1),$$
$$(6k^2+2k-2, 12k^2-2k-3), (3k^2+k-2, 6k^2-k-2).$$

The area of this parallelogram enclosed by these vertices can be computed by

$$\frac{1}{2}\begin{vmatrix} 0 & 3k^2+k & 6k^2+2k-2 & 3k^2+k-2 & 0 \\ 0 & 6k^2-k-1 & 12k^2-2k-3 & 6k^2-k-2 & 0 \end{vmatrix}$$
$$= 9k^2 - 3k - 2$$

The total number of pairs of integer points satisfying the boundary segment $t_2 = \frac{2k-1}{k}t_1$ is easily checked to be $3k+2$. Since $\gcd(k+1,\, 2k+1) = 1$; for the boundary segment $t_2 = \frac{2k+1}{k+1}t_1$, it is $3k-1$. For the boundary segment $t_2 = \frac{2k-1}{k}t_1 + \frac{3k-2}{k}$, it is $3k+2$. For the boundary segment $t_2 = \frac{2k+1}{k+1}t_1 - \frac{3k+1}{k+1}$, it is $3k-1$. Hence, the total number of integer points lying on the boundary (excluding double counting at the vertices) is

$$2(3k+2) + 2(3k-1) - 4 = 12k - 2.$$

By Pick's theorem, the number of integer points lying strictly within the enclosed region is thus

$$9k^2 - 3k - 2 - \frac{1}{2}(12k-2) + 1 = 9k(k-1).$$

Finally, by adding up the remaining integer points on 3 of the boundaries, we obtain the desired number to be $9k^2 - 4$. □

While the number of admissible points for t_i is of order $O(k^2)$ in Theorem 3.4.1, the number of such admissible points in Theorem 3.4.2 is of the order $O(k)$.

Definition 3.7.2 *For given* $k \in \mathbb{Z}^+$, t_1 *and* t_2 *are admissible if they satisfy*

$$\frac{2k+1}{k+1}t_1 \leq t_2 \leq \frac{2k+1}{k+1}t_1 + \frac{3k+2}{k+1},$$

$$2t_1 - 2 < t_2 \leq 2t_1.$$

Theorem 3.7.2 *Let $k \in \mathbb{Z}^+$. The number of integer pairs (t_1, t_2) satisfying*

$$\frac{2k+1}{k+1}t_1 \leq t_2 \leq \frac{2k+1}{k+1}t_1 + \frac{3k+2}{k+1},$$
$$2t_1 - 2 < t_2 \leq 2t_1$$

is $9k+6$.

Proof. The admissible values of t_i can be viewed to lie on the x-y plane. By solving the linear equations at equality, it straightforward to obtain the 4 vertices to be

$$(0,0),\ \ (2k+2, 4k+2),$$
$$(5k+3, 10k+4), (3k+1, 6k+2).$$

The area of this parallelogram enclosed by these vertices can be computed by

$$\frac{1}{2}\begin{vmatrix} 0 & 2k+2 & 5k+3 & 3k+1 & 0 \\ 0 & 4k+2 & 10k+4 & 6k+2 & 0 \end{vmatrix}$$
$$= 6k+2$$

The total number of pairs of integer points satisfying the boundary segment $t_2 = 2t_1$ is easily checked to be $3k+2$. For the boundary segment $t_2 = \frac{2k+1}{k+1}t_1$, it is 3 since $\gcd(k+1,\, 2k+1) = 1$. For the boundary segment $t_2 = 2t_1 - 2$, it is $3k+2$. For the boundary segment $t_2 = \frac{2k+1}{k+1}t_1 + \frac{3k+2}{k+1}$, it is 3. Hence, the total number of integer points lying on the boundary (excluding double counting at the vertices) is

$$2(3) + 2(3k+1) - 4 = 6k+6.$$

By Pick's theorem, the number of integer points lying strictly within the enclosed region is thus

$$6k + 2 - \frac{1}{2}(6k+6) + 1 = 3k.$$

Finally, by adding up the remaining integer points on all of the boundaries, we obtain the desired number to be $9k+6$. $\square$

Figure 3.3 and Figure 3.4 illustrate the region of admissible t_i values and the location of CLP parameters for $t_1 = 9$, $t_2 = 17$, $k = 2$ respectively.

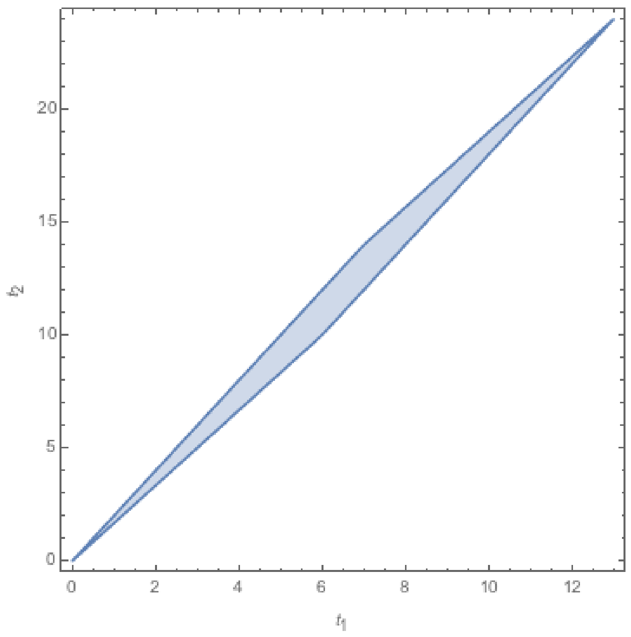

Figure 3.3: Region of admissible t_i values ($t_1 = 9$, $t_2 = 17$, $k = 2$)

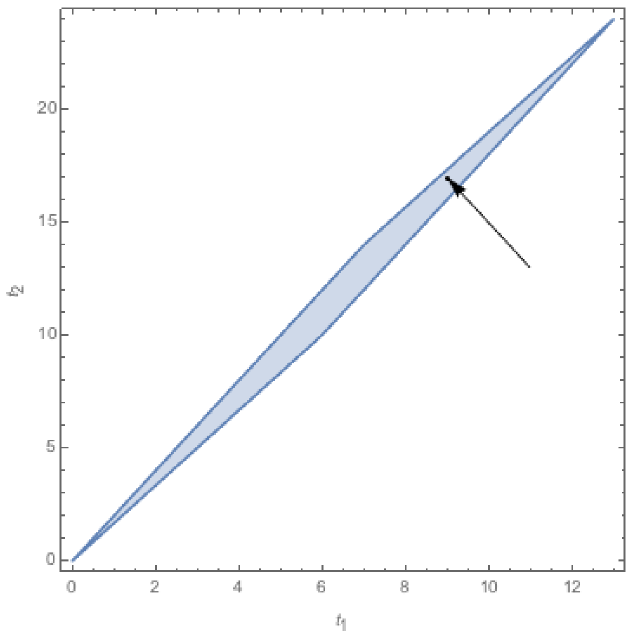

Figure 3.4: Location of CLP parameters ($t_1 = 9$, $t_2 = 17$, $k = 2$)

Table 3.1: Results

Attacks	Method	Exponentiations	Storage
[18]	CSS	$2^{77.3}$	$2^{43.9}$
[14]	PSS	$2^{65.48}$	$2^{56.09}$
[12]	PSS	$2^{64.53}$	$2^{54.58}$
[This work]	PDSS	$2^{64.49}$	$2^{54.58}$
[This work]	PDSS	$2^{63.95}$	$2^{55.83}$

3.8 Results

Table 3.1 highlights the results of our work when compared with other existing state of the art.

As evident from the results of Table 3.1, the method of PDSS introduced in this work provides improvements to current known attacks and analysis of the GPS identification scheme. They are two ways of approach to apply the PDSS method. The former provides a reduced complexity of the best current known GPS scheme without additional increment in storage requirement. The latter provides an even lower time complexity that falls under 2^{64} for the first time at the expense of some storage increment.

