

1

Introduction

Passwords have been in use since antiquity to allow certain people access to some exclusive resource. A set of people who are allowed access to the resource, called legitimate users, are first selected. They are then given passwords (possibly different) which are meant to be kept secret. When attempting to access a resource these users are asked to reproduce the password. This process is known as password-based authentication and is a form of user authentication or user identification. The use of passwords has carried on through the digital era whereby people are granted access to their computers or other digital resources.

Of course, nowadays passwords are not the only way to authenticate a user. In general, user authentication methods can be divided into three categories [1]; knowledge-based authentication or authentication based on what a person knows, e.g., passwords, token-based authentication or authentication based on what a person possesses, e.g., smart cards, and characteristics-based authentication or authentication based on what a person is, e.g., biometrics. Knowledge-based authentication can be further classified into weak and strong authentication [1, 2]. Textual or graphical passwords can be considered as weak forms of authentication. On the other hand, one-time passwords, zero-knowledge identification protocols, and other challenge-response identification protocols are classified as strong authentication methods. Evidently, weak authentication is prone to eavesdropping. Strong authentication aims to make it harder for an eavesdropper to obtain the secret and consequently impersonate the user.

One-time passwords were first introduced by Lamport in [3], who proposed to successively apply a one-way function on an initial password. Each intermediate value of the one-way function serves as a one-time password. As the name implies, each of these passwords have to be used only once. Zero-knowledge identification protocols such as the Fiat-Shamir identification protocol [4], enable a prover (the user) to prove his identity to a verifier. These protocols do not involve the sharing of any secrets, and at the end of the protocol the identity of the legitimate user can be proved with

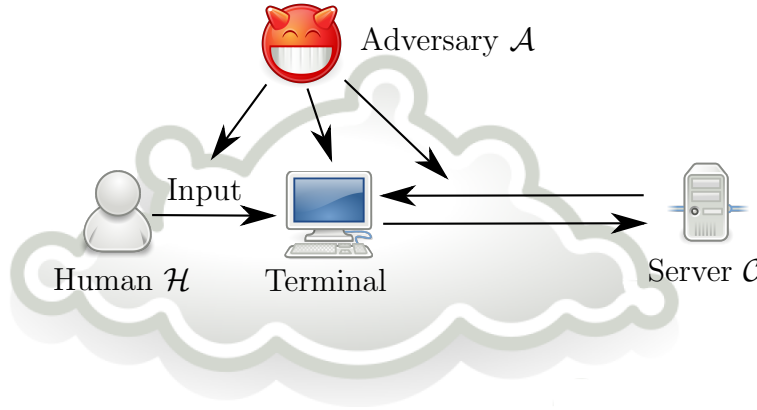


FIGURE 1.1: Authentication under Matsumoto and Imai's threat model.

overwhelmingly high probability without revealing *any* information about the secret (a zero-knowledge proof); hence the name zero-knowledge. An example of challenge-response identification protocols is identification through symmetric-key encryption [2, §10, p. 400]. In such protocols, the prover and the verifier share a secret. The verifier sends a random message to the prover, who encrypts it with a block cipher using the shared secret, which serves as the key to the cipher. The prover sends the resulting ciphertext to the verifier. Since the verifier possesses the same secret, verification of the ciphertext can be done straightforwardly. Although theoretically, these strong authentication methods authenticate a user based on what he knows, the computations involved are hard for a human and thus a user has to possess a secure computational device to do all the computations.

Even though strong authentication mitigates some of the security problems with weak authentication, all the aforementioned authentication methods have a number of weaknesses associated with them when examined under a specific threat model. To understand this, consider a scenario sketched in Figure 1.1.

Matsumoto and Imai's Threat Model

Suppose a user wants to authenticate himself¹ to a remote server. He is given a computer terminal which he uses to authenticate himself to the server in the presence of an adversary. The adversary has access to the terminal and the communication link between the terminal and the server. To make matters worse, the adversary can also observe the user's actions, including his use of any input/output device (such as a mouse or a keyboard). Is it possible to securely authenticate the user in this environment? Informally, secure authentication means that the adversary should not be able to impersonate the user even after observing the interaction between the user and the server. This problem scenario was conceived by Matsumoto and Imai [5], and will therefore be referred to as Matsumoto and Imai's threat model.

¹Or herself. Any subsequent use of a gender-specific pronoun in this thesis can similarly be interchanged with the corresponding pronoun of the opposite gender.

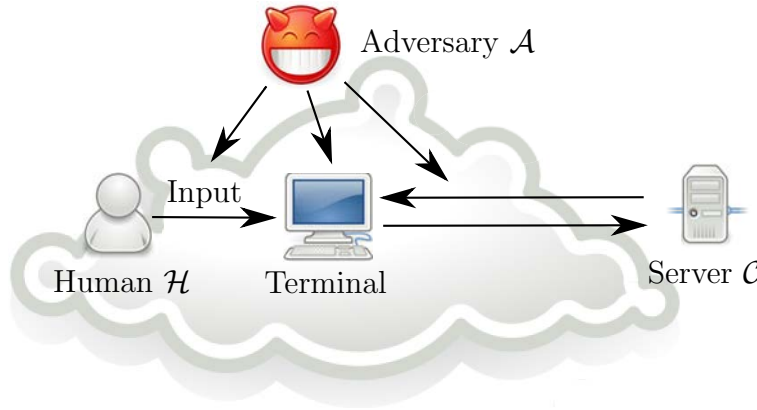


FIGURE 1.1: Authentication under Matsumoto and Imai's threat model.

overwhelmingly high probability without revealing *any* information about the secret (a zero-knowledge proof); hence the name zero-knowledge. An example of challenge-response identification protocols is identification through symmetric-key encryption [2, §10, p. 400]. In such protocols, the prover and the verifier share a secret. The verifier sends a random message to the prover, who encrypts it with a block cipher using the shared secret, which serves as the key to the cipher. The prover sends the resulting ciphertext to the verifier. Since the verifier possesses the same secret, verification of the ciphertext can be done straightforwardly. Although theoretically, these strong authentication methods authenticate a user based on what he knows, the computations involved are hard for a human and thus a user has to possess a secure computational device to do all the computations.

Even though strong authentication mitigates some of the security problems with weak authentication, all the aforementioned authentication methods have a number of weaknesses associated with them when examined under a specific threat model. To understand this, consider a scenario sketched in Figure 1.1.

Matsumoto and Imai's Threat Model

Suppose a user wants to authenticate himself¹ to a remote server. He is given a computer terminal which he uses to authenticate himself to the server in the presence of an adversary. The adversary has access to the terminal and the communication link between the terminal and the server. To make matters worse, the adversary can also observe the user's actions, including his use of any input/output device (such as a mouse or a keyboard). Is it possible to securely authenticate the user in this environment? Informally, secure authentication means that the adversary should not be able to impersonate the user even after observing the interaction between the user and the server. This problem scenario was conceived by Matsumoto and Imai [5], and will therefore be referred to as Matsumoto and Imai's threat model.

¹Or herself. Any subsequent use of a gender-specific pronoun in this thesis can similarly be interchanged with the corresponding pronoun of the opposite gender.

A real world example of this model can be visualized as a user attempting to authenticate to a remote server using a computer terminal in a public internet cafe. The terminal can potentially be infected by malware, such as trojan horses and key-loggers. There can be an added risk of shoulder-surfers peeping at the user's input. Hidden cameras, network snoops and spoof websites can complete a realization of Matsumoto and Imai's threat model. Criminals have realized this threat model in many ways to attack different authentication systems. An example is authentication to an Automated Teller Machine (ATM). Typically, ATM authentication involves swiping of a bank card and entering of a Personal Identification Number (PIN). Criminals have used a technique known as *card skimming* to impersonate an ATM user [6]. First, they place an unobtrusive magnetic stripe reader on the same place where an ATM card is swiped [6]. This stores the information contained in the card locally, which can be retrieved later. To obtain the PIN, they also place a small hidden camera which records the user's PIN entry [6]. The card is then easily cloned through magnetic stripe writers.

Both the customers and the bank can lose large sums of money due to a successful card skimming attack. Furthermore, even if countermeasures are taken to mitigate this specific attack, through the use of electromagnetic fields [6] for instance, criminals are coming up with new and innovative ways to attack ATM authentication and hence impersonate users. Indeed, recently more sophisticated ways have been used, such as malware installed into ATMs [6]. The malware functions the same way as a magnetic stripe reader, i.e., it gathers and stores users' information which can be retrieved later. This situation is of course not specific to ATM authentication, and other user authentication methods not secure under Matsumoto and Imai's threat model are, at least in principle, prone to innovative attacks. Matsumoto and Imai's threat model is thus a real-world model, and if an authentication system is not secure under this model various real world attacks can be mounted on it to impersonate legitimate users. If it is shown that an authentication system is secure under this threat model then it will be secure against most, if not all, kinds of attacks mentioned above.

Challenge-Response Authentication

It is evident that traditional authentication methods, such as password or PIN based authentication, are not secure under this threat model since the adversary can impersonate the user after observing a single session. More elaborate traditional authentication methods discussed earlier are not sufficiently secure either. Knowledge-based strong authentication methods such as one-time passwords and zero-knowledge identification protocols require the user to possess a *secure* device, and without such a device the computations involved are too hard for humans to perform on their own. Characteristics-based authentication methods do not fare much better either. For instance, biometrics-based authentication supposes that the biometrics sensing equipment cannot be accessed and hence the data cannot be read by the adversary. Additionally, compared to passwords or bank cards (tokens), biometrics have more privacy concerns. Unlike knowledge or token-based authentication, this type of authentication uses data that is characteristic of a user, which once compromised cannot be reissued

and can lead to unwanted consequences such as tracking of the user's behavior [7]. However, considerable research is being done to overcome these privacy problems [7], and it is an interesting area of research. Indeed a system that utilizes all three types of authentication (multi-factor authentication) is potentially more secure, and research in all three fronts can lead to a better solution to the problem of user authentication. The focus of this thesis is mainly on the first type of authentication, i.e., authentication based on what a user knows, or knowledge-based authentication. In what follows, the term authentication (or identification) will imply this type of authentication.

Most, if not all, knowledge-based authentication systems can be classified as some type of challenge-response authentication. In challenge-response authentication, the server asks a challenge (or a series of challenges) to the user, who responds to the challenge according to his secret. For example, password-based authentication can be seen as challenge-response authentication. The challenge is the server's prompt for a password, and the response is the user's password itself. Of course, as already mentioned, password-based authentication is only secure if the adversary does not observe a *single* response. In general, the security of a challenge-response authentication system can be measured by the number of observed challenge-response *pairs* the adversary requires to successfully impersonate the user. Design and analysis of challenge-response type authentication schemes secure under Matsumoto and Imai's threat model is the main focus of this thesis.

The hitherto discussed knowledge-based authentication methods that can be classified as challenge-response protocols are either too weak (e.g., passwords) or too computationally intensive (e.g., zero-knowledge protocols, identification using symmetric-key encryption, etc.) under Matsumoto and Imai's threat model. A candidate challenge-response protocol should preserve the ease of use of passwords, with security guarantees similar to strong authentication methods. The literature on this important area of research is sparse and progressive work is sporadic. Worse still, there is no accepted solution. The main hurdle is the usual difficulty in finding an acceptable balance between security and usability, which is heightened in the aforementioned threat model. Note that to be secure under this threat model, the adversary should not be able to see any information other than the challenge-response pairs being communicated. This implies that if the authentication method requires any computations from the user, these have to be done mentally. Recall that in a password-based system, the only computation required from the user is *recalling* the password. More sophisticated challenge-response authentication methods inevitably require more complex computations, and the undesired consequence is a longer login time. The goal is to make this time as close as possible to the case of password-based authentication.

Human Identification Protocols

There are several terms coined for these challenge-response authentication systems. Matsumoto and Imai [5], and Hopper and Blum [8] have used the term human identification protocols. Elsewhere they have been referred to as shoulder-surfing resistant graphical passwords [9]. The term 'graphical password' alludes to graphical implementation of some of these challenge-response protocols and is not all-encompassing. We

will show later that all such protocols can be implemented graphically. The other term, shoulder-surfing, refers to the act of observing the user’s interaction with the terminal either directly or through a hidden camera. Since the term ‘graphical password’ is somewhat of an oxymoron, it is sometimes replaced with terms like pass-icons or pass-pictures depending on whether the implementation is based on software icons or pictures, respectively. Some other terms used in literature are cognitive authentication schemes [10], virtual passwords [11], etc.