3.9 Conclusion

In this work, we introduce a method of parameters dependent splitting system (PDSS) which can be applied to analyze the security of the GPS scheme which invokes the DLP with low hamming weight exponents as its security basis. The method is shown to provide better results over existing state of the art. In particular, we show for the first time that the security barrier for the GPS scheme is under 2^{64}. This work also serves to provide a framework to identify suitable and more secure parameters for future constructions of cryptographic schemes involving DLP with LHWP exponents. The analysis of the minimum number of blocks required in the parameterized splitting of given inputs might also be of independent interest in the field of Combinatorics.

Chapter 4

Password Recovery via Rainbow Tables

4.1 Introduction

One-way functions have important relevance in the context of passwords for user authentication purposes. For instance, storing user passwords in plaintext will result in a breach of security should the password file be compromised. In particular, cryptography hash functions are employed to store the hash digests of passwords instead. The authentication process then compares the stored hash with the hash of a user's input.

There are various ways to invert such hash functions to recover the plaintext password. Some of these methods include exhaustive search, precomputations and time-memory trade-offs. In exhaustive search, every feasible password combination is searched until the correct one is obtained. In precomputations, a large table of all feasible plaintext passwords and their corresponding hashes are stored. Due to the large password space, both of these methods suffer the drawbacks of requiring long online computation time and large storage space for exhaustive search and precomputations respectively.

To alleviate such issues, cryptanalytic time-memory trade-off was introduced by Hellman [27]. This is a method to invert generic one-way functions by utilizing a trade-off between time and memory cost. Such technique can be applied in password recovery by inverting the relevant hash function. Time-memory trade-off techniques have also been applied in practical attacks such as A5/1 [11] and LILI-128 [44].

There are also other methods which can be performed for passwords or encryption key recovery by identifying weaknesses of certain hashing algorithms based on the collisions of hashes [16, 22], [47] and [46]. However, these methods are only applicable to specific hashing algorithms.

Rainbow tables were introduced by Oechslin [40] which provided certain improvements over the original method by Hellman. The main difference lies in the usage of different reduction functions when generating a table during the offline phase. They have been shown to be efficient in recovering LM hash passwords [40] as well as UNIX passwords [38].

Subsequently, further analysis and improvements to these time-memory trade-off methods were proposed. For instance, endpoints can be stored more efficiently by applying prefix-suffix decomposition [5, 6] or by applying compressed delta encoding [4]. Checkpoints were also proposed in [5] to reduce false alarms. A theoretical framework of analyzing false alarms was provided in [29] by comparing some trade-off algorithms in [30]. A formula for choosing parameters to achieve a higher success rate in Hellman's time-memory trade-off method was discussed in [45]. Further improvements to the trade-off efficiency of rainbow tables were proposed in [52, 53, 55].

4.1.1 Our Contributions

This chapter is motivated by the fact that distributions of passwords tend to heavily skewed. Indeed, recent surveys in [31, 56] indicate that top frequently used user passwords constitute a significant proportion of the database. Such common passwords also tend to be identical across various databases. In this chapter, we wish to incorporate such frequently used passwords in rainbow tables generated to ensure that they can be recovered during the online phase. A canonical method of incorporating such passwords involves assigning passwords at the start points of rainbow chains. We show that it is possible to incorporate multiple passwords across a single chain instead. This is an extension of the results in [54] where only 3 or 4 passwords assigned in a chain were analyzed. It is an open problem whether more than 4 given passwords can be incorporated. We show that this is possible using a different technique in this chapter which affirm that this can be generalized to incorporate more passwords regardless of the type of hash applied. Furthermore, we prove that this method of incorporating frequently used passwords provide a faster recovery time during the online phase as opposed to the natural way of assigning them at the start of each chain. We also include results for typical rainbow change lengths in practical settings.

4.1.2 Chapter Organization

The organization of this chapter is as follows. Chapter 4.2 describes some of the methods to invert hash functions, including time-memory trade-off techniques which our work revolves upon. In Chapter 4.3, we show our method of incorporating multiple given passwords along a rainbow chain is feasible. An example is provided in Chapter 4.4. Chapter 4.5 considers the scenarios of multiple passwords recovery. Explicit constructions on how to achieve long dictionary rainbow chains are provided in Chapter 4.6. Theoretical evaluations are analyzed in Chapter 4.7 and results are given in Chapter 4.8. We conclude in Chapter 4.9.

4.2 Inverting Cryptographic Hash Functions

Let h be a cryptographic hash function, p_i a password and c_i its corresponding hash digest such that $c_i = h(p_i)$. We outline a number of ways to invert hash functions fundamental in password recovery. They are exhaustive search, precomputations and time-memory trade-off.

Exhaustive Search
Given a hash digest c, the hash of each possible p_i in the password space is computed until a match $c = h(p_i)$ is obtained. The drawback is that since the password space is large, this method has a large online time complexity.

Precompuatations
A table of (p_i, c_i) are precomputed and stored for all possible p_i in the password space. Given a hash digest c, its preimage can be obtained when there is a match $c = c_i$. The drawback is that large memory cost is required to store such pairs since the password space is large.

Time-Memory Trade-off
We present an overview of Oecshlin's method using rainbow tables. More details can be found in [40]. It is best described in two phases; the offline phase and the online phase. Let R_i be a reduction function which maps hash digests to plaintext passwords uniformly. For example a typical choice of

reduction function can be given by $R_i(c_i) = p_j$ where $p_j \equiv c_i + r_i \bmod n$, where n is the size of the password space and $r_i \in \mathbb{Z}$. A reduction function simply maps the hash of p_i to some p_j. Here, the hash digests c_i are converted to decimal representations and each password p_j is represented by an integer value between 0 and $n-1$ inclusive. During the offline phase, a list of passwords can be generated as the start points. Each of these passwords are then alternatively hashed and reduced (via the reduction functions) for a designated t number of times. The start and end points are then stored as pairs in a table. During the online phase, the desired hash H to be inverted is then reduced using R_t. If the outcome $R_t(h)$ is not a match with any end point entries in the table, one proceeds to compute $R_t(h(R_{t-1}(H)))$ and search for a match of with the end point entries of the table. This process continues to $R_t \circ h \circ R_{t-1} \circ \cdots \circ h \circ R_1(H)$ until a match is found. If no matches are found throughout the process, then the given hash cannot be inverted. If a match is found, one can then identify to which starting point it corresponds to and reapply the series of hashes and reductions operations to recover the password corresponding to its hash H. One drawback of this method is that the success rate is not 100%.

Hellman Tables vs Rainbow Tables

One big distinction between Hellman tables [27] and Rainbow tables [40] is that the former applies the same reduction function throughout the chains while the latter utilizes distinct reduction functions. Chains which comprise of distinct reduction functions are referred to as rainbow chains. In doing so, Rainbow tables provide an advantage in that their generated chains have lower chances of merging, thereby reducing redundancies. More details of its improvements can be found in [40]. In view of this, our work considers such chains having distinct reduction functions.

4.3 Incorporation of Fixed Passwords in Rainbow Tables

There are certain password compositions that are known to be much more popular among users, usually due to easy remembrance. Indeed, recent surveys provided in [31, 56] corroborate this. However, as discussed in Chapter 4.2.3, rainbow tables do not provide a guaranteed means of recovering the password of its corresponding hash digest. To ensure that a frequently used password can be recovered, it has to be present at some point during the construction of the rainbow chains. One natural way of ensuring this is to assign such passwords at the beginning of each chain, i.e. at their start points. This enables successful recoveries of given starting point passwords regardless of any subsequent constructions for the remainder of the chains. In this chapter, we show how multiple frequently used passwords can be incorporated along a rainbow chain. This facilitates a more efficient means of recovering such passwords during the online phase. In addition, the evaluation of its superiority will also be discussed. Some results have been given in [54] showing that 3 or 4 passwords can be incorporated along a chain. We prove a generalization which shows that up to t such passwords can be incorporated along a chain of length t where the length refers to the total number of hash digests a chain computes until termination.

Suppose we have a list of t distinct frequently used passwords p_1, p_2, $\ldots$, p_t to be incorporated in a rainbow chain of length t. Denote an ordered t-tuple $(p_{i,1}, p_{i,2}, \ldots, p_{i,t})$ to be an arrangement in which the t passwords can be incorporated in a rainbow chain such that the reduction functions will remain distinct. Here, $p_{i,1}$, $p_{i,2}$, $\ldots$, $p_{i,t}$ is a permutation of p_1, p_2, $\ldots$, p_t. For example, if $t = 3$, (p_2, p_1, p_3) implies that p_2 can be incorporated at the start of a chain, followed by p_1 and then p_3 in a way that the reduction functions in between each pair of consecutively placed passwords will differ. In other words, there exists distinct reduction functions R_1, R_2 such that $R_1(h(p_2)) = p_1$ and $R_2(h(p_1)) = p_3$. We recall that $R_i(c_i) = p_j$ where $p_j \equiv c_i + r_i \bmod n$, where n is the size of the password space and $r_i \in \mathbb{Z}$. In this context, two reduction functions are distinct if and only if the corresponding r_i are distinct.

Our objective is to show that $(p_{i,1}, p_{i,2}, \ldots, p_{i,t})$ exists. Let the size of the password space be n. For brevity, $\equiv$ refers to $\equiv \bmod n$.

We first demonstrate the cases $t = 3$ and $t = 4$ which were proven in [54]. We include these in to highlight the differences between the general case and $t = 3, 4$. Each of the three proofs provides a different flavour of approach. The case $t = 4$ is handled by applying case by case considerations. This method does not seem to extend to proof the general case and hence we undertake a graph theoretic approach.

Theorem 4.3.1. *Any given 3 passwords can be recovered regardless of the size of keyspace and the hash applied.*

Proof. To prove the theorem, we show that there exists at least one arrangement to insert these 3 passwords such their corresponding reduction functions will be distinct. Let the 3 passwords be p_1, p_2 and p_3. Let $h(p_1) = a_1$, $h(p_2) = a_2$ and $h(p_3) = a_3$. Suppose for all 6 arrangements, each arrangement results in having identical reduction functions. Thus, we obtain the following system of equations:

$$p_2 - a_1 = p_3 - a_2 \tag{4.1}$$

$$p_3 - a_1 = p_2 - a_3 \tag{4.2}$$

$$p_1 - a_2 = p_3 - a_1 \tag{4.3}$$

$$p_3 - a_2 = p_1 - a_3 \tag{4.4}$$

$$p_1 - a_3 = p_2 - a_1 \tag{4.5}$$

$$p_2 - a_3 = p_1 - a_2 \tag{4.6}$$

Comparing equations (4.1) and (4.3), we get $p_1 + p_2 = 2p_3$.
Comparing equations (4.2) and (4.5), we get $p_1 + p_3 = 2p_2$.
Comparing equations (4.4) and (4.6), we get $p_3 + p_2 = 2p_1$.

Solving these 3 new equations, we obtain $p_1 = p_2 = p_3$. This is a contradiction since p_1, p_2, p_3 are distinct modulo n; thus proving Theorem 4.3.1. □

Theorem 4.3.2. *Any given 4 passwords can be recovered regardless of the size of keyspace and the hash applied.*

Proof. Denote the 4 passwords to be p_1, p_2, p_3 and p_4 and let $h(p_1) = a_1$, $h(p_2) = a_2$, $h(p_3) = a_3$ and $h(p_4) = a_4$.

Consider the 3 passwords p_1, p_2, p_3 to be placed in the first 3 entries of the chain. Applying Theorem 4.3.1, there exists an arrangement which will result in distinct reduction functions. Without loss of generality, assume that the first 3 passwords in the chain are p_1, p_2, p_3 in that order such that the corresponding reduction functions are distinct. Then, p_4 will be in the 4^{th} entry of the chain. Suppose $p_4 - a_3 \neq p_2 - a_1$ and $p_4 - a_3 \neq p_3 - a_2$. Then $p_1p_2p_3p_4$ is the desired order to place the passwords which ensures distinct reduction functions.

Suppose either $p_4 - a_3 = p_2 - a_1$ or $p_4 - a_3 = p_3 - a_2$.

Case 1: $p_4 - a_3 = p_2 - a_1$
Consider the arrangement $p_4p_1p_2p_3$. If $p_1 - a_4 \neq p_2 - a_1$ and $p_1 - a_4 \neq p_3 - a_2$, we are done. Otherwise, either $p_1 - a_4 = p_2 - a_1$ or $p_1 - a_4 = p_3 - a_2$.

Case 1(a): $p_1 - a_4 = p_2 - a_1 = p_4 - a_3$.
Consider the arrangement $p_1p_3p_4p_2$. Suppose not all the reduction functions are distinct, then $p_3 - a_1 = p_2 - a_4$. Next, consider the arrangement $p_4p_1p_3p_2$. If not all the reduction functions are distinct, then $p_3 - a_1 = p_2 - a_3$. This implies $a_4 = a_3$ and thus $p_1 = p_4$ which is a contradiction.

Case 1(b): $p_1 - a_4 = p_3 - a_2$ and $p_4 - a_3 = p_2 - a_1$
If $p_1 - a_4 = p_1 - a_3$, then $a_3 = a_4$. Thus $p_4p_3p_1p_2$ will be the desired arrangement. If $a_3 \neq a_4$, consider the arrangements $p_4p_2p_3p_1$, $p_1p_3p_4p_2$, $p_4p_1p_3p_2$, $p_2p_4p_1p_3$ and $p_1p_4p_2p_3$ in the order as stated. Suppose none of these arrangements result in distinct reduction functions.
By considering $p_4p_2p_3p_1$, we get $p_2 - a_4 = p_1 - a_3$.
By considering $p_1p_3p_4p_2$, we get $p_1 - a_3 = p_2 - a_4 = p_3 - a_1$.
By considering $p_4p_1p_3p_2$, we get $p_1 - a_4 = p_2 - a_3$.
By considering $p_2p_4p_1p_3$, we get $p_4 - a_2 = p_3 - a_1$.
By considering $p_1p_4p_2p_3$, we get $p_4 - a_1 = p_3 - a_2$.
From above arrangement $p_2p_4p_1p_3$, we obtain $p_3 - a_1 = p_4 - a_2$ and from arrangement $p_1p_4p_2p_3$, we obtain $p_4 - a_1 = p_3 - a_2$. Hence, $p_3 = p_4$, a contradiction.

Case 2: $p_4 - a_3 = p_3 - a_2$ and $p_4 - a_3 \neq p_2 - a_1$
Consider the arrangement $p_3p_4p_1p_2$. If $p_4 - a_3 \neq p_1 - a_4$ and $p_2 - a_1 \neq p_1 - a_4$, we are done. Otherwise, either $p_4 - a_3 = p_1 - a_4$ or $p_1 - a_4 = p_2 - a_1$.

Case 2(a): $p_4 - a_3 = p_3 - a_2 = p_1 - a_4$
Consider the arrangement $p_3p_4p_2p_1$. If this arrangement results in distinct reduction functions, we are done; otherwise $p_2 - a_4 = p_1 - a_2$. Next, consider arrangement $p_4p_2p_3p_1$. If this arrangement does not result in distinct reduction functions again, then we must have $p_2 - a_4 = p_1 - a_3$. Hence, $a_2 = a_3$ which implies $p_3 = p_4$, a contradiction.