It is argued in this thesis that the use of these different terms wrongly suggests that the referred protocols target secure authentication under different threat models. As we will show in the next chapter, the difference is nominal and most of these protocols can be grouped under one umbrella term. The term ‘user identification protocols’ does not seem to be appropriate either, since it can also mean those identification protocols in which the user’s device, instead of the user himself, is actually authenticated to the server; a la zero-knowledge identification protocols. The terms identification and authentication are considered synonyms in this thesis, and in accordance with the adopted term *human identification protocols*, identification instead of authentication shall be used more frequently. Thus, human identification protocols are protocols (authentication systems) that are designed to help a human authenticate to a server under Matsumoto and Imai’s threat model. It is important to note that this term does not reflect any light on the security of a protocol. A protocol satisfies the definition of a human identification protocol even if it is only secure for one observed session, such as password-based authentication.

Passive Adversaries

Human identification protocols considered in this thesis target security against passive as opposed to active adversaries. An active adversary has the additional capability of actively interfering with the communication link between the user’s terminal and the server (see Figure 1.1). More precisely, an active adversary can also insert, modify or delete messages. Although human identification protocols secure against active attacks is coveted, it is extremely hard to construct one that ensures both acceptable security and good human executability. To date there have been a handful of proposals known to resist some active attacks [8, 12–14]. Among them only the *sum of k mins* protocol proposed in [8] has been constructed to be secure against generic active adversaries; yet, the protocol falls short of usability. The rest of the protocols only treat security against a known set of active attacks. Due to the difficulty of constructing usable protocols secure against active attacks, recently the focus of the research community has been on security against passive (eavesdropping) adversaries [10, 15–18].

Despite this being a weaker threat model, there is still no widely accepted human identification protocol secure against passive adversaries and this remains an open problem. This weaker threat model is by no means an entirely unnatural model. Mounting of active attacks can be made harder by employing other techniques. For instance, one way to detect active attacks is through the use of digital signatures. Each time the server sends a challenge to the user, a signature of the challenge (and a timestamp) is also sent. The signature can be verified through a third device, such as

the user's mobile phone. This can be achieved seamlessly without invoking the user. The user is only prompted once a challenge does not match the signature. Once an active attack is detected, the user's secret can be renewed, and further action can be taken by the network administrator. Note that the user's mobile phone is not being used to authenticate him to the server. Instead, it is just used for authenticating the source of the challenge. Thus, we argue that it is not at all unnatural to restrict the focus to passive adversaries.

Why Human Identification Protocols?

It is clear that a solution to the problem of identifying a human under Matsumoto and Imai's threat model has the potential to solve many security issues with current state of user authentication, and hence it is important to further the research in human identification protocols. While it seems improbable that a solution which is both secure and takes time equivalent to password-based authentication can ever be found, research in this direction is interesting in many other ways. As will be evident later, different aspects have to be considered during the design of a human identification protocol. These touch different areas of scientific research, such as cryptography, human-computer interaction, and human psychology to name a few prominent ones. Research in each of these aspects can lead to better understanding of human computational abilities, and how this can be facilitated by interaction with a computer.

From the perspective of cryptography, it is interesting to know what mathematical functions can be efficiently computed by humans mentally, and what are the limitations. More specifically, since cryptography deals with constructing cryptosystems whose security is argued on the basis of the difficulty of solving some underlying mathematical problem, it is worth exploring the extent to which this approach can be applied in the design of human identification protocols. Matsumoto and Imai [5], Matsumoto [19], and Hopper and Blum [8] were the pioneers in trying to apply this approach of finding mathematical problems that can be used as building blocks for human identification protocols. Following this direction, this thesis focuses on the mathematical and theoretical aspects of human identification protocols.

While the real world implementation of these protocols have many different interesting aspects, we only briefly touch this to show that the protocols can in principle be implemented in a user friendly way. It is important to stress that both the design and implementation aspects need to be considered in a real world deployment of human identification protocols, but it is necessary to first consider the security of the protocol from an analytical point of view, taking a leaf out of research in mainstream cryptography. Research in this direction is worthwhile since it has the potential to improve our understanding of how an insecure computer terminal can be used to assist a human in computing mathematical functions that are hard for an adversary to solve without the knowledge of the secret.

Contributions of the Thesis

This thesis furthers the research in human identification protocol by first making a clear distinction between the theoretical and implementation aspects of such protocols. This distinction has not always been followed in the literature and some protocols have been constructed with mainly the implementation aspect in mind. Often, such protocols succumb to simple but innovative attacks. This is illustrated later in this thesis with a detailed security analysis of two human identification protocols proposed in the literature. The result of the analysis shows that these protocols are secure for a much smaller number of identification sessions than originally claimed. This analysis helps in the understanding of the taxonomy of attacks that are to be avoided in the construction of human identification protocols.

This leads to another major contribution of this thesis, i.e., the proposal of two human identification protocols designed to be secure against a passive adversary, with clearly defined goals in mind. The second protocol is based on a mathematical problem which is shown to be intractable in a relatively new complexity theoretic sense, i.e., fixed-parameter intractability [20]. This is the last main contribution of this thesis, the link between the area of fixed-parameter intractability and the design of human identification protocols. It is argued that one way to construct human identification protocols is to first find a mathematical problem that is fixed-parameter intractable, and then construct a protocol based on this problem. In fact, we show that many protocols in literature are actually based on such problems or at least conjectured to be based on such problems, without their inventors discovering the link with the theory of fixed-parameter intractability. This helps in establishing hardness of problems used in human identification protocols by analyzing them under the complexity theoretic framework.

While the two protocols proposed in this thesis can be efficiently implemented, and examples are given as such, the research focus of this thesis is not efficient implementation. Substantial improvements can be made in the implementation of the protocols through user studies and by studying user behavior. However, the focus of this research is mainly on the mathematical and analytical aspects; the underlying theory behind the construction of human identification protocols. As mentioned before, often times the distinction between analytical and implementation aspects of these protocols is blur. The result is that some of these protocols offer similar security to traditional methods (such as one-time passwords) with decreased usability. This thesis also attempts to clarify this distinction, in the hope that future research into this area should consider the mathematical aspects in great detail before coming up with a proposed solution. Indeed, the main focus of an authentication protocol is security, with usability considerations in mind; but it should not be the other way around.

Outline of the Thesis

This thesis can be roughly divided into four main parts. The first part is background; a formal introduction to human identification protocols. Chapter 2 constitutes this part. It introduces the notation, definitions and general results that shall be used

throughout the rest of this thesis. The chapter formally defines the concepts alluded to in this introduction, and contains a review of the related work in light of the notions thus developed. Chapter 2 also contains a comprehensive discussion of an example human identification protocol which will be helpful in understanding the issues related to the design and implementation of these protocols. Chapters 3 and 4 make up the second part, security analysis of human identification protocols. This part analyzes the security of two human identification protocols, namely Bai et al.'s Predicate based Authentication Service (PAS) [17] and, Sobrado and Birget's Convex Hull Click (CHC) protocol [9, 21], and shows that these protocols are far less secure than originally claimed. The security analysis of these two protocols precedes the description of two proposed protocols in Chapters 5 and 6, which constitute the third main part of this thesis. Protocol construction succeeds cryptanalytic part of the thesis so that it is clear what sort of general attacks are to be avoided in construction of protocols. Chapter 7 is the last main part of the thesis, and contains some interesting theoretical results about the problems used in different human identification protocols. More specifically, the chapter briefly describes the theory of fixed-parameter intractability [20] and shows that many protocols, including the proposed protocol in Chapter 6, are based on fixed-parameter intractable problems. Chapter 8 concludes this thesis with emphasis on the limitations of this work and some future directions.

2

Preliminaries and Related Work

This chapter begins with the introduction of the general notation used in this thesis. While most of the notation introduced here is consistent throughout the text, to avoid excess some symbols are used with different meanings in different chapters. However, the meaning implied will be clear from the context, and where there is fear of ambiguity we will reiterate the intended meaning. Section 2.2 contains a list of definitions pertaining to human identification protocols. Some initial and general results applicable to most, if not all, human identification protocols are described in Section 2.3. In Section 2.4 we give an example of a human identification protocol, followed by a security and usability analysis. This will be helpful in understanding the design and implementation issues concerning the construction of human identification protocols. The chapter concludes with a thorough discussion of related work in light of the framework developed in the beginning of this chapter.

2.1 Notation

In accordance with convention we will call the two parties in an identification protocol, *prover* and *verifier*, where the prover is the party who wishes to prove his identity to a verifier. The prover shall be denoted by \mathcal{H} and the verifier by \mathcal{C} . This is to acknowledge that in the real world setting of human identification protocols, the prover is a human and the verifier, a remote computer. The adversary shall be denoted by \mathcal{A} .

Let S be a set. Any subset of S of s elements is called an s -element subset of S . $|S|$ denotes the number of elements of S . The set of all non-negative integers modulo an integer $d > 1$ is denoted \mathbb{Z}_d . $\mathbf{x} \in \mathbb{Z}_d^n$ denotes an n -tuple of elements from \mathbb{Z}_d , or equivalently a vector of n elements from \mathbb{Z}_d . Any vector \mathbf{x} will be considered a column vector unless otherwise specified. Let $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_d^n$, then $\mathbf{x} \cdot \mathbf{y}^T$ denotes the usual dot product of the vector \mathbf{x} and the transpose of the vector \mathbf{y} . Where there is no fear

of ambiguity, the superscript T shall be omitted, and the dot product will simply be denoted by $\mathbf{x} \cdot \mathbf{y}$. The Hamming weight of \mathbf{x} is the number of non-zero entries in \mathbf{x} . Denote the Hamming weight of \mathbf{x} by $\text{wt}(\mathbf{x})$. The vector $\mathbf{0}$ (resp. $\mathbf{1}$) denotes the vector of all 0's (resp. 1's). An n -tuple is defined as an ordered set of n elements. An n -tuple can be represented by a vector of n -elements in a natural way, and the two terms will be used interchangeably. Let a and b be non-negative integers, with $a < b$. The interval $[a, b]$ denotes the set $\{a, a+1, \dots, b\}$. The symbol \approx has the intuitive meaning 'approximately'. $\mathcal{O}(\cdot)$ is the usual Big O notation.

Many terms in thesis are treated as synonyms. We have already mentioned that identification and authentication define the same concept in this thesis. Likewise we shall often call the prover \mathcal{H} , a human, a human user, or simply, a user. The term attacker shall also be used at times as a synonym for adversary. We shall often call the verifier \mathcal{C} , the server, or the remote server. An attack is an algorithm that is designed to exploit weaknesses in a human identification protocol. We shall often refer to attacks as algorithms.

2.2 Human Identification Protocols

The definitions of identification protocols and human executable protocols stated here are taken from [8]. Informally, they first define an identification protocol in terms of an interaction between two interactive Turing machines (ITMs), the prover and the verifier. A human identification protocol is then defined as an identification protocol in which the steps from the prover can be performed by a human. To avoid excessive notation, we use the same symbol \mathcal{H} for the ITM as well as the human user. However, the implied meaning shall be clear from the context.

A protocol is defined as a sequence of interactions between a pair of public and probabilistic ITMs, denoted $(\mathcal{H}, \mathcal{C})$. The result of the interaction between these two ITMs with respective inputs x and y is denoted by $\langle \mathcal{H}(x), \mathcal{C}(y) \rangle$. The transcript of bits exchanged between \mathcal{H} and \mathcal{C} during this interaction is denoted by $\text{tr}(\mathcal{H}(x), \mathcal{C}(y))$.

Definition 1. *An identification protocol is a pair of public, probabilistic interactive Turing machines $(\mathcal{H}, \mathcal{C})$ with shared auxiliary input z (the secret), such that the following conditions hold:*

- *For all auxiliary inputs z , $\Pr[\langle \mathcal{H}(z), \mathcal{C}(z) \rangle = \text{accept}] \geq q$*
- *For each pair $x \neq y$, $\Pr[\langle \mathcal{H}(x), \mathcal{C}(y) \rangle = \text{accept}] \leq 1 - q$ where $0.5 < q \leq 1$.*

When $\langle \mathcal{H}, \mathcal{C} \rangle = \text{accept}$, it is said that \mathcal{H} authenticates to \mathcal{C} . Intuitively, the above definition states that a protocol is an identification protocol if a legitimate prover (one who shares the same secret as \mathcal{C}) can identify himself to \mathcal{C} with a probability greater than 0.5. On the other hand, any entity not possessing the secret can only succeed in impersonating \mathcal{H} with probability strictly less than 0.5. Thus, this definition also takes those protocols into account which even reject legitimate provers, albeit with a small probability. An example is the Hopper and Blum (HB) protocol from [8]. Note that this definition does not explicitly mention that \mathcal{H} is a human user.