Case 2(b): $p_1 - a_4 = p_2 - a_1$ and $p_4 - a_3 = p_3 - a_2$
If $a_1 = a_3$, consider the arrangement $p_3p_2p_1p_4$. If this is not the desired arrangement, then $a_2 = a_4$. Hence, $p_2p_1p_3p_4$ will be the desired arrangement. If $a_1 \neq a_3$, consider the arrangements $p_4p_1p_3p_2$, $p_2p_1p_3p_4$, $p_3p_2p_4p_1$ and $p_2p_3p_1p_4$

in this order. Suppose none of these arrangements result in distinct reduction functions.
By considering $p_4p_1p_3p_2$, we get $p_3 - a_1 = p_2 - a_3$.
By considering $p_2p_1p_3p_4$, we get $p_3 - a_1 = p_2 - a_3 = p_1 - a_2$.
By considering $p_3p_2p_4p_1$, we get $p_4 - a_2 = p_1 - a_4 = p_2 - a_1$.
By considering $p_2p_3p_1p_4$, we get $p_1 - a_3 = p_4 - a_1$.
Thus, we obtain the following set of equations: $p_3 - a_1 = p_2 - a_3 = p_1 - a_2$, $p_4 - a_2 = p_1 - a_4 = p_2 - a_1$ and $p_1 - a_3 = p_4 - a_1$. From $p_3 - a_1 = p_1 - a_2$ and $p_2 - a_1 = p_4 - a_2$, we obtain $p_1 - p_4 = p_3 - p_2$. From $p_1 - a_3 = p_4 - a_1$ and $p_3 - a_1 = p_2 - a_3$, we obtain $p_2 - p_1 = p_3 - p_4$. Thus after simplifying, $p_2 = p_3$, a contradiction.

This proves Theorem 4.3.2. □

Theorem 4.3.3 *Any t given distinct passwords can be incorporated in a rainbow chain such that corresponding reduction functions are distinct i.e. $(p_{i,1}, p_{i,2}, \ldots, p_{i,t})$ exists.*

Proof. Consider all expressions of $p_j - c_i \bmod n$, $1 \le i, j \le t$, $i \ne j$. There are a total of $t(t-1)$ distinct expressions of $p_j - c_i$. Construct a graph G with $t(t-1)$ vertices where $v_{j,i}$ corresponding to $p_j - c_i$ are the vertices of the graph.

There a total of $t!$ possible ways for the passwords to be arranged. Suppose for a contradiction $p_i \ne p_j$ for $i \ne j$ and that none of these $t!$ arrangements will result in a feasible chain where the reduction functions are distinct. Thus, there are a total of $t!$ equations of the form $p_j - c_i = p_{j'} - c_{i'}$. However, some number of these equations might appear multiple times. We can remove these redundancies by noting that each $p_j - c_i = p_{j'} - c_{i'}$ can repeat at most $(t-2)!$ times. Hence, there are at least $\frac{t!}{(t-2)!} = t(t-1)$ distinct equations.

Next, we identity all of these $t(t-1)$ distinct equations. We connect an edge between vertices $v_{j,i}$ and $v_{j',i'}$ if $p_j - c_i = p_{j'} - c_{i'}$. If there is a subgraph of G with more than t connected components, then $p_i = p_j$ for some i and j, a contradiction. Hence, any subgraph of G has at most t connected components. The resulting graph of G has $t(t-1)$ edges.

Suppose there are a total of k connected components. Let each of these connected components $G_1, ..., G_k$ be subgraphs of G such that each G_i contain n_i vertices. In the most extreme case, each of these G_i graphs are complete graphs so that they have $\binom{n_i}{2}$ edges. Since G has a total of $t(t-1)$ edges, we have that in the most extreme case that

$$\sum_{i=1}^{k} \binom{n_i}{2} = t(t-1). \tag{4.7}$$

Each subgraph corresponds to n_i equations. Also, there are at most $2t$ unknown variables $p_1, ..., p_t, c_1, ..., c_t$. Hence, we get a contradiction if $\sum_{i=1}^{k} n_i > 2t$. Thus, $\sum_{i=1}^{k} n_i = 2t$.

Therefore, we need to handle the specific case where $k = 2$ and $n_1 = n_2 = t$. In this case, we get a series of equations

$$p_{j_1} - c_{i_1} = p_{j_2} - c_{i_2} = \cdots = p_{j_t} - c_{i_t} \tag{4.8}$$

and

$$p_{j'_1} - c_{i'_1} = p_{j'_2} - c_{i'_2} = \cdots = p_{j'_t} - c_{i'_t} \tag{4.9}$$

where each of $\{j_1, \ldots, j_t\}$, $\{i_1, \ldots, i_t\}$, $\{j'_1, \ldots, j'_t\}$ and $\{i'_1, \ldots, i'_t\}$ is a permutation of $\{1, 2, \ldots, t\}$. By subtracting equations (4.8) from equations (4.9) and eliminating all c_i in the process, we obtain that

$$p_1 - p_{\sigma(1)} = p_2 - p_{\sigma(2)} = ... = p_t - p_{\sigma(t)} \tag{4.10}$$

where σ is a permutation of $\{1, 2, \ldots, t\}$. Let $p_1 - p_{\sigma(1)} = x$. Then by summing all of the equal expressions given in equations (4.10), we get $tx = 0$ and so $x = 0$ and hence $p_1 = p_2 = \ldots = p_t$, a contradiction. □

4.4 An Example

Suppose we want to recover a 7 letter password where the characters in the keyspace consist of all the lower case letters in the alphabet and hashed by MD5. In addition, we want to ensure that common passwords are recoverable. Given that "letmein", "abcdefg" and "testing" are three common 7 letters passwords, the goal is to include these passwords in rainbow chains so that they can be recovered. Denote H_i to be the hash digest of the i^{th} plaintext of a rainbow chain.

Set the beginning of the rainbow chain to be the password "testing". Upon hashing, the hashed value will be

$$H_1 = ae2b1fca515949e5d54fb22b8ed95575.$$

Convert this to the decimal representation and apply a reduction function R_1 to H_1 where

$$R_1(H_1) = H_1 + 4938209469 \mod 26^7.$$

Converting $R_1(H_1)$ back to the password representation will correspond to the password "letmein". The next step of the chain is to hash "letmein". This will result in

$$H_2 = 0d107d09f5bbe40cade3de5c71e9e9b7.$$

Then, apply a reduction function R_2 to H_2 where

$$R_2(H_2) = H_2 + 3129034064 \mod 26^7.$$

Converting $R_2(H_2)$ back to the password representation will correspond to the password "abcdefg". Hence, this rainbow chain consists of the 3 passwords which we want to be able to recover.

4.5 Scenario of Recovering Multiple Passwords

We shall aptly refer to the inclusion of given common passwords into a rainbow chain as dictionary rainbow chain since many of such common user chosen passwords are those that can found in a dictionary. Intuitively, there does not seem to be a need for constructing a dictionary rainbow chain since rainbow tables with very high probability of success will already seem very likely to include dictionary words. However, this is not the case. Suppose that an arbitrary rainbow table has a high success rate of 99.9%. The probability that all 5000 of the most common passwords is recoverable = 0.999^{5000} = 0.00672. Hence, without the use of a dictionary rainbow chain, it is incredibly unlikely that all 5000 of such passwords can be recoverable. On the other hand, by implementing a dictionary rainbow chain, these 5000 passwords can be generated in a single rainbow chain. Since a typical chain length exceeds that number, using a dictionary rainbow chain will indeed be feasible with 100% recovery rate.

For the purpose of further clarity, suppose a few passwords are allowed to be off the radar in the rainbow table, i.e. a few passwords cannot be recovered. We will like to find out what is the probability that 99.9% of the 5000 passwords are still recoverable from a rainbow table with a success rate of 99.9%. If this probability is extremely close to 1, then generating a dictionary rainbow chain is unnecessary since most of the passwords can be recoverable by using the usual rainbow table. The details of the computations are described below.

Let X be the distribution of the number of passwords that can be recovered from the rainbow table. Then $X \sim B(5000, 0.999)$. The objective is to evaluate the value of $P(4995 \leq X \leq 5000)$.

de Moivre–Laplace theorem. *As n grows large and for k in the neighbourhood of np, the following approximation holds. For positive values of p and q such that $p + q = 1$,*

$$\binom{n}{k} p^k q^{n-k} \approx \frac{1}{\sqrt{2\pi npq}} e^{-\frac{(k-np)^2}{2npq}}$$

and the ratio of the two expressions converge to 1 as $n \to \infty$.