Definition 2. *An identification protocol is a human identification protocol if the computations \mathcal{H} are done by a human mentally.*

In principle, any protocol whose steps can be performed by an unaided human can be called a human identification protocol. However, to be a reasonable candidate for a human identification protocol, the computational time per authentication session should be low enough. The following definition from [8] will be used to establish the usability of a protocol, which states that the computations done by the probabilistic Turing machine \mathcal{H} should be easy enough to be carried out by a human.

Definition 3. *An identification protocol $(\mathcal{H}, \mathcal{C})$ is (α, β, τ) -human executable if at least a $(1 - \alpha)$ portion of the human population can perform the calculations \mathcal{H} unaided and without errors in at most τ seconds with probability greater than $(1 - \beta)$.*

The goal is to minimize α , β and τ . Concrete values to these parameters can be assigned by either an intuitive approximation or where it is hard to do so, actual usability studies can be carried out [8]. Although this definition will be used to theoretically analyse the usability of protocols in this thesis, this should not be considered as a replacement for empirical user studies. Indeed, the parameters α , β and τ cannot be accurately estimated without a reliable theoretical model of human computation. In the absence of such a model, only empirical user studies can give an accurate indication of usability. We acknowledge this as a drawback, but shall use the definition nonetheless in order to at least show the comparative usability of the protocols.

As an exemplary use of these definitions, it is easy to see that the traditional password-based authentication system satisfies the definition of an identification protocol, since if \mathcal{H} and \mathcal{C} share the same password, then \mathcal{H} authenticates to \mathcal{C} with probability 1. Otherwise, \mathcal{H} is accepted with probability 0. Also, it can be conjectured that it is $(\approx 0, \approx 0, 5)$ -human executable for say a length 10 password, based on our everyday experience.

2.2.1 Challenge-Response Protocols

A challenge-response identification protocol is defined as the following sequence of messages communicated between \mathcal{H} and \mathcal{C} :

request, $(c_1, r_1), (c_2, r_2), \dots, (c_\mu, r_\mu)$, **accept/reject**

The message, **request**, is sent by \mathcal{H} to \mathcal{C} . It symbolizes a request to start a protocol and contains information about \mathcal{H} , such as his identity. We shall mostly omit explicit mention of the exchange of this message in subsequent descriptions of protocols. Each pair (c_i, r_i) consists of a challenge c_i drawn from some *challenge space* C by \mathcal{C} and a response r_i composed from a *response space* R by \mathcal{H} . \mathcal{H} and \mathcal{C} also share a secret from a *secret space* S . C , R and S are all publicly known sets. At the end of μ challenge-response pairs, \mathcal{C} sends the message **accept** or **reject**. This sequence of messages is called one *session* (or an identification/authentication session) of the challenge-response identification protocol. An iteration consists of one challenge-response pair. Each

session, therefore, has μ iterations, where $\mu \geq 1$. Let ρ be a subset of the ternary cartesian product $C \times S \times R$. In other words, ρ is a ternary relation. The challenge, response and the secret are related through the publicly known relation ρ . If a tuple $(c, s, r) \in \rho$, then the response is considered correct, otherwise it is considered wrong or incorrect.

2.2.2 Security Definitions

As mentioned before, the adversary \mathcal{A} considered in this thesis is a passive adversary. The goal of \mathcal{A} is to impersonate \mathcal{H} by initiating a new identification session with \mathcal{C} , after observing a few identification sessions between \mathcal{H} and \mathcal{C} . From a theoretical point of view, the adversary is given a transcript of communication between \mathcal{H} and \mathcal{C} , and is then allowed to play a game with \mathcal{C} trying to fool \mathcal{C} into accepting \mathcal{A} as \mathcal{H} . The success of \mathcal{A} is measured by the probability of successful impersonation. Notice that \mathcal{A} may not need to find the shared secret between \mathcal{H} and \mathcal{C} in order to impersonate \mathcal{H} . For instance, \mathcal{A} might be able to randomly guess a response to a challenge. More on this will be detailed later. The following definition, taken from [8], defines the security of an identification protocol against the passive adversary \mathcal{A} .

Definition 4. *An identification protocol $(\mathcal{H}, \mathcal{C})$ is (p, m) secure against passive adversaries if for all computationally bounded adversaries \mathcal{A} and for all auxiliary inputs z :*

$$\Pr [\langle \mathcal{A}(\text{tr}^m(\mathcal{H}(z), \mathcal{C}(z))), \mathcal{C}(z) \rangle = \text{accept}] \leq p$$

Here $\text{tr}^m(.,.)$ represents the transcript of m independent identification sessions between \mathcal{H} and \mathcal{C} .

If $\langle \mathcal{A}(\text{tr}^m(\mathcal{H}, \mathcal{C})), \mathcal{C} \rangle = \text{accept}$ for some m , it is said that \mathcal{A} impersonates \mathcal{H} . The above definition treats \mathcal{H} and \mathcal{C} as interactive Turing machines. In human identification protocols the computations done by \mathcal{H} have to be carried out by a human. Therefore, all such computations should be done by the human mentally, or else any security proved against \mathcal{A} will be superfluous. Recalling Matsumoto and Imai's threat model, the above definition does not explicitly mention that \mathcal{H} possesses an insecure computer terminal. But from a theoretical viewpoint, the definition describes the same model. That is, there is no further secure computation done at the terminal.

From a challenge-response protocol perspective, \mathcal{A} is given all challenge-response pairs from m identification sessions, i.e., a set of $m\mu$ challenge-response pairs. Unless otherwise mentioned, it is assumed that all such pairs belong to *successful* identification sessions. Abusing notation, from now onwards, m challenge-response pairs shall represent an arbitrary number of pairs. The security of a protocol shall then be evaluated against the number of challenge-response pairs m , required by \mathcal{A} such that the probability of impersonation is non-negligible. For the sake of human identification protocols, an impersonation probability p of 10^{-6} shall be considered sufficient for security [8]. This implies that \mathcal{A} has one in a million chance of impersonating \mathcal{H} . It is important to reiterate that the only information unknown to \mathcal{A} is the shared secret.

2.3 Some General Results and Attacks

Before introducing concrete examples of human identification protocols, it is worth looking at the general structure of such protocols. This leads to some generalized attacks and results that are applicable to all human identification protocols. A generic challenge-response type human identification protocol can be described as follows. Recall that C , S and R denote challenge, secret and response spaces, respectively. Let $\{0,1\}^n$ be the set of all binary strings of length n over $\{0,1\}$. C , S and R can be considered as sets of binary strings of length $\leq n$.

Protocol: Generic Challenge-Response Protocol.

Setup: \mathcal{C} and \mathcal{H} share a secret $s \in S$. The sets C , S , and R together with the relation ρ are publicly known.

- 1: \mathcal{C} sends a $c \in C$ to \mathcal{H} .
- 2: \mathcal{H} sends an $r \in R$ to \mathcal{C} such that $(c, s, r) \in \rho$.
- 3: \mathcal{C} to \mathcal{H} : if r is such that $(c, s, r) \in \rho$ then output **accept** else output **reject**.

This general structure is followed by most (if not all) human identification protocols. Note that there is no mention of randomness in the above description. This is done so to make the description as general as possible, thus including even such trivial challenge-response protocols as password-based authentication. In order to satisfy the definition of an identification protocol (cf. Definition 1), a straightforward conclusion is that the size of R should be greater than 1. Thus, $|R| \geq 2$. The relation ρ need not be a function. An example is the convex hull click based protocol [9, 21], analysed later in this thesis, which uses a relation rather than a function.

2.3.1 Random Guesses

In light of Definition 4, two general attacks can be carried out by \mathcal{A} on the above protocol, and hence on all human identification protocols. Recall that \mathcal{A} 's goal is to impersonate \mathcal{H} with or without knowing the secret. Both these attacks are a form of *random guess*, and can be carried out without observing any challenge-response pair. The first targets the response space, and the second targets the secret space.

Attack: Random Guess 1.

Input: A challenge $c \in C$.

Output: A response $r \in R$.

- 1: Given a $c \in C$, \mathcal{A} uniformly at random returns an $r \in R$.

The success probability of this attack is $|R|^{-1}$. Thus, in the generic challenge-response protocol, \mathcal{A} can always impersonate \mathcal{H} with probability $|R|^{-1}$ by simply choosing an r randomly from R . This holds true even if the adversary has not observed any challenge-response pair. There are two ways to mitigate this attack. Either

make $|R|$ large enough or iterate the challenge-response process a fixed μ number of times, so that the probability of a successful guess is small. If the responses are not uniformly distributed in R , then \mathcal{A} can improve his guess by choosing the most likely response. Thus, a key goal in designing a protocol is to ensure that the response is uniformly distributed. Otherwise, the number of iterations of the protocol have to be increased to compensate for the improved probability of guessing the response. The second random guess attack is as follows.

Attack: Random Guess 2.

Input: A challenge $c \in C$.

Output: A response $r \in R$.

- 1: Given a $c \in C$, \mathcal{A} randomly selects an $s \in S$ and returns an $r \in R$ such that $(c, s, r) \in \rho$.

The probability of success of this attack is *at least*:

$$\frac{1}{|S|} + \frac{|S| - 1}{|S||R|}$$

This attack can be mitigated by choosing a large enough S . Typically, the size of S is large enough so that the success probability of this type of attack is negligible. Indeed, it is easy to see that as $|S|$ grows, the success probability approaches the success probability of Random Guess 1. As a first line of defense, values of protocol parameters, in this case the sizes of S and R , should be chosen such that the probability of success of the random guesses is less than 10^{-6} .

As already mentioned, both these attacks do not require any observed challenge-response pairs. Once a single challenge-response pair is observed (belonging to a successful identification session), \mathcal{A} gains some knowledge about the secret, and can try to impersonate \mathcal{H} by finding the secret. However, below a certain number of observed challenge-response pairs, there are on average many candidates for the secret, and \mathcal{A} has no way to distinguish between them. This number is the information theoretic bound on m ; the number of challenge-response pairs required to find a unique solution, i.e., the secret.

2.3.2 Information Theoretic Bound on m

Intuitively, for a discrete random variable X , the Shannon entropy $H[X]$ is a measure of uncertainty in the knowledge of X . If $H[X] = 0$ then there is no uncertainty about X . Once one challenge-response pair is observed, some information about the secret is leaked. However, to reduce the uncertainty in the knowledge of the secret to 0, more challenge-response pairs are required. Let X_S be a discrete random variable representing the secret set S . Let $X_{C,R}$ be a discrete random variable representing

challenge-response pairs. Then the entropy of X_S is:

$$\begin{aligned}
H[X_S | X_{C,R} = (c, r)] &= - \sum_{s \in X_S} \Pr[X_S = s | X_{C,R} = (c, r)] \\
&\quad \times \log_2(\Pr[X_S = s | X_{C,R} = (c, r)]) \\
&\leq - \sum_{s \in X_S | (c, s, r) \in \gamma} \frac{|R|}{|S|} \log_2 \left(\frac{|R|}{|S|} \right) \\
&= \log_2 |S| - \log_2 |R|
\end{aligned}$$

where the equality holds if S is uniformly distributed over all challenges and responses. To be more accurate, the above expression should explicitly include the case when X_S equals the shared secret, in which case the conditional probability is 1. However, this is omitted for simplicity. The effect of this on the overall entropy is negligible. Given m distinct challenge-response pairs, and the random variables representing them $X_{C,R}^{(1)}, \dots, X_{C,R}^{(m)}$, this implies that:

$$H[X_S | X_{C,R}^{(1)} = (c_1, r_1), \dots, X_{C,R}^{(m)} = (c_m, r_m)] \leq \log_2 |S| - m \log_2 |R|$$

Equating the above expression to 0 and observing that $|S| \leq 2^n$, we get:

$$m \leq \frac{\log_2 |S|}{\log_2 |R|} = \frac{n}{\log_2 |R|}$$

Denote the above bound by m_{lb} . This means that a *computationally unbounded* adversary can always obtain a unique secret after observing more than m_{lb} challenge-response pairs. On the other hand, below m_{lb} it is information theoretically impossible to obtain a unique secret; there are more than one candidate. Since $|R| \geq 2$, we have the upper bound of $m_{\text{lb}} \leq n$. Therefore theoretically, in any challenge-response human identification protocol, the adversary \mathcal{A} can obtain the secret with probability 1, by observing less than n challenge-response pairs. If $|S| = 2^n$ and $|R| = 2$, there exists a protocol that can be used for n challenge-response pairs before the secret can be found uniquely. An example is Matsumoto's protocol from [19], a variant of which will be discussed shortly.