A proof of de Moivre–Laplace theorem can be found in [52].

Let Z be the standard normal distribution. Since we are only interested with an approximate value, we can apply the de Moivre - Laplace theorem to approximate X to a normal distribution Y such that $Y \sim N(4995, 4.995)$. This facilitates the computations and so we can compute $P(4995 \leq X \leq 5000)$ with the required continuity correction in the following way.

$$\begin{aligned} &P(4995 \leq X \leq 5000) \\ &= P(X \leq 5000) - P(X \leq 4994) \\ &\approx P(Y \leq 5000.5) - P(Y \leq 4994.5) \\ &= P(Z \leq \frac{5.5}{\sqrt{4.995}}) - P(Z \leq -\frac{0.5}{\sqrt{4.995}}) \\ &= P(Z \leq 2.4609) - P(Z \leq -0.2237) \\ &= 0.582 \end{aligned}$$

Since P$(4995 \leq X \leq 5000) \approx 0.582$ which is nowhere high enough to 1, therefore generating a dictionary rainbow chain is a necessity if one wants to recover most of the common passwords. Furthermore by doing so, all the common passwords will be recoverable with absolute certainty.

4.6 Construction of Chain

In this section, we describe two methods on how such chains incorporating given passwords can be constructed. We present a deterministic method and a probabilistic approach. For practical usage, the probabilistic approach is more efficient.

4.6.1 Deterministic Method

Consider all expressions of $p_j - c_i$ mod n, $1 \leq i, j \leq t$, $i \neq j$. Construct a $t \times t$ matrix such that the $p_j - c_i$ mod n value is in the (i,j) entry of the matrix. Since $p_i - c_i$ is undefined, we fill the main diagonal of the matrix with zeros.

Lemma 4.6.1 *($p_{i,1}$, $p_{i,2}$, ..., $p_{i,t}$) exists if and only if $t-1$ cells of the matrix (none of which belongs to the main diagonal) can be selected which satisfy the following properties:*
1) All have distinct entries and no two cells belong to either the same row or column.
2) The union of the i^{th} row with the i^{th} column of the matrix contain at least one of the selected cells $\forall$ i.
3) No two cells are reflections along the main diagonal.

Proof. Suppose all the above 3 properties are satisfied. Then, by property 1, the $t-1$ selected cells of the matrix say (i_1,j_1), (i_2,j_2), ..., (i_{t-1},j_{t-1}) are such that all the i's are distinct and all the j's are distinct. By property 2, there exists j_k such that i_k does not belong to the set $\{i_1, i_2, \ldots, i_{t-1}\}$ for some k. Then by property 3, we are able to generate the chain with p_k as the start point. Conversely, given a valid chain, if property 3 is not satisfied, then both $p_i - c_j$ mod n and $p_j - c_i$ mod n are selected for some i and j. This implies that either (p_i, p_j, p_i) or (p_j, p_i, p_j) exists, a contradiction. Suppose property 2 is not satisfied, then p_i will not be incorporated in the chain, a contradiction. Suppose 2 cells are picked from the same row, say p_i - a_j mod n and p_i - a_k mod n. However, these expressions cannot be both present in a valid chain. The same reasoning applies to columns. This, together with the fact that a valid chain requires distinct reduction functions which results in distinct entries and that $n-1$ reduction functions are required will imply property 1 is satisfied. This proves Lemma 5.6.1. □

Now, we proceed by inductively. Then for $t = k$, the $k \times k$ matrix contains entries which satisfy all the 3 properties above. Consider $t = k + 1$, we know from inductive hypothesis that there exists a row and column with no selected cells, say row i and column j where $i \neq j$. Now, we consider the entries $(i, k + 1)$ and $(k + 1, j)$.

Lemma 4.6.2. *If either $(i, k + 1)$ or $(k + 1, j)$ is distinct from the previously selected entries, we are done.*

Proof. It suffices to show that this new point together with the previous selected points will satisfy the 3 properties stated above since by lemma 1, this will imply this selected set of entries will form a valid chain. Property 1 is trivially true by construction. Suppose row i and column j do not have any entries selected. If the cell chosen is $(i, k+1)$, then row $k+1$ and column j will not contain any selected cells. Since $k + 1 \neq j$, property 2 is satisfied. A similar argument applies if the chosen cell is $(k + 1, j)$. Again, suppose $(i, k + 1)$ is the chosen cell, then obviously its reflection along the main diagonal $(k+1, i)$ cannot be one of the selected cells. A similar argument applies if the chosen cell is $(k + 1, i)$. Hence, property 3 is satisfied. This proves lemma 5.6.2. □

Suppose not, then we identify the cell whose value is identical to one of the previous selected. We deselect that original cell and consider the lines formed by the now 2 'empty' rows and 2 'empty' columns. By 'empty', we refer to no selected cells. The 2 'empty' rows and columns can be viewed to form the sides of a rectangle.

Lemma 4.6.3. *We can always pick a new cell (i.e. not the one which was just deselected) from the 3 remaining vertices of the formed rectangle such that this cell together with the previously selected ones, satisfy properties 2 and 3.*

Proof. First, we verify that the reflection of a rectangle along its main diagonal is still a rectangle such that its reflected edges are at right angles to its original edges. We can translate and rotate the rectangle and its main diagonal such that the main diagonal is a line $y = x$ on the Cartesian plane. Suppose that the vertices of this new rectangle are (a, b), (a, d), (c, d) and

(c, b). Upon reflection along the line $y = x$, the new coordinates are (b, a), (d, a), (d, c) and (b, c). This is a rectangle with corresponding edges at right angles to its original edges. Next, to prove Lemma 5.6.3, we consider the following 3 cases.

Case 1: Main diagonal does not intersect the vertices of the rectangle. Suppose we are unable to pick such a cell, this implies that the reflected rectangle has 3 of its vertices having the selected entries which is a contradiction since no two selected entries can be in the same row or column. Hence, property 3 is satisfied. Also, since the main diagonal does not intersect the vertices of the rectangle, any of the vertices of the rectangle being chosen will not violate property 2.

Case 2: Main diagonal intersects rectangle at two of its diagonally opposite vertices.
In this instance, we pick the only vertex that can be chosen. In this case, property 2 is automatically satisfied. Property 3 is also satisfied since this reflection of this chosen point has just been removed.

Case 3: Main diagonal intersects rectangle at exactly one of its vertex. Pick the chosen cell to be the vertex that is diagonally opposite to the deselected cell. Property 2 is then automatically satisfied. Suppose property 3 is not satisfied. This implies its reflection is one of the selected points but the removed point is on the same row or column as this removed point. This results in having 2 selected points in either the same row or column, a contradiction. This proves Lemma 4.6.3. □

If this new entry is distinct from the previous selected group of cells, we are done since property 1 will then be satisfied. Otherwise, we repeat the process of replacing cells having identical entries until we obtain k cells having distinct entries. If the process continues in a periodic cycle, we can always begin with a different starting point. The process can then be repeated. Eventually, the selected cells must contain distinct entries after a finite number of steps since we have proven in the earlier theorem that such a construction exists.

4.6.2 Probabilistic Method

There is a straightforward probabilistic method which is to incorporate passwords in some random order until we have arrived at one where the all the resulting reduction functions are distinct. Since we have proven that such

Table 4.1: Success probability of a chain construction

t	n	$p(t,n)$
5700	7 characters alphanumeric lowercase	99.9586%
10000	7 characters alphanumeric lower/upper case	99.997%
15000	8 characters alphanumeric lower/upper case	99.9999%

a chain exists, this probabilistic method is of Las Vegas type. In practice, this is the preferred method to employ the chain construction. We detail the reason as follows.

Let n denote the password space. Since values of c_i are digests of hash functions, they are assumed to be randomly distributed. Hence, $p_j - c_i$ are assumed to be randomly distributed. Therefore the probability that any arrangement of t ($t \geq 3$) incorporated password result in a valid chain where the reduction functions are all distinct is given by

$$\prod_{i=1}^{t-2} \frac{n-i}{n} > \left[\frac{n-(t-2)}{n}\right]^{t-2}. \tag{4.11}$$

Since n is large and $t \ll n$, the above probability expressed in (4.11) is in fact very large. For brevity, denote $p(t,n) = \left[\frac{n-(t-2)}{n}\right]^{t-2}$. Table 4.1 highlights the success rate that a random arrangement of passwords incorporation will result in a successful feasible chain for various password space in practical settings. This shows that any single random arrangement of password order will almost surely yield a feasible chain. As such, this is the more efficient way of chain construction.

4.7 Evaluations

Theorem 4.3.3 shows that any t given distinct passwords can be incorporated in a rainbow chain during the online phase construction. In this section, we provide theoretical comparisons of the online recovery phase between the average recovery time of frequently used passwords when they are incorporated along a chain and the recovery time when they are assigned at the start of the chains. Define t to be the length of rainbow chains in both instances. Let $T(t)$ denote the number of pairs of hash and reduction operations required before a successful recovery for passwords assigned at the start of chains. Let $T'(t)$ denote the average number of pairs of hash and reduction operations required before a successful recovery for passwords incorporated along a chain.

$$T(t) \approx 1 + 2 + 3 + \cdots + t = \frac{1}{2}t(t+1).$$

To compute $T'(t)$, we need to consider the cases when the password is located in position i counting from the next to last endpoint of the chain $\forall\ 1 \leq i \leq t$. In particular, position t corresponds to the start of chain while position 1 corresponds to the location of password just before the end of the chain. Then the number of pairs of hashed, reduced operations required for a password at position i can be approximated by

$$1 + 2 + 3 + \cdots + i + (t - i) = \frac{1}{2}i(i-1) + t.$$

Since the desired password to be recovered is equally likely to be each position of the chain, hence $T'(t)$ can be approximated by the following

$$T'(t) \approx \frac{1}{t}\sum_{i=1}^{t}(t + \frac{1}{2}i(i-1)) = \frac{1}{6}(t^2 + 6t - 1). \tag{4.12}$$

Clearly, $T'(t) < T(t)$, $\forall\ t > 1$. Moreover,

$$\lim_{t\to\infty}\frac{T'(t)}{T(t)} = \frac{1}{3}.$$

Therefore for large chain lengths, the time taken to recover a commonly used password is $\frac{1}{3}$ that of the time required when such passwords are assigned at the start points of chains. This translates to a 66.7% decrease in recovery time for long chains.

4.7.1 Partial Chain Flexibility

In cases where the password space is enormous, typical chain lengths exceed over ten thousand. In such instances, a user might only require only a partial of a chain to be incorporated with frequent passwords. For example, for a chain length of 10000, a user might want to ensure that 5000 of the most frequently used passwords can be guaranteed a successful recovery. We also provide an analysis of our improvements arising in such situations.

Let $T(n,t)$ denote the number of pairs of hash, reduction operations required before a successful recovery for passwords assigned at the start of chains. Let $T'(n,t)$ denote the average number of pairs of hash, reduction operations required before a successful recovery for passwords assigned along the start of chains. Here, n refers to the number of frequent passwords included at the start of the chain and t refers to the length of the the chain. Then, we have that

$$T(n,t) = T(t) \approx \frac{1}{2}t(t+1)$$

while

$$T'(n,t) \approx \frac{1}{n}\sum_{i=t-n+1}^{t}(t+\frac{1}{2}i(i-1))$$

$$= \frac{1}{n}\sum_{i=1}^{t}(t+\frac{1}{2}i(i-1)) - \frac{1}{n}\sum_{i=1}^{t-n}(t+\frac{1}{2}i(i-1))$$

$$= t - \frac{1}{6} + \frac{1}{6n}[t^3-(t-n)^3]$$

$$\approx t + \frac{1}{6n}[t^3-(t-n)^3].$$

4.7.2 Generalizing Partial Chain Improvements

We wish to evaluate the improvements of utilizing the partial chains over conventional methods. We establish the percentage improvements with respect to the ratio $\frac{n}{t}$. First, we show that that such partial chain construction result in more efficient recovery. Recall that n, t are positive integers and that $n < t$.

Corollary 4.7.1 *For all $n > 1$,*

$$T'(n,t) < T(n,t)$$

Proof. By straightforward computations,

$$n > 1 \Longrightarrow 3n(n-1) - n^2 > 0$$

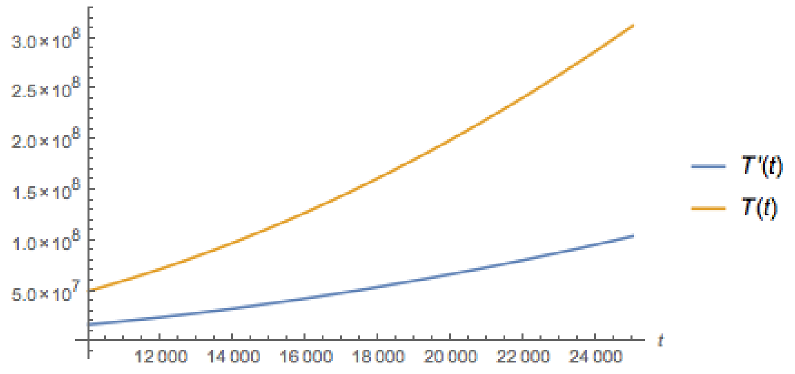

Figure 4.1: Number of hash evaluations against chain length

$$\implies 3t(n-1) - n^2 > 0$$
$$\implies t + \frac{1}{6n}[t^3 - (t-n)^3] < \frac{1}{2}t(t+1)$$
$$\implies T'(n,t) < T(n,t)$$

□

Percentage decrease over the conventional method is given by

$$1 - \frac{T'(n,t)}{T(n,t)} \approx \frac{n}{t} - \frac{1}{3}\left(\frac{n}{t}\right)^2.$$

4.8 Results

We highlight the results of our technique in this section. In Figure 4.1, the orange curve represents the usual method of inserting frequent passwords at the start of chains while the blue curve represents our improvements. In Figure 4.2, the chain length is varied together with the number of common passwords to be recovered. The orange surface represents the usual method while the blue surface represents our improvements. As can be seen, our method results in significant improvements with regards to the number of hashes needed to be computed before a successful recovery. This directly translates to a corresponding reduction in recovery time. Tables 4.2 and 4.3 highlight the explicit improvements.

Table 4.2 shows that for typical large chain lengths, the decrease in the number of hash evaluations required is about 66.7%. Table 4.3 shows that

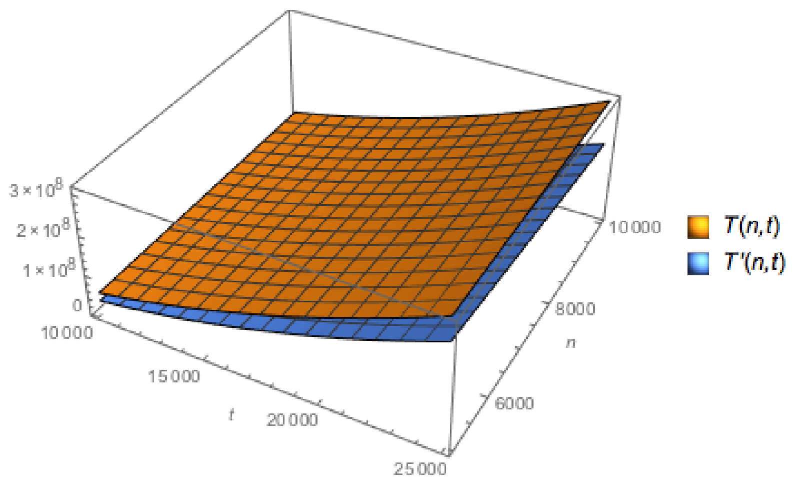

Figure 4.2: Number of hash evaluations against chain length and number of passwords

Table 4.2: Percentage decrease in the number of hash computations

t	$T(t)$	$T'(t)$	% decrease
10000	5.0×10^7	1.668×10^7	66.65%
15000	1.125×10^8	3.75×10^7	66.656%
20000	2.0×10^8	6.669×10^7	66.658%
25000	3.125×10^8	1.042×10^8	66.66%

Table 4.3: Percentage decrease in the number of hash computations

n	t	$T(n,t)$	$T'(n,t)$	% decrease
5000	6000	1.8×10^7	7.173×10^6	60.15%
10000	12000	7.2×10^7	2.868×10^7	60.17%
15000	18000	1.62×10^8	6.452×10^7	60.17%
20000	24000	2.88×10^8	1.147×10^8	60.17%

when the number of frequently used passwords inserted is $\frac{5}{6}$ of the chain length, the decrease in the number of hash evaluations required is about 60%.