There are two major restraints in constructing information theoretically secure human identification protocols. First, if $|R|$ is too small, the protocol needs to be iterated a fixed number of times before it is secure against Random Guess 1. For instance, if $|R| = 2$, then the protocol needs to be iterated 20 times during a session to ensure that the probability of success of Random Guess 1 is approximately one in a million. This implies that the number of identification sessions is in fact $\frac{n}{20}$, instead of n . Secondly, if $|S| = 2^n$ it becomes prohibitive for humans to remember the secret, since if the number of sessions are to be increased, values of n should be in the range of 100 to 500, thus making the size of the secret very large. Therefore, any attempts at constructing a protocol with information theoretic security in mind can at best be used for a handful of authentication sessions. As shall be seen later, many protocols in literature are only secure in this sense.

2.3.3 Computational Security

Since m_{lb} cannot be too large, these protocols need to be designed with computational security in mind. Thus, to be secure for $> m_{\text{lb}}$ challenge-response pairs, all attacks on the protocol should be *computationally infeasible*. Given m challenge-response pairs, following is an example generic attack from literature, used frequently in this thesis.

Attack: Intersection Attack.

Input: m challenge response pairs $(c_1, r_1), \dots, (c_m, r_m)$.

Output: An $s \in S$.

- 1: Choose the pair (c_1, r_1) and find all $s \in S$, such that $(c_1, s, r_1) \in \rho$ (One can in fact choose any of the m pairs). Let S' be the set containing all such “candidates” for the secret.
- 2: **for** $i = 2$ **to** m **do**
- 3: **if** an $s \in S'$ is such that $(c_i, s, r_i) \notin \rho$ **then** discard the element from S' .
- 4: **if** S' contains only one element **then** output the element **else** sample an element randomly from S' as the output.

In general any element from the secret space S , that satisfies a given set of m challenge-response pairs shall be called a *candidate* for the secret. If $m > m_{\text{lb}}$, it is expected that the shared secret is the only remaining element in S' , and thus the Intersection Attack will output the correct secret. One way to carry out this attack is through brute force (exhaustive search). The time complexity of the brute force attack is $\mathcal{O}(|S|)$. Here lies the dilemma in constructing such protocols. Increasing $|S|$ increases the memory load on humans, while decreasing $|S|$ makes the brute force attack computationally feasible. However, as shall be seen next, there is a clever way to get around this, using the display of the terminal used by \mathcal{H} .

Before that, two important considerations are in order. First, it is important to know the distinction between an intersection attack and a brute force based intersection attack. While the two appear synonymous, it is possible to carry out the intersection attack on *part* of the secret instead of the whole secret. More specifically, since the secret is a bit string of length $\leq n$, a cleverly constructed intersection attack might find some bits of the secret instead of the whole. The computational time of this attack will be less than a brute force attack. An example of this will be given in the security analysis of the convex hull click based protocol from [9, 21]. Thus, an intersection attack does not always mean a brute force attack. The second important point is that there might be a computationally efficient algorithm, other than the intersection attack or brute force, to find the secret once some $m \geq m_{\text{lb}}$ number of challenge-response pairs are observed. That is, let f be some polynomial, then there might be an efficient algorithm that can find the secret uniquely with $m \geq f(n)$ challenge-response pairs. The goal is to expand the gap: $m_{\text{lb}} < m < f(n)$ as much as possible. An example protocol, discussed next, achieves $m = \mathcal{O}(n)$ number of observations, before the secret can be found through an efficient algorithm.

2.4 An Example Protocol

The example protocol mentioned here bears resemblance to Matsumoto's protocol from [19]. The shared secret is a binary vector of a fixed low Hamming weight. Low Hamming weight decreases the memory load on humans. This will be explained when discussing the implementation of the Example Protocol.

Protocol: Example Protocol.

Setup: Let μ , n and $k \leq n$ be publicly known positive integers. \mathcal{H} and \mathcal{C} share a secret $\mathbf{x} \in \mathbb{Z}_2^n$, such that $\text{wt}(\mathbf{x}) = k$.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} samples a random $\mathbf{c} \in \mathbb{Z}_2^n$ and sends to \mathcal{H} .
- 3: \mathcal{H} computes $r = \mathbf{c} \cdot \mathbf{x}^T \bmod 2$, and sends r to \mathcal{C} .
- 4: **if** all responses are correct \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

Note that the challenge space C is the space of all n -element binary vectors, the secret space S is the space of all n -element binary vectors of Hamming weight k , and the response space R is $\{0, 1\}$. The relation ρ in this protocol is a function, specifically the dot product modulo 2.

2.4.1 Security Analysis

In general, security analysis of human identification protocols in this thesis will follow this order. Random guess and brute force attacks shall be discussed first, followed by an estimation of the information theoretic bound on m , and concluded by a more thorough analysis considering the feasibility or infeasibility of other, more sophisticated attacks.

Random Guess and Brute Force Attacks

The success probability of Random Guess 1 is $2^{-\mu}$, and that of Random Guess 2 is at least $\frac{1}{2^n} + \frac{2^n-1}{2^n} \cdot \frac{1}{2}$. Choosing large enough μ and n can mitigate these attacks. Since the Hamming weight of \mathbf{x} is k , a brute force intersection attack mentioned before has complexity $\mathcal{O}\left(\binom{n}{k}\right)$. Choosing large enough n , and modest size of k can make this attack infeasible in practice. Once suitable values of the parameters involved are chosen, \mathcal{A} needs to find other types of attacks to impersonate \mathcal{H} .

Obtaining a Unique Secret

Assume that \mathcal{A} is given $m \geq 1$ challenge-response pairs. The information presented to the adversary consists of a series of binary challenge vectors and the corresponding response bits. \mathcal{A} has to find the secret vector \mathbf{x} . Given one challenge-response pair, there are on average:

$$\frac{1}{2} \binom{n}{k}$$

vectors of Hamming weight k that produce the same response. With m challenge-response pairs, this number reduces to:

$$\left(\frac{1}{2}\right)^m \binom{n}{k}$$

To reduce this to a single candidate gives the following estimate:

$$\begin{aligned} & \left(\frac{1}{2}\right)^m \binom{n}{k} < 1 \\ \Rightarrow & \binom{n}{k} < 2^m \\ \Rightarrow & \log_2 \binom{n}{k} < \log_2(2^m) \\ \Rightarrow & m > \log_2 \binom{n}{k} \end{aligned}$$

Denote the above lower bound by m_{lb} . Thus, if $m > m_{\text{lb}}$ a unique secret can be found, at least information theoretically. Note that here m_{lb} is far less than n ; it is logarithmic in n as a consequence of restricting the Hamming weight of the secret binary vector. An intersection attack like the one mentioned before can be used to find the secret with $m > m_{\text{lb}}$. However, if the intersection attack and/or brute force are infeasible, \mathcal{A} needs to find an efficient alternative to find the secret given $m > m_{\text{lb}}$ challenge-response pairs.

System of Linear Equations

The problem presented to the adversary can also be represented as a system of linear equations mod 2 as follows. Let C be an $m \times n$ matrix, whose i th row is the i th challenge vector, \mathbf{c}^T .¹ Let \mathbf{b} be an m -element binary vector, whose i th entry is the i th response bit from \mathcal{H} . Then \mathcal{A} can solve the following system of linear equations to find the secret:

$$C\mathbf{x} \equiv \mathbf{b} \pmod{2}$$

where $\text{wt}(\mathbf{x}) = k$. Given $m > n$ challenge-response pairs, it is possible to use Gaussian elimination to solve the above system of equations and efficiently find the secret. Recall that Gaussian elimination is a polynomial time algorithm. The protocol's security is conjectured on the assumption that there is no feasible algorithm (attack) to find a solution to the above system of linear equations with $m_{\text{lb}} < m < n$ observed challenge-response pairs. Note that below m_{lb} it is possible to find a solution to this system, as there are so many candidates for the secret. However, it is not possible to distinguish a solution from the secret. Between m_{lb} and n , however, a unique solution is guaranteed which is the secret itself. Unfortunately, as shall be seen next, due to implementation constraints n cannot be too large.

¹From now onwards, the symbol C will not denote the challenge space.

Of course, for Gaussian elimination to work, the adversary should have n linearly independent rows in the matrix C . Although C is a random matrix, with large enough n , it is highly likely that all m rows of C are linearly independent. Given a random $n \times n$ matrix, whose elements are from \mathbb{Z}_d , where d is a prime, the probability p_n that it is invertible (has n linearly independent rows) is given by [22].²

$$\lim_{n \rightarrow \infty} p_n = \prod_{i=1}^{\infty} \left(1 - \frac{1}{d^i}\right)$$

Thus, given a random matrix A with elements from \mathbb{Z}_2 , the probability that it has the full rank n , converges quickly to 0.2888 as n grows [22]. From a security perspective, therefore, it is safer to assume that all m rows of C are linearly independent. If not, the attacker can still choose m linearly independent rows from $\mathcal{O}(m)$ observations.

2.4.2 Implementation

A thorough discussion on the implementation of the Example Protocol will shed light on its design and feasibility. As described before, the protocol is similar to Matsumoto's protocols from [19]. The major difference being the restriction on the Hamming weight, which was suggested by Hopper and Blum, in a different protocol [8]. Matsumoto suggested the use of graphical objects in the implementation of his protocols in [19]. Following his work, typical implementation of human identification protocols is graphical. For instance, the Example Protocol can be implemented as follows. In the setup phase, a pool of n graphical icons are shown to \mathcal{H} . \mathcal{H} selects any k of these icons which constitute the shared secret between \mathcal{H} and \mathcal{C} . In the authentication phase, \mathcal{C} sends a challenge to \mathcal{H} which is a grid of n cells. Each cell contains a unique icon from the set of n icons and a random bit. \mathcal{H} looks at the cells containing his k icons and adds the corresponding bits modulo 2, and sends the result to \mathcal{C} . Figure 2.1 shows an implementation of the Example Protocol with specific (but small) values of parameters. Here $n = 10$ and $k = 4$. \mathcal{H} sums the bits corresponding to his set of secret icons and checks if the result modulo 2 is 0 or 1, and selects the appropriate option (radio button) at the prompt. The shared secret shown in the figure is, of course, not displayed with the challenge.

The position of the n icons does not change with each challenge. However, the bits in each cell are randomly selected by \mathcal{C} for each challenge. Note that \mathcal{H} does not have to memorize the whole n -element secret vector or even the locations of k 1's in his secret vector. \mathcal{H} only needs to recall the k secret icons present in the grid. Since the position of the icons in the grid does not change, after using the protocol many number of times, \mathcal{H} can easily locate his k icons in the grid. For a different implementation, icons can be replaced with pictures or even text. Since most human identification protocols involve challenges of n objects, this type of implementation is generic. In such an implementation, a large n is practical, since the only requirement is to have a large enough display to show all n icons (or a large subset of them).

²The result in [22] is in fact a general result for all finite fields.

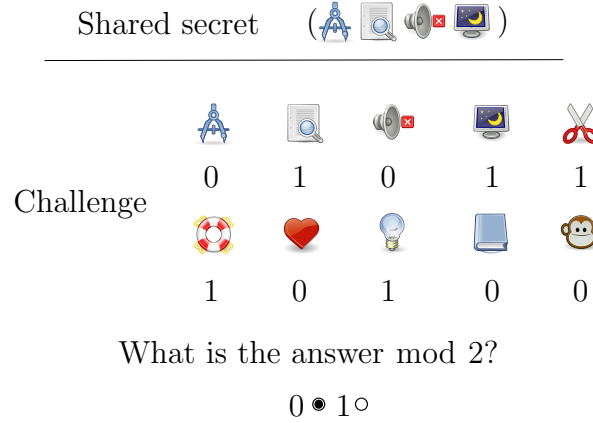


FIGURE 2.1: A challenge and response from the Example Protocol.

However, n cannot be arbitrarily large, since a reasonably sized display unit cannot accommodate a huge n . So, typically n can range between 100 to 300. If a subset of n icons is displayed, then it is possible to increase n . However, this introduces some other constraints, as we shall show during the security analysis of the convex hull click based protocol from [9, 21] in Chapter 4. Briefly, when only a subset of total icons are shown in each challenge, the probability that a secret icon appears in the subset should be the same as the probability of appearance of a non-secret icon. Otherwise, the adversary can do a “frequency analysis” to find the secret. Thus, even in this case, n cannot be made arbitrarily large, as that would mean an increase in the number of secret icons, and hence additional memory load on humans.