4.9 Conclusion

As the success rate of inverting a given hash using rainbow tables is not 100%, it is inevitable that such frequently used passwords cannot be recovered aside from performing an exhaustive search. This can be resolved by assigning them to the start points of rainbow chains. We prove that it is also possible to incorporate such frequently used passwords along a rainbow chain. This can be achieved regardless of the type of hash applied. Moreover, we proceed to show that this method of incorporation enables a faster online recovery time as opposed to a canonical way of assigning them at the start of chains. For long chains, the online running time to recover these passwords is on average shortened to a factor of $\frac{1}{3}$. We also show our technique also offers flexibility improvements when the passwords are incorporated only along a fraction of the chain. In the case when $\frac{5}{6}$ of the rainbow chain is incorporated with given passwords, our result translates to an improvements of about 60% decrease in recovery time.

Chapter 5

Concluding Remarks

The bound for the generic hardness of discrete logarithm problem was first derived by Shoup. Kuhn and Struik then obtained lower bounds for solving 1 out of n instances of the discrete logarithm problem based on that result. Yun provided an extended result of Shoup in obtaining the hardness bound for the multiple discrete logarithm problem. Our work can be regarded as a generalization of the results by Yun. Figure 5.1 provides a summary of known bounds for various settings of the classical discrete logarithm problem. Our techniques are also applicable to obtain similar bounds for other variations of the DLP.

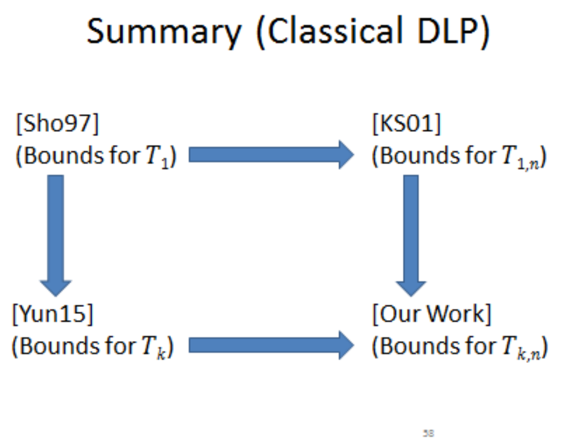
Summary (Classical DLP)
[Sho97] (Bounds for T_1)
[KS01] (Bounds for $T_{1,n}$)
[Yun15] (Bounds for T_k)
[Our Work] (Bounds for $T_{k,n}$)
38

Figure 5.1: Known bounds for the classical DLP

Splitting systems are useful tools to tackle types of low Hamming weight discrete logarithm problems. Coppersmith splitting system is particularly effective at solving the standard low Hamming weight discrete logarithm problem. However, CSS is less efficient at handling DLP with LHWP exponents. Kim and Cheon introduced PSS to obtain better results for this particular type of problem. We introduce PDSS and show that for parameters satisfying certain conditions required by PDSS, improved results over PSS can be achieved. We characterize certain classes of conditions required by PDSS to obtain such improvements. It will be interesting to consider if other characterizations exist which can also similarly offer improvements over PSS. Apart from cryptographic applications, this might also be of independent interest in discrete mathematics and Combinatorics.

The introduction of Hellman tables has enabled the computations of preimages of one way functions like cryptographic hash functions via a time-memory trade-off approach. The subsequent development of rainbow tables has since given rise to improved trade-off efficiencies. One particular important application of such rainbow tables is the decryption of password hashes. Rainbow tables serve as a powerful tool to uncover the passwords inputs of given cryptographic hash digests. However, an inherent feature of rainbow tables is that this method does not ensure a successful recovery. In this thesis, we demonstrate that certain sets of predetermined passwords can in fact be incorporated during the generation of the rainbow tables. Furthermore, we showed that this design of rainbow tables provide an advantage over earlier works.

Bibliography

[1] Digital signature standard (DSS). NIST (National Institute of Standards and Technology) FIPS, 186-4 (2013)

[2] National Institute of Standards and Technologies. Digital Signature Standard (DSS). Federal Information Processing Standards, Publication 186, November 1994

[3] Agnew, G., Mullin, R., Onyszchuk, I.,Vanstone, S.: An Implementation for a Fast Public-Key Cryptosystem. In: J. Cryptology, vol. 3, no. 2, pp. 63-79 (1991)

[4] Avoine, G., Carpent, X.: Optimal Storage for Rainbow Tables. In Information Security and Cryptology–ICISC 2013 (pp. 144-157). Springer International Publishing (2014)

[5] Avoine, G., Junod, P., Oechslin, P.: Time-memory trade-offs: false alarm detection using checkpoints. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 183-196. Springer, Heidelberg (2005)

[6] Avoine, G., Junod, P., Oechslin, P.: Characterization and improvement of time-memory trade-off based on perfect tables. ACM Trans. Inf. Syst. Secur. 11(4), 1-22 (2008)

[7] Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 1-16. Springer Berlin Heidelberg (2014)

[8] Barbulescu, R., Pierrot, C.: The multiple number field sieve for medium and high characteristic finite fields. In: LMS Journal of Computation and Mathematics, 17(A), 230-246 (2014)

[9] Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Public Key Cryptography, pp. 207-228. Springer Berlin Heidelberg (2006)

[10] Bernstein. D.J., Lange, T., Schwabe, P.: On the correct use of the negation map in the Pollard Rho method. In: Public Key Cryptography, pp. 128-146 (2011)

[11] Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 1-18. Springer, Heidelberg (2001)

[12] Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223-238. Springer, Heidelberg (2004)

[13] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56-73. Springe, Heidelberg (2004)

[14] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258-275. Springer, Heidelberg (2005)

[15] Cheon J.H., Hong J., Kim, M.: Accelerating Pollard's Rho algorithm on finite fields. In: J. Cryptol. **25**(2), 195-242 (2012)

[16] Contini, S., and Yin, Y. L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. Annual International Conference on the Theory and Application of Cryptology and Information Security (AsiaCrypt), Lecture Notes in Computer Science, 4284(1), 37-53 (2006)

[17] Coppersmith, D., Seroussi, G.: On the Minimum Distance of Some Quadratic Residue Codes. In: IEEE Trans. Inform. Theory 30, pp. 407-411 (1984)

[18] Coron, J., Lefranc, D., Poupard, G.: A New Baby-Step Giant-Step Algorithm and Some Application to Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 47-60. Springer, Heidelberg (2005)

[19] Diffie, W., Hellman, M.E.: New directions in cryptography. In: IEEE Trans. Inf. Theory **22**(6), 644-654 (1976)

[20] El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithm. In: IEEE Trans. Inf. Theory **31**(4), 469-472 (1985)

[21] Fouque, P. A., Joux, A., Mavromati, C.: Multi-user Collisions: Applications to Discrete Logarithm, Even-Mansour and PRINCE. In: Advances in Cryptology-ASIACRYPT 2014 (pp. 420-438). Springer Berlin Heidelberg (2014)

[22] Fouque, P. A., Leurent, G., and Nguyen, P. Q.: Full key recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. Advances in Cryptology, Lecture Notes in Computer Science, 4622(1), 13-30 (2007)