Text or Pictures?

As mentioned before, the protocol can also be implemented using a text-based implementation as opposed to graphical. For instance, in each cell an alphanumeric character can be shown instead of an icon. The user can then remember an alphanumeric password, and perform the steps by following the cells according to the alphanumerics in his password. However, a graphical implementation is preferred due to two main reasons. First, humans can recall a graphical object more quickly than text [1, 23]. In fact, it is much easier for humans to recall a picture due to the characteristics of human memory, as opposed to words including complete nouns, as some psychological studies have shown [24]. Secondly, in contrast to a textual implementation, a graphical implementation is less prone to dictionary attacks [24, 25]. Although there is some indication that a dictionary can be created from finding patterns in the selection of secret graphical objects by humans [24], the resulting dictionary attacks are less likely to be as severe as in the case of (textual) passwords. Thus, from both security and usability perspectives, a graphical implementation is preferable.

TABLE 2.1: Suggested parameter values for the Example Protocol.

n	k	μ	p_{rg1}	p_{rg2}	τ_{bf}	Sessions
200	16	20	10^{-6}	10^{-6}	2^{77}	10

Choosing Values of System Parameters

The values of system (protocol) parameters to be used in an implementation are chosen according to the success probability and time/space complexity of the attacks, and with usability in mind. Table 2.1 shows a list of suggested values of protocol parameters, and the time complexities and probabilities of success of known attacks. These values should be taken with a grain of salt, as the security analysis of the Example Protocol is by no means comprehensive. Nevertheless, the table is indicative of how the parameter values are selected for human identification protocols. Throughout this thesis, the symbols n , k and μ will be mostly reserved for the size of the challenge space, the “cap” on the secret space, and, the number of iterations of a protocol in one identification session, respectively.

In the table, μ is set to 20, which implies that the success probability of Random Guess 1, p_{rg1} is 2^{-20} or approximately 10^{-6} . The success probability of Random Guess 2, p_{rg2} is approximately 10^{-6} . This means that randomly guessing a secret is almost as good as randomly guessing a reply. Thus, the secret space is large enough such that this attack is not a threat. The time complexity of the brute force attack, τ_{bf} is 2^{77} with these values. Typically, we will consider a time complexity of 2^{70} to 2^{80} , infeasible. Finally, the “Sessions” column in the table indicates the number of sessions the Example Protocol can be used before the secret needs to be renewed. This is obtained as $\frac{n}{\mu} = \frac{200}{20} = 10$. The term $\frac{n}{\mu}$ is obtained by observing that, as discussed before, the secret can be easily found after the observation of $\mathcal{O}(n)$ challenge-response pairs through Gaussian elimination, and each session has μ challenge-response pairs. Thus, the protocol’s security is based on the conjecture that there is no efficient algorithm to find the secret with less than 10 sessions, or 200 challenge-response pairs. Note that $m_{\text{lb}} = \log_2 \binom{n}{k} \approx 77$. Thus, information theoretically it is possible to find the secret after about 77 challenge-response pairs. However, the conjecture is that there is no computationally feasible algorithm (attack) to do so between $77 < m < 200$.

Human Executability and Limitations

In light of Definition 4, it can now be conjectured that the Example Protocol is $(10^{-6}, 10)$ -secure against passive adversaries. However, as is perhaps evident, summing up 16 bits per challenge over 6 challenges will inevitably take considerably larger time than password-based authentication. Typically, the time taken to authenticate a user in human identification protocols ranges from 1 to 3 minutes [8, 21]. User studies have shown that it is desirable to bring this time down to less than 1 minute [1]. This

time bound, although still not close to the ideal time of password-based authentication, is acceptable due to increased security guarantees offered by these protocols. The two protocols proposed in this thesis, do not improve much on the time of 3 minutes. However, they provide a better balance between security and usability as compared to the existing protocols in literature. The two protocols from literature analyzed in this thesis, take lesser time for authentication. However, the resulting analysis shows that they are secure only for a handful of sessions. Furthermore, the second protocol, the convex hull click based protocol, takes time of more than 1 minute per authentication, even when implemented with values of protocol parameters that are too small for sufficient security.

It should be noted that substantial improvements on authentication times can be obtained through detailed usability studies and better implementation of these protocols. However, whatever the implementation, there exists a one-to-one correspondence between the theoretical and implementation-specific description of the protocol. In other words, it is always possible to describe the task of the adversary in terms of solving the underlying mathematical problem. In the Example Protocol, this task translates into solving a system of linear equations mod 2. Thus, it is important to first analyze the security of the protocol by carefully examining attacks on the underlying mathematical problem. Since the main focus of this thesis is on the analytical aspects of human identification protocols, more emphasis shall be given to the security analysis of the theoretical description of protocols. As such, usability and human-executability shall be given a cursory glance, to show the proof of concept.

2.5 Related Work

The threat model sketched in Figure 1.1 was first described by Matsumoto and Imai in [5]. They also proposed the first human identification protocol purported to be secure under this threat model [5]. A simplified version of their protocol can be described as follows. \mathcal{H} and \mathcal{C} share two secrets of length k . The first secret, known as the secret window, is a *set* of k characters from some alphabet, called the question alphabet. The question alphabet itself is composed of n characters. The second secret, called the secret word, is a k -character *string* from an answer alphabet, where the answer alphabet is a set of k characters. The challenge is a random string of n characters from the question alphabet. When presented with a challenge, \mathcal{H} locates his secret window, and one by one inserts characters from his secret word underneath the k locations corresponding to the characters in the secret window. The remaining locations are filled with random characters from the answer alphabet. Figure 2.2 shows a simple example of this protocol, which is the same example used in [5]. In the figure, Q stands for the question alphabet, W stands for the secret window, A stands for the answer alphabet, and S stands for the secret word. The bars on the secret window characters are only there for illustration, and are not present in an actual challenge.

In order to impersonate \mathcal{H} , the adversary \mathcal{A} has to find both the secret window and the secret word. According to Matsumoto and Imai's analysis [5], \mathcal{A} needs to observe

$$\text{Secret} \begin{cases} W = \{1, 2, 4, 6\} \\ A = \{1, 2, 3, 4\} \\ S = 3124 \end{cases}$$

Challenge	$\bar{2}$	8	5	$\bar{1}$	7	3	$\bar{6}$	$\bar{4}$
Response	3	4	3	1	2	1	2	4

$$Q = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

FIGURE 2.2: An iteration of Matsumoto and Imai's protocol.

at most $\binom{n}{k}$ challenge-response pairs to find the secret window. However, Wang et al. [12] showed that a much smaller number of observations are enough to find the secret window. They proposed two attacks on this protocol, only one of which is a passive attack. The attack is based on the observation that since the response contains all characters from the answer alphabet at least once, the number of challenge-response pairs required by the adversary is less than $\binom{n}{k}$. For instance, for the values used in Figure 2.2, the adversary requires 2^4 instead of $\binom{8}{4}$ pairs, since the response contains all characters from the answer alphabet exactly twice. Wang et al. proposed a fix to Matsumoto and Imai's protocol, but they did so to avoid the active attack. The number of observations required for the passive attack is the same as before [12]. Furthermore, the fix introduces another secret to be memorised by the user, and the resulting protocol is more complex than the original. Li and Teng [13] came up with another attack on Matsumoto and Imai's scheme. They observed that given a few challenge-response pairs, the adversary can find the secret word by finding subsequence strings common in all the challenge-response pairs; i.e., the adversary can perform a type of intersection attack. Matsumoto and Imai recommended the size of the question alphabet, the secret window and the answer alphabet to be 36, 10 and 4, respectively [5]. With these values, Li and Teng's attack can find the secret with only 2^7 challenge-response pairs. Li and Teng's also proposed protocols [13] which make the common subsequence string attack computationally infeasible, but the resulting protocols seem impractical as they require a large size of secret (three secrets of 20 to 40 bits).

The examples shown thus far are of protocols whose security is based on heuristic arguments. Matsumoto [19] attempted to change this by constructing protocols whose security can be claimed theoretically.

The $\mathcal{O}(n)$ Observed Challenge-Response Pairs Conundrum

The Example Protocol described before is similar to Matsumoto's linear algebra based protocols from [19], with a few differences. The first main difference is that Matsumoto didn't restrict the modulus to 2 in his protocols. Instead, his protocols are generalized over finite fields. The second major difference is that the Hamming weight of the secret in Matsumoto's protocol is not restricted. There are three protocols proposed in [19], but two of them can be considered variants of the first protocol [1]. The two variants

are, essentially, demonstrations of how the main protocol can be implemented in a user friendly way. The implementation of the Example Protocol and many other human identification protocols in literature follow the example of Matsumoto's implementation. The problem with Matsumoto's protocol is that the secret can be revealed after just $\mathcal{O}(n)$ observed challenge-response pairs through Gaussian elimination. Below this bound it is not possible to find a unique secret, as there are many possible candidates for the secret (recall the discussion on information theoretic security). Since the size of the secret is n , the protocol cannot be used for many authentication sessions as the secret size will increase dramatically with increasing n .

A possible fix is to restrict the Hamming weight of the secret to k , where k is much smaller than n , as is done in the Example Protocol. However, Gaussian elimination can still be applied after $\mathcal{O}(n)$ observed challenge-response pairs. Subsequently, researchers have attempted to construct protocols whose underlying problems cannot be represented as a system of linear equations that can be solved after $\mathcal{O}(n)$ observations. Hopper and Blum proposed two protocols in [8]. One of the protocols was based on the NP-Hard problem of learning parity in the presence of noise. This protocol is known as the HB Protocol in literature. The protocol essentially introduces noise in the response. Specifically, instead of sending the correct response bit in the Example Protocol, \mathcal{H} sends a wrong response with an error probability $\epsilon < \frac{1}{2}$. The result is that the adversary needs to observe at least a polynomial in n number of challenge-response pairs to find the secret. Hopper and Blum also proposed restricting the Hamming weight of the secret to k , and using the modulus 10 instead of 2; which humans are more familiar with. A usability study showed that humans can authenticate within 160 seconds [8], although the values of parameters used for the study were not large enough for sufficient security. There are some other drawbacks of the HB protocol as well, such as the requirement to send the wrong response with probability $\epsilon < 0.5$. It is arguably hard for humans to sample with a fixed probability of $\epsilon < 0.5$.

The other protocol from [8], namely the Sum of k Mins Protocol, involves the use of a secret which is a set of ordered pairs of locations. Again, the modulus suggested for the protocol's implementation is 10. Considering that the protocol is implemented in a similar way to the Example Protocol, the user has to compute the minimum of two random integers from the set $\{0, 1, \dots, 9\}$ corresponding to his ordered pairs of locations in the challenge, and sum the results of all k minimums mod 10. If the adversary wishes to convert this into a system of linear equations, he needs to "expand" the minimums, resulting in a system of $\binom{n}{2}$ unknowns. Gaussian elimination would require $\binom{n}{2}$ observed challenge-response pairs, which is quadratic in n instead of being linear in n as in the case of Matsumoto's protocols from [19]. The shared secret from this system of equations is a vector of $\binom{n}{2}$ unknowns with Hamming weight k . The minimum number of challenge response pairs required to find a unique secret in this protocol is:

$$m_{\text{lb}} = \log_2 \binom{n(n-1)/2}{k}$$

Thus, the security of the protocol is based on the conjecture that there is no efficient algorithm between the bounds $m_{\text{lb}} < m < \binom{n}{2}$. In Chapter 7, we will show that the

conjecture is most probably true, since the sum of k mins problem is hard in the sense of fixed parameter intractability [20]. However, the main drawback of the protocol is the requirement of memorizing a secret which is a set of k ordered pairs. Remembering pairs of secret locations can be hard for a human in case of a graphical implementation. On the other hand, if a textual implementation is used, then learning pairs of locations might not be hard, as the user can easily recall consecutive letters in a password. However, still the size of the secret is double than the secret in HB Protocol or the Example Protocol. Note that the Sum of k Mins Protocol discussed in this thesis is the variant which targets a passive adversary. As such we do not discuss the full protocol which is constructed to be secure against active adversaries [8]. Needless to say, the resulting protocol suffers from severe lack of usability.

Li and Shum's protocols [14] are also designed to be used for more than $\mathcal{O}(n)$ challenge-response pairs. Their design follows some principles, which include introducing uncertainty in the system of equations so that Gaussian elimination cannot be used after only $\mathcal{O}(n)$ observed challenge-response pairs. One way of introducing uncertainty is sending a noisy response with a fixed probability, as is done in the HB protocol from Hopper and Blum [8]. Li and Shum themselves introduce uncertainty through the use of a non-linear map [14]. The result is that the adversary cannot represent the problem as a system of linear equations with a non-negligible probability, and hence the protocol can be used for a large number of sessions. We show that one of the protocols from [14], namely Foxtail, is likely to be hard, as it is based on a fixed-parameter intractable problem.