[23] Girault, M.: Self-Certified Public Keys. In: Davies, D.W. (eds.) Eurocrypt 1991. LNCS, vol. 547, pp. 490-497. Springer-Verlag (1991)

[24] Girault, M., Lefranc, D.: Public Key Authentications with one Single (on-line) Addition. In: Joyce M., Quisquater, J.J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 413-427. Springer-Verlag (2004)

[25] Girault, M., Poupard, G., Stern J.: Some Modes of Use of the GPS Identification Scheme. In: 3rd Nessie Conference. Springer, Heidelberg (2002)

[26] Heiman, R.: A Note on Discrete Logarithms with Special Structure. In: Rueppel, R.A. (eds.) EUROCRYPT 1992. LNCS, vol. 658, pp. 454-457. Springer, Heidelberg (1993)

[27] Hellman, M.: A cryptanalytic time-memory trade off. IEEE Trans. Inf. Theory IT-26(4), 401-406 (1980)

[28] Hitchcock, Y., Montague, P., Carter, G., Dawson, E.: The efficiency of solving multiple discrete logarithm problems and the implications for the security of fixed elliptic curves. International Journal of Information Security **3**(2), 86-98 (2004)

[29] Hong, J.: The cost of false alarms in Hellman and rainbow tradeoffs. Designs, Codes and Cryptography, 57(3), 293-327 (2010)

[30] Hong, J., Moon, S.: A comparison of cryptanalytic tradeoff algorithms. Journal of cryptology, 26(4), 559-637 (2013)

[31] Jaeger, D., Graupner, H., Sapegin, A., Cheng, F., Meinel, C.: Gathering and analyzing identity leaks for security awareness. In: Pre proceedings Passwords'14, Trondheim, Norway (2014)

[32] Joux, A.: A new index calculus algorithm with complexity $L(1/4+o(1))$ in very small characteristic. In: Selected Areas in Cryptography–SAC 2013 (pp. 355-379). Springer Berlin Heidelberg (2013)

[33] Kim S., Cheon, J.H.: A Parameterized Splitting System and its Application to the Discrete Logarithm Problem with Low Hamming Weight Product Exponents. In: PKC 2008. LNCS, vol. 4939, pp. 328-343 (2008)

[34] Kim, S., Cheon, J.H.: Parameterized splitting systems for the discrete logarithm. In: IEEE Transactions on Information Theory, 56(5), 2528-2535 (2010)

[35] Kim, T.: Multiple Discrete Logarithm Problems with Auxiliary Inputs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 174-188. Springer Berlin Heidelberg (2015)

[36] Kuhn, F., Struik, R.: Random walks revisited: Extensions of Pollard's Rho algorithm for computing multiple discrete logarithms. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 212-229. Springer, Heidelberg (2001)

[37] Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, pp. 128. CRC press, Boca Raton (1997)

[38] Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: Cracking UNIX passwords using FPGA platforms. In: SHARCS – Special Purpose Hardware for Attacking Cryptographic Systems (2005)

[39] Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **85**(2), 481-484 (2002)

[40] Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617-630. Springer, Heidelberg (2003)

[41] Papoulis, A., Pillai, S. U.: Probability, random variables, and stochastic processes. Tata McGraw-Hill Education (2002)

[42] Pollard, J.: Monte Carlo methods for index computations mod p. In: Math. Comput., **32**(143), 918-924 (1978)

[43] Poupard G., Stern J.: Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In: Nyberg K. (eds.) Eurocrypt 1998, vol. 1403. LNCS, pp. 422-436. Springer-Verlag (1998)

[44] Saarinen, M.-J.O.: A time-memory tradeoff attack against LILI-128. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 231-236. Springer, Heidelberg (2002)

[45] Saran, N., Doğanaksoy, A.: Choosing parameters to achieve a higher success rate for Hellman time memory trade off attack. In Availability, Reliability and Security, 2009. ARES'09. International Conference on (pp. 504-509) IEEE (2009)

[46] Sasaki, Y., Wang, L., Ohta, K., and Kunihiro, N.: Security of MD5 challenge and response: Extension of APOP password recovery attack. The Cryptographers Track at the RSA Conference on Topics in Cryptology, 4964(1), 1-18 (2008)

[47] Sasaki, Y., Yamamoto, G., and Aoki, K.: Practical password recovery on an MD5 challenge and response. Cryptology ePrint Archive, Report 2007/101 (2007)

[48] Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard G. (eds.) CRYPTO 1989. LNCS, vol. 435, pp. 239-252. Springer-Vaerlag (1990).

[49] Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256-266. Springer, Heidelberg (1997)

[50] Stinson, D.: Some Baby-Step Giant-Step Algorithms for the Low Hamming Weight Discrete Logarithm Problem. Mathematics of Computation 71(237), pp. 379-391 (2002)

[51] Teske, E.: On random walks for Pollard's Rho method. In: Math. Comput. **70**, 809-825 (2000)

[52] Thing, V. L., Ying, H. M.: A Novel Time-Memory Trade-Off Method for Password Recovery. Digital Forensics Research Conference (DFRWS), Montreal, Quebec, Canada, August 2009. In: Digital Investigation - International Journal of Digital Forensics and Incident Response, Elsevier, Vol. 6, Supplement, pp. S114-S120 (2009)

[53] Thing, V. L., Ying, H. M.: Rainbow Table Optimization for Password Recovery. In: International Journal on Advances in Software, Vol. 4, No. 3&4, pp. 479-488 (2012)

[54] Thing, V. L., Ying, H. M.: Enhanced Dictionary Based Rainbow Table. In: 27th IFIP International Information Security and Privacy Conference (SEC) (pp. 513-524) (2012)

[55] Thing, V. L., Ying, H. M.: Password Recovery Research and Its Future Direction. In: Integrated Information and Computing Systems for Natural, Spatial, and Social Sciences, Chapter 10, pp. 192-206, ISBN 978-1-466-62190-9, IGI-Global (2012)

[56] Tihanyi, N., Kovacs, A., Vargha, G., Lenart, A.: Unrevealed patterns in password databases Part one: analyses of cleartext passwords. In: Pre proceedings Passwords'14, Trondheim, Norway (2014)

[57] Trainin, J.: An elementary proof of Pick's theorem. In: Mathematical Gazette. **91** (522): 536–540 (2007)

[58] Yun, A.: Generic Hardness of the Multiple Discrete Logarithm Problem. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 817-836. Springer Berlin Heidelberg (2015)

List of Publications

International Conferences (Peer-Reviewed)

[1] Ying, J.H.M., Kunihiro, N.: Bounds in Various Generalized Settings of the Discrete Logarithm Problem. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds) Applied Cryptography and Network Security. ACNS 2017. LNCS, vol. 10355, pp. 498-517. Springer, Cham (2017)

[2] Ying, J.H.M., Kunihiro, N.: Solving the DLP with Low Hamming Weight Product Exponents and Improved Attacks on the GPS Identification Scheme. In: Pieprzyk, J., Suriadi, S. (eds) Information Security and Privacy. ACISP 2017. LNCS, vol. 10343, pp. 460-467. Springer, Cham (2017)

[3] Ying, H.M., Kunihiro, N.: Decryption of Frequent Password Hashes in Rainbow Tables. In: Computing and Networking (CANDAR), 2016 Fourth International Symposium on, pp. 655-661. IEEE. (2016)

[4] Ying, H.M., Kunihiro, N.: Cold Boot Attack Methods for the Discrete Logarithm Problem. In: Computing and Networking (CANDAR), 2016 Fourth International Symposium on, pp. 154-160. IEEE. (2016)

Domestic Conferences (Non Peer-Reviewed)

[5] Ying, H.M., Kunihiro, N.: On the Complexity Complexity of the DLP with Low Hamming Weight Product Exponents. In: SCIS (2017)

[6] Ying, H.M., Kunihiro, N.: On the Complexity in Certain Classes of Multiple Discrete Logarithms. In: SCIS (2016)