There have also been a few attempts at constructing protocols that use the gap between human and artificial intelligence in certain computational tasks. An example is Jameel et al.'s image-based protocols from [15, 16]. The protocols borrow the concept from CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [26], which means that it is hard for an automated adversary to impersonate the user or learn the secret after observing challenge-response pairs. The main difference from CAPTCHAs is the claim that even a human cannot impersonate the legitimate user without the knowledge of the secret. However, these security claims are based on unproven assumptions, as it seems hard to exactly model the problem in mathematical terms. Thus, it is not clear how the security of these protocols can be assessed analytically. Furthermore, it appears difficult to automatically generate random challenges without human intervention from the server side.

Theoretical Work

Major theoretical advances in the area of human identification protocols have been few and far between. Matsumoto first discussed information theoretic security in his linear algebra based protocols described before [19]. This gives rise to the information theoretic bound on the number of challenge-response pairs required to find a unique secret. Hopper and Blum [8] showed a generalized attack on human identification protocols which use a secret of k objects from a pool of n objects, and in which the function computed by the human can be decomposed into intermediate values. The attack is a form of meet-in-the-middle attack which has been used in one form or the

other in solving many hard mathematical problems in cryptography. The generalized meet-in-the-middle attack on human identification protocols from Hopper and Blum is a form of time-memory tradeoff, and has time/space complexity $\mathcal{O}(\binom{n}{k/2})$. Thus, any protocol on which this attack can be applied should use parameter values such that the attack is infeasible. We will consider a time of 2^{70} to 2^{80} , and a space complexity of 2^{60} , infeasible.

To boost usability, in some protocols the user only uses a random subset of the secret, as opposed to the whole secret, to compute responses to challenges. This decreases the number of operations per iteration required by the user. More specifically, suppose such a protocol is implemented similar to the implementation of the Example Protocol. Then, using a subset of the secret means that only a subset of secret icons are shown in each challenge. The challenge itself is a subset of total icons in the pool. The protocols from [9, 10], are examples of such protocols. Coskun and Herley [27] showed a generalized brute force attack that reveals the secret in these protocols which utilize only part of the secret per response. They observed that if only a small subset of the secret is used to compute each response, then there are many “candidate secrets” that give the same response as the actual secret. For instance, let S be the secret space and let s be the secret. Assume $s' \in S$ is different from s in e bits. If e is small, say 1 bit, then the responses constructed from s and s' will be the same, unless the bit that is different is used in constructing the response. Thus, if a small number of bits are used to compute response, candidates for the secret that are similar to s (have a small value of e) can be easily distinguished from those that are dissimilar (have a large value of e). After finding one such candidate for the secret, the attack can “move” towards finding the actual secret by gradually finding the e different bits. This attack reduces the complexity of naive brute force, which is $\mathcal{O}(|S|)$, to:

$$\mathcal{O}\left(\frac{|S|}{\binom{\log_2 |S|}{e}}\right)$$

The improvement comes from the fact that the adversary does not need to search through the whole secret space, and instead need only retain a subset of the secret space. The tradeoff is that increasing e makes the success probability small, and decreasing it makes the space of secrets retained higher. Coskun and Herley did not show an example implementation of their attack on an actual human identification protocol in [27]. Instead, the attack is only analyzed theoretically. Nevertheless, a protocol should take into account the success probability of this attack, which requires that the number of bits of a secret needed to compute a response should not be too small.

Shoulder-Surfing Resistance

Some authentication systems have been designed to be secure against a restricted adversary; the shoulder-surfer. A shoulder-surfer can be a human or a hidden camera, and the term itself alludes to looking over a user’s shoulder to obtain information. Similarly, some other authentication systems target security against malware. The cognitive authentication scheme from Weinshall [10] is an example. However, we argue that all

these schemes can be considered human identification protocols. Recalling the definition of security (Definition 4), a passive adversary only observes the challenge-response pairs communicated between \mathcal{H} and \mathcal{C} . The exact manner in which the adversary eavesdrops is not specified. Shoulder-surfing or gathering information through malware are different manifestations of eavesdropping. Thus, from a theoretical viewpoint, these schemes can be considered as human identification protocols. Indeed, a protocol that is secure under Definition 4, by definition resists shoulder-surfing or malware.

However, our definition of human identification protocols does not include protocols that use auxiliary secure channels. For instance, a different research direction is to construct alternative input devices that use different human senses to hide the challenges to or the responses from the user, thus effectively using a secure auxiliary channel. These devices can potentially be more secure than traditional input devices, such as a keyboard or a mouse, since the adversary’s view is restricted. For example, Sasamoto et al.’s Undercover [18] uses a haptic device on which the user places his/her palm. The palm hides any external observation, while the user can receive part of the challenge from the haptic device through touch sensations on his palm. It is important to mention that there have been attacks reported on the Undercover scheme [28], which are based on observing the user’s behavior. In general, any human identification protocol is susceptible to these “side-channel” attacks, and it is important to train users not to leak any extra information through their actions during an authentication session. For instance, by avoiding pointing on the screen at one of the secret icons.

Returning to shoulder-surfing resistant authentication schemes, most of these are built primarily with usability in mind, and as a result, many have been shown to be insecure. Weinshall’s scheme [10] was cryptanalysed in [29], and was shown to be susceptible to Satisfiability (SAT) solvers. Golle and Wagner [29] represented the problem presented to the adversary in Weinshall’s scheme as a set of SAT clauses, and showed that SAT solvers, programs to solve instances of SAT, can find the secret in a matter of seconds. The implication is that the parameter values in Weinshall’s protocol required to make SAT based attacks infeasible, are too large to be practical for humans. Another example of an insecure system is the virtual password system from [11] which was broken by Li et al. in [30]. The schemes from [31] and [32] focus on human shoulder-surfers, and are not secure against an automated adversary. The security against an automated shoulder-surfer (a hidden camera) is only up to the aforementioned information theoretic bound on the number of observations. This bound is much less for these schemes, typically less than 10. Beyond this bound the adversary can easily find the secret, since the parameter values are small in these schemes (for usability) making the brute force attack feasible.

However, some schemes purported to be secure against shoulder-surfers or malware, are potentially secure for a larger number of authentications. Examples are the convex hull click based identification protocols from [9, 21], and a similar protocol based on a geometric problem from Zhao and Li [33] which can not only provide good security against shoulder-surfers, but appear to be based on a hard to learn mathematical problem. In the convex hull based protocol, the user locates at least three of his secret icons in a challenge, mentally forms the convex hull of the located secret icons, and then randomly clicks anywhere within the convex hull. We will show in Chapter 4

that this protocol is susceptible to an attack which finds information about the secret by utilizing the structure of the convex hulls of three points (triangles). Nevertheless, the protocol can still be used for a much larger number of authentication sessions, then say, protocols that are designed to be secure only for a handful of authentication sessions. An example is Bai et al.'s Predicate-based Authentication Service (PAS), which is claimed to be secure for 10 sessions. However, as we show next, PAS is only secure for a much smaller number of authentication sessions. Thus, without a careful mathematical analysis, there is no guarantee that a protocol is secure, even if the security is claimed to hold for a very small number of authentication sessions.

3

Security Analysis of PAS (Predicate-based Authentication Service)

Recently a new human identification protocol called PAS (predicate-based authentication service) was proposed by Bai et al. in [17], purported to be secure under Matsuoto and Imai's threat model for a limited number of authentication sessions. In this chapter we give a detailed security analysis of PAS and show that PAS is insecure against both brute force attack and a probabilistic attack. In particular we show that the security of PAS against brute force attack was highly overestimated by Bai et al. Furthermore, we introduce a probabilistic attack, which breaks part of the password (the shared secret) even with a very small number of observed authentication sessions. Although the proposed attack cannot completely break the password, it can downgrade the PAS system to a much weaker system similar to common OTP (one-time password) systems.

This chapter is organized as follows. We briefly describe how the PAS scheme works in Section 3.1. A re-evaluation of security and usability of the PAS scheme is given in Section 3.2, and a probabilistic attack on PAS is described in Section 3.3. The last section contains concluding remarks.

3.1 Predicate-based Authentication Service

In this chapter, we try to keep the original notations used in [17], due to the bulk of notation used in that work. As a result, the meaning of most symbols used in this chapter are quite different from their usage in the rest of this thesis. Using similar notation allows for easier comparison of the two works. We also keep some of the terminology used by Bai et al. In particular, Bai et al. used the word *password* for the shared secret between the human \mathcal{H} and the server \mathcal{C} . The password is then

further subdivided into entities which Bai et al. refer to as secrets. Thus, this different terminology is also adopted in this chapter so that the results from Bai et al. [17] and this chapter can be easily cross-referenced. However, we do change some symbols so that the description is not too different from the rest of the thesis.

The PAS Scheme

In the PAS scheme, the prover \mathcal{H} (the human user) and the verifier \mathcal{C} (the PAS server) share a *password* S composed of p *secrets* S_1, \dots, S_p . Each secret S_i consists of a 2-D *secret cell index* (u_i, v_i) and a *secret word* of size len $W_i = w_i[1] \cdots w_i[len]$. The 2-D index denotes a cell at position (u_i, v_i) in an $m \times n$ 2-D grid, so $1 \leq u_i \leq m$ and $1 \leq v_i \leq n$. Each character of the secret word belongs to an alphabet \mathbb{H} of size H . Since the 2-D cell index can simply be transformed to a 1-D index $c_i = (u_i - 1) \cdot n + v_i \in \{1, \dots, M = mn\}$, in this paper we will analyze the PAS system by replacing (u_i, v_i) with the equivalent 1-D cell index $c_i \in \{1, \dots, M\}$. That is, each secret will be represented as $S_i = (c_i, W_i) = (c_i, w_i[1] \cdots w_i[len])$. A password with parameter $p = 2$, $len = 7$, $M = 25$ looks like “(12,catchme; 25,beathim)”.

Broadly speaking, the PAS scheme is a challenge-response human identification protocol, in which the verifier \mathcal{C} raises a number of challenges and the prover \mathcal{H} must give correct responses to all challenges in order to pass the authentication process. To achieve security against passive adversaries, Bai et al. suggested using p “*predicates*” (instead of the password S) to make responses to challenges. The p predicates are dynamically calculated by \mathcal{H} from the secret S and a *predicate index* I , which is sent from \mathcal{C} to \mathcal{H} at the beginning of each authentication session. \mathcal{H} calculates $\hat{I} = (I \bmod len) + 1$ and generates the p predicates as follows: $\forall i = 1, \dots, p$, $\mathbf{pred}_i = (c_i, h_i)$, where $h_i = w_i[\hat{I}]$. We shall say $\mathbf{pred} = (\mathbf{pred}_i)_{i=1}^p$ is a *p-predicate vector*. When $p = 2$, the p -predicate vector is also called a *predicate pair*. The predicate pair derived from the password “(12,catchme; 25,beathim)” and $I = 2$ will be “(12,a; 25,e)”.

Each challenge raised by \mathcal{C} includes l challenge tables, each of which contains M cells filled with a certain number of distinct characters in \mathbb{H} . To ensure that each character occurs in each cell with probability 0.5, the number of characters in each cell is always $\frac{H}{2}$ when H is even, and is $\frac{H-1}{2}$ or $\frac{H+1}{2}$ with probability $\beta = 0.5$ when H is odd. To simplify our analysis, we assume H is even and so each cell always contains $\frac{H}{2}$ characters. Note that in the default setting of the PAS scheme $H = 26$. In addition to the l challenge tables, \mathcal{C} also sends a p -dimensional response table to \mathcal{H} . Each dimension of the response table has 2^l possible values, so there are 2^{pl} cells in the response table. All the cells are filled with 2^l distinct character strings, each of which occurs exactly in $2^{(p-1)l}$ cells.¹ See Figures 1 and 2 in [17] for examples of the challenge and the response table.

\mathcal{H} constructs a response to each challenge based on the response table and p *hidden responses* generated from the p predicates. For the i -th predicate $\mathbf{pred}_i = (c_i, h_i)$, the corresponding hidden response is an l -bit integer $B_i = b_i[1] \cdots b_i[l]$, where $b_i[j] = 1$ if h_i occurs in the c_i -th cell of the j -th challenge table and $b_i[j] = 0$ otherwise. With

¹Note that for $p = 2$, this rule actually implies a $2^l \times 2^l$ Latin square filled with 2^l distinct elements.

the p hidden responses, \mathcal{H} finds the cell at the position (B_1, \dots, B_p) in the response table, and sends the character string in that cell as the response to the challenge. A step-by-step description of the authentication process of the PAS scheme is as follows.

Protocol: PAS.

Setup: Let \mathbb{H} , p , M , l , len and n_r be public parameters. \mathcal{C} and \mathcal{H} share a password $S = (S_1, S_2, \dots, S_p)$.

- 1: \mathcal{C} sends a predicate index I to \mathcal{H} .
- 2: \mathcal{H} calculates the p -predicate vector $(\text{pred}_1, \dots, \text{pred}_p)$ from the password $S = (S_1, \dots, S_p)$ and the predicate index.
- 3: **for** $i = 1$ to n_r **do**
- 4: \mathcal{C} sends a challenge with l challenge tables and a response table with 2^{pl} cells.
- 5: \mathcal{H} calculates p hidden responses B_1, \dots, B_p and finds the cell at position (B_1, \dots, B_p) in the response table.
- 6: \mathcal{H} sends the character string in the cell (B_1, \dots, B_d) to \mathcal{C} .
- 7: \mathcal{C} outputs **accept/reject** by checking if all the responses are correct.

For reference, Figure 3.1 shows a very simplified example with a single challenge table, i.e., $l = 1$, which contains $M = 2$ cells. The alphabet H is composed of all uppercase English letters. The two cells in the challenge table contain 13 letters each. Suppose \mathcal{H} has two secrets in his password ($p = 2$): (1, hate; 2, love). The response table has $2^{pl} = 2^{2 \cdot 1} = 4$ cells, and $2^l = 2^1 = 2$ characters (bits), which occur $2^{(p-1)l} = 2^{1 \cdot 1} = 2$ times each. Suppose now that the verifier \mathcal{C} sends the index $I = 3$ to \mathcal{H} . This is to indicate to \mathcal{H} that he has to reply to the challenge by using the third letter in the two secret words. \mathcal{H} looks at cell 1, since the secret cell index of the first secret is 1, and checks if the letter ‘t’ is present or not. Since it is absent, he mentally remembers the answer ‘n’. \mathcal{H} then looks at cell 2, since the secret cell index of the second secret is 2, and checks if the letter ‘v’ is there. Since it is present he remembers the answer ‘y’. Finally \mathcal{H} replies by sending the bit corresponding to ‘ny’, where ‘n’ is the row label and ‘y’ is the column label. The bit turns out to be 1 in this case. Bai et al. in fact use CAPTCHAs in the response table entries to prevent automated gathering of responses. However, this assumption is not necessary in the threat model considered in this thesis, since a human can easily solve the CAPTCHAs after gathering challenge-response pairs. Note that in the Matsumoto and Imai’s threat model, the adversary can be automated or human, or both.

Generation of the Predicate Index I

In [17], Bai et al. did not clearly mention how the predicate index I should be generated. Instead, they discussed the number of authentication sessions (denoted by t) each predicate index \hat{I} can be used. The maximal number t_{max} turns out to be 1 for the default setting of the PAS scheme. This means that each possible value of \hat{I} is used for

<div style="background: linear-gradient(to top right, #f08080, #f0f0f0); padding: 5px; display: inline-block;"> A B E Z V U X D R S W G H </div>	<div style="background: linear-gradient(to top right, #f08080, #f0f0f0); padding: 5px; display: inline-block;"> C V Y Z T U Q R D S I O N </div>	y	n
1	2	0	1
		1	0

FIGURE 3.1: A challenge and response table from PAS.

one authentication session only, and the password has to be renewed after all the len possible values are exhausted. The predicate indices of the len authentication sessions may simply be chosen as $1, \dots, len$ or a permutation of the len values. In this chapter, we assume the PAS scheme runs in a “random permutation mode”, in which a random permutation of $1, \dots, len$ determines the predicate index used for each authentication session. Note that this is the most complicated (and thus the most “secure”) way one can adopt to assign the len values of the predicate index to all the authentication sessions.

Extended PAS

Bai et al. also extended the above basic PAS scheme to allow $k > 1$ cell indices in each secret S_i . In this case, the i th secret in the password is redefined as $S_i = (c_{i,1}, \dots, c_{i,k}, W_i)$. Accordingly, k predicate indices I_1, \dots, I_k will be sent from \mathcal{C} to \mathcal{H} for each authentication session. \mathcal{H} calculates the i th predicate pred_i as a set of k sub-predicates $\{\text{pred}_{i,j}\}_{j=1}^k$, where:

$$\begin{aligned}
\text{pred}_{i,j} &= (c_{i,\hat{I}_{j,k}}, h_{i,j}), \\
h_{i,j} &= w_i[I_{j,len}], \\
\hat{I}_{j,k} &= (I_j \bmod k) + 1, \\
\text{and } \hat{I}_{j,len} &= (I_j \bmod len) + 1.
\end{aligned}$$

With this extended predicate containing k sub-predicates, the hidden response B_i of the i th predicate is obtained as follows: \mathcal{H} first calculates k hidden sub-responses $B_{i,1}, \dots, B_{i,k}$ for the k sub-predicates in the same way as in the basic PAS scheme, and then determines B_i as the bitwise OR of the k hidden sub-responses: $B_i = B_{i,1} \vee \dots \vee B_{i,k}$. To ensure uniform distribution of B_i over $\{0, \dots, 2^l - 1\}$, the number of distinct characters in each cell of each challenge table and the corresponding probability β should be determined by Eqs. (6) and (8) in [17], respectively.

List of Parameters

A list of the parameters (with the default values) and notations involved in the description of the PAS scheme is given in Table 3.1. The column labeled “Notation (extended)” shows the notations from the extended PAS scheme.

TABLE 3.1: List of parameters/notations used in the description of PAS.

Parameter	Description	Default
p	The number of secrets in the password	2
len	The number of characters in a secret word	10
\mathbb{H}	The set of all possible characters in a secret word	$\{A, \dots, Z\}$
H	The size of \mathbb{H} , i.e., the number of all possible characters	26
l	The number of challenge tables in a challenge	2
$M = mn$	The number of cells in a challenge table	25
n_r	The number of challenges (rounds) in an authentication session	5
k	The number of cell indices in each secret S_i	1
	The number of sub-predicates in each predicate \mathbf{pred}_i	
Notation	Description	
$S = (S_1, \dots, S_p)$	The password shared between \mathcal{H} and \mathcal{C}	
$S_i = (c_i, W_i)$	The i -th secret in the password S	
$c_i \in \{1, \dots, M\}$	The secret cell index in the i -th secret S_i	
$W_i = w_i[1] \dots w_i[len]$	The secret word in the i -th secret S_i , where $w_i[1], \dots, w_i[len] \in \mathbb{H}$	
$I \in \mathbb{Z}^+$	The predicate index sent from \mathcal{C} to \mathcal{H}	
$\hat{I} = (I \bmod len) + 1$	The predicate index modulo len	
$\mathbf{pred} = (\mathbf{pred}_i)_{i=1}^p$	The p -predicate vector used by \mathcal{H} in an authentication session	
$\mathbf{pred}_i = (c_i, h_i)$	The i -th predicate, where $h_i = w_i[\hat{I}]$	
$B_i = b_i[1] \dots b_i[l]$	The hidden response corresponding to the i -th predicate \mathbf{pred}_i	
$b_i[j] = 1$ (or 0)	h_i occurs (or does not occur) in the c_i -th cell of the j -th challenge table	
t	The number of authentication sessions a predicate index can be used	
Notation (extended)	Description	
$S_i = (c_{i,1}, \dots, c_{i,k}, W_i)$	The i -th secret in the password S	
$c_{i,k} \in \{1, \dots, M\}$	The k -th secret cell index in the i -th secret S_i	
$I_1, \dots, I_j \in \mathbb{Z}^+$	The predicate indices sent from \mathcal{C} to \mathcal{H}	
$\hat{I}_{j,len}$ and $\hat{I}_{j,k}$	The j -th predicate index modulo k and len , respectively	
$\mathbf{pred}_i = \{\mathbf{pred}_{i,j}\}_{j=1}^k$	The i -th predicate, where $h_i = w_i[\hat{I}]$	
$\mathbf{pred}_{i,j} = (c_{i,\hat{I}_{j,k}}, h_{i,j})$	The j -th sub-predicate in the i -th predicate, where $h_{i,j} = w_i[\hat{I}_{j,len}]$	
$B_i = B_{i,1} \vee \dots \vee B_{i,k}$	The hidden response corresponding to the i -th predicate \mathbf{pred}_i	
$B_{i,j}$	The hidden sub-response corresponding to the sub-predicate $\mathbf{pred}_{i,j}$	

Security and Usability Analysis by Bai et al.

In [17], the security of the PAS scheme was analyzed against three different possible attacks: brute force attack, random guess attack and SAT (satisfiability) solver attack. Three different attack targets were checked: password, predicate, and response. By assuming each predicate index is used for t authentication sessions, the security was measured in terms of the cardinality of the attack set, i.e., the size of the reduced target space, or the number of candidate targets passing all the observed authentication sessions. Table 3.2 shows the results reported by Bai et al. in [17]. In the table:

$$N = \frac{pk(MH)^{pk}}{2^{ln_r t} (k!)^p}$$

By setting a minimal security level for each possible attack, Bai et al. also described how to get t_{max} , the maximal number of authentication sessions a predicate index \hat{I}

TABLE 3.2: The security of PAS, estimated by Bai et al.

	Password	Predicate	Response
Brute Force	$M^{pk} H^{p \cdot len}$	NA	NA
Random Guess	$M^{pk} H^{p \cdot len}$	$(MH)^{pk} / (k!)^p$	2^{ln_r}
SAT	$\left(M \left(1 - \left(1 - \frac{1}{M}\right)^N\right)^{len/k}\right)^{pk} H^{p \cdot len}$	$\left(M \left(1 - \left(1 - \frac{1}{M}\right)^N\right)^{len/k} H\right)^{pk} / (k!)^p$	NA

can be repeatedly used. For the default setting of the basic PAS scheme, it was claimed that $t_{max} \approx 1$ so that the same password S can be used for *at least* $t_{max} \cdot len = 10$ times before renewal.

Bai et al. also did a usability study based on a prototype system with the default parameters and $n_r = 2, 3, 4, 5$. The average time consumed on deriving the predicates from secrets was around 35 seconds, and the time taken for each round (iteration) ranged from 8.37 to 10.5 seconds. When $n_r = 5$, the total login time for one authentication session was around 84 seconds on average. A survey on the upper bound of the login time was also conducted, and more than half of the participants chose 2 minutes. We will use these statistics to discuss the relationship between security and usability of the PAS scheme.

3.2 Re-Evaluating Security and Usability

First of all, we point out that the definitions of two of the three attacks in [17] are problematic. Observing Table 3.2, one can see there are two “NA”-s for brute force attack, and security against brute force attack is the same as security against random guess attack. In fact, according to the definitions given in [17], the brute force attack and the random guess attack are actually the same attack if the target is the password.

In our opinion, the brute force attack should be defined as exhaustively searching the whole password/predicate space \mathbb{S} to determine a subspace (i.e., the so-called “attack set” according to the terminology used by Bai et al. in [17]) $\mathbb{S}^* \subseteq \mathbb{S}$, which is composed of all candidates of the password/predicate that pass all the authentication sessions observed by a passive adversary. Apparently, the correct password/predicate used by the human prover \mathcal{H} is always in the subspace \mathbb{S}^* . When $|\mathbb{S}^*| = 1$ or small enough, we say the brute force attack is successful. Just as its name implies, the random guess attack should be defined as randomly guessing the correct password, predicate or response of each challenge in order to pass the authentication session. Note that in the brute force attack the goal is to (maybe partially) reveal the password, but in the random guess attack the goal is to simply impersonate a claimed identity without trying to break any target. Recall the definitions of random guess attacks from Chapter 2.

In [17] Bai et al. claimed that brute force attack does not take the predicates as the target, because they vary from session to session. We have a different opinion. Since

TABLE 3.3: Re-evaluated security of PAS against three attacks.

	Password	Predicate	Response
Brute Force / SAT	$\left(\frac{1 + \left(\frac{(MH+k-1)^p - 1}{2^{ln_r t}} \right)}{2^{ln_r t}} \right) \frac{len!}{(len-k)!}$	$1 + \frac{(MH+k-1)^p - 1}{2^{ln_r t}}$	NA
Random Guess	$\frac{1}{1/2^{ln_r} + (2^{ln_r} - 1) / \left(2^{ln_r} \binom{MH+k-1}{k}^p \right)}$	$< 2^{ln_r}$	2^{ln_r}

the cell indices remain the same for all predicates, breaking the cell indices (as part of each predicate) may help an attacker pass a later authentication attempt with higher probability before password renewal. As a consequence, it is important to consider brute force attack targeting predicates. In fact, the first step of the probabilistic attack described in the next section of this chapter is based on the brute force attack on predicates.

In the following, we re-evaluate the security of PAS, and point out that the security of the PAS scheme was over-estimated in [17]. Our new estimate is shown in Table 3.3. We also point out that the extended PAS scheme is not practical, which allows us to focus only on the basic PAS scheme in the next section.

3.2.1 Security against Brute Force Attack Targeting Predicates

To facilitate the following discussion, denote the number of distinct p -predicate vectors by $N(p, k)$. In [17], the value of $N(p, k)$ was estimated to be $(MH)^{pk}/(k!)^p$. Unfortunately, this estimate is wrong. This can be easily verified when $k > 1$ and $\gcd(MH, k) = 1$. In this case, $(MH)^{pk}/(k!)^p$ is not an integer. To derive the correct value of $N(p, k)$, note the following fact: the number of distinct sub-predicates in the i th predicate ranges from 1 to k . Thus, we immediately have:

$$\begin{aligned}
 N(p, k) &= \left(\binom{MH}{1} + \binom{MH}{2} + \dots + \binom{MH}{k} \right)^p = \binom{MH+k-1}{k}^p \\
 &= \left(\frac{(MH+k-1) \cdots (MH)}{k!} \right)^p \geq \frac{(MH)^{pk}}{(k!)^p}
 \end{aligned}$$

Although Bai et al. did not over-estimate the value of $N(p, k)$, they neglected the influence of n_r and t on the size of the attack set. However, when the adversary tries to use a randomly selected incorrect p -predicate vector to calculate the response to each challenge, the probability of getting the correct response is only $\frac{1}{2^t}$ (under the assumption that the calculated response uniformly distributes over all the 2^l possible responses). Assuming that the responses of different challenges are independent of each other (which is so if all the challenges are generated independently by the verifier \mathcal{C}),

the probability that a randomly selected predicate will pass t observed authentication sessions will be $1/2^{\ln_r t}$. Since there are one correct p -predicate vector and $\binom{MH+k-1}{k}^p - 1$ incorrect ones, with t observed authentication sessions the average size of the attack set will be:

$$1 + \frac{\left(\binom{MH+k-1}{k}\right)^p - 1}{2^{\ln_r t}}$$

which is much smaller than the original estimate in [17]. Note that the computational complexity of the brute force attack is still $\mathcal{O}\left(\left(\binom{MH+k-1}{k}\right)^p\right)$, since all the possible predicates have to be checked one by one.

3.2.2 Security against Brute Force Attack Targeting Password

When the target of brute force attack is the password S , Bai et al. estimated the password space as $M^{pk} H^{p \cdot len}$, which is the number of all possible p -dimension vectors (S_1, \dots, S_p) . However, due to the special design of the PAS scheme, a password S can be equivalently represented as $\frac{len!}{(len-k)!}$ distinct p -predicate vectors: $\text{pred} = (\text{pred}_i)_{i=1}^p$, where $\frac{len!}{(len-k)!}$ is the number of all possible values of the k -tuple predicate-index vector $(\hat{I}_{1,len}, \dots, \hat{I}_{k,len})$ and:

$$\text{pred}_i = \left(c_{i,\hat{I}_{1,k}}, \dots, c_{i,\hat{I}_{k,k}}, w_i[\hat{I}_{1,len}] \cdots w_i[\hat{I}_{k,len}]\right).$$

Note that any change in one predicate will not influence any other predicates, so they are independent of each other. As a result, the password space can be calculated as the union of all the predicate spaces. Then, we can estimate the size of the modified password space to be:

$$\left(\binom{MH+k-1}{k}\right)^p \frac{len!}{(len-k)!}$$

which may be much smaller than the size of the original password space in case $len > k$ and $H > len$. For the default parameters, Table 3.4 shows how the ratio:

$$r = \log_{10} \left(\frac{M^{pk} H^{p \cdot len}}{\left(\binom{MH+k-1}{k}\right)^p \frac{len!}{(len-k)!}} \right)$$

changes as k increases from 1 to $len = 10$. We can see r is always much larger than 1, which means the size of the re-represented password space is always much smaller than $M^{pk} H^{p \cdot len}$. This can be best demonstrated for the basic PAS scheme. In this case, each password can be represented as len independent predicates, and the password space is reduced to $(MH)^p \cdot len$, which is smaller than $M^p H^{len \cdot p}$ as long as $H^{len \cdot (p-1)} > len$. For the default setting of the basic scheme, we can calculate that the password space is only $(MH)^p \cdot len = (25 \times 26)^2 \cdot 10 \approx 2^{22}$, which is too small from a cryptographic point of view. Since the cell index for each predicate is always the same, we can separately store the p cell indices c_1, \dots, c_p and the len p -character words $\{W_j^* = w_1[j] \cdots w_p[j]\}_{j=1}^{len}$.

TABLE 3.4: The ratio of sizes of re-represented versus original password space.

k	1	2	3	4	5	6	7	8	9	10
r	24.470	21.286	18.505	16.030	13.814	11.835	10.085	8.5749	7.3418	6.499

Apparently, this is just a reorganization of different parts of the password, so no extra memory is needed.

After representing the password space as the union of $\frac{len!}{(len-k)!}$ predicate spaces, we can easily obtain the size of the attack set with t observed authentication sessions for each predicate based on the result we obtained in the last subsection. That is:

$$\left(1 + \frac{\binom{MH+k-1}{k}^p - 1}{2^{ln_r t}}\right) \frac{len!}{(len-k)!}$$

In addition, it deserves mention that a dictionary attack can narrow down the password space even further, since human users have to choose the p secret words as something that can be easily recalled. This is an inherent drawback of any human authentication scheme based on textual characters. Changing \mathbb{H} to a set of graphical objects (such as a set of some small icons) may help mitigate this problem to some extent.

3.2.3 Security against Random Guess Attack

As discussed in Chapter 2, in a random guess attack the adversary does not need to try *all* passwords/predicates/responses, but randomly pick one from the password, predicate or response space and see if he can pass the authentication session. For random guess attack, there is no attack set, but we can use the reciprocal of the success probability of passing the authentication session as an equivalent metric of security.

When the adversary chooses a random response, the original estimate in [17] is correct, since there are 2^l possible responses. But the attacker can get a higher success rate if he chooses a random predicate or a random password. It is because the attacker has a chance to guess the correct predicate, which will definitely lead to the correct response. For all the other incorrect predicates, the success rate is the same as that of randomly guessing the response. Thus, the overall success rate is:

$$1 \cdot \frac{1}{\binom{MH+k-1}{k}^p} + \frac{1}{2^{ln_r}} \cdot \frac{\binom{MH+k-1}{k}^p - 1}{\binom{MH+k-1}{k}^p} = \frac{1}{2^{ln_r}} + \frac{2^{ln_r} - 1}{2^{ln_r} \binom{MH+k-1}{k}^p} > \frac{1}{2^{ln_r}}. \quad (3.1)$$

3.2.4 Security against SAT Attack

The SAT attack can be considered as a special form of the brute force attack. Observing our result obtained for the brute force attack and the one Bai et al. got for the SAT attack (when the attack target is the password), one can easily see the former is much

smaller than the latter in most cases. For instance, for the basic PAS scheme with the default parameters and $t = 1$, the latter is as high as $2^{103.3}$, but the former is only about $2^{22} \ll 2^{103.3}$. This implies that the security analysis on the SAT attack given in [17] was also highly over-estimated.

Because of the space limit, in [17] Bai et al. did not publish details of the security results on the SAT attack shown in Table 3.2. Fortunately, an appendix was available upon request from the first author of [17], which includes the derivation process. After checking the derivation process, we noticed that Bai et al. actually had not considered any specific features of the SAT attack. The attack was not transformed to a typical SAT problem formatted in Conjunctive Normal Form (CNF), either. As a matter of fact, since the SAT problem is NP-complete and there are many specific SAT solving algorithms, an analytic estimation on the number of solutions (i.e., the size of the attack set) and the time complexity of a specific practical SAT problem like the one from the PAS scheme is often very difficult [34].

Bai et al. actually derived the equations shown in Table 3.2 as follows: 1) assume the SAT solver is capable of making full use of the information leakage as we did in Section 3.2.1; 2) estimate the number of predicates that pass all the t authentication sessions; 3) calculate the probability that each cell index appears in all the candidate predicates, and then replace M by the number of candidate cell indices $M^* = M(1 - (1 - 1/M)^N)^{len/k}$. Unfortunately, this derivation process does not reflect the real security level against the SAT attack. As a matter of fact, the process is not only incorrect for the SAT attack but also for the brute force attack. If a SAT solver can eliminate an incorrect predicate, then the brute force attack can do so, too. It is well known that SAT solvers work like an optimized brute force algorithm for searching the whole solution space. The main difference from a naive brute force searching algorithm and a SAT algorithm is the time complexity of finding one or more solutions. Since Bai et al. chose the size of the attack set (i.e., the number of solutions) as the security metric, the SAT attack should have the same “performance” as the naive brute force attack. This means that the one we obtained for the brute force attack is a more reasonable upper bound of the SAT attack.

3.2.5 Usability

Although Bai et al. claimed that the usability of the (basic) PAS scheme is much better than some other solutions (see the last sentence of Section 5.1 of [17]), we doubt if it is a fair comparison. The main problem is the lack of a consistent security analysis of the solutions. The existence of multiple security factors also makes it difficult to find a reasonable parameter set of each solution to compare the usability. For instance, the Cognitive Authentication Scheme (CAS) proposed in [10] has a low-complexity variant, which has relatively good usability but a lower security level according to the cryptanalysis reported in [29]. Comparing the CAS solution with the default setting of the basic PAS scheme, we have the following results:

- average login time: CAS – 1.5 minutes = 90 seconds, PAS – 84.23 seconds;
- security against random guess attack: CAS – $2^{20} \sim 2^{25}$, PAS – 2^{10} ;

- maximal number of authentication sessions a password can be used: CAS – less than 12, PAS – around 10 (actually less according to our analysis given in the next section).

Based on the above data, it is obvious that the basic PAS scheme is worse than the low-complexity variant of CAS in terms of both security and usability. Actually, even the above comparison is not a fair one, either, since we do not consider all security and usability factors. In our opinion, comparing performance of different human identification protocols is not an easy task without a comprehensive security and usability study of all the systems involved. But one principle is undoubtedly clear: the comparison of usability should be made for the same level of security against various kinds of attacks, and vice versa. In other words, the performance comparison should be done by considering both security and usability simultaneously.

Another problem with the basic PAS scheme is that it requires, probably, too long passwords. For the default setting, each user has to remember two cell indices and two words of length 10. In total there are 4 digits and 20 characters to be remembered. Although Bai et al. discussed several ways to create easily memorable and still strong passwords, we doubt if they indeed work in reality for average users. In [17] it was not reported if the participants in the user study had difficulties choosing their passwords and how likely it was for them to forget their passwords. According to a large-scale user study on web password habits [35], the average password length is around 6 to 9 and passwords longer than 13 characters are rare. Hence, it remains a question if 4 digits plus 20 characters are indeed usable.

In case the usability of the basic PAS scheme may be a problem, the extended PAS scheme seems even more difficult for average users to handle. Even when $k = 2$, the average login time will be at least doubled, which is about 2×84 seconds ≈ 2.8 minutes, exceeding the upper bound of more than half of the average users according to the user study reported in [17]. In addition, if the value of len remains the same, the number of digits and characters to be remembered will also be doubled. By using a smaller value of len , the memorability problem can be relaxed, but it has no obvious influence on the average login time, which does not depend on the value of len . Furthermore, we expect the error rate will also significantly increase due to the added complexity of handling more terms in each predicate.

To sum up, although we cannot definitely say if the basic PAS scheme is usable or not, it is clear that the extended PAS scheme is not usable as an acceptable solution against passive adversaries for most average users. Because of this fact, in the next section we will focus our attention mainly on the basic PAS scheme.

3.3 A Probabilistic Attack

Our security analysis given in the previous section has shown that security of the PAS scheme is much weaker than claimed in [17]. In Section 3.2.1, we also showed that the number of candidate predicates decreases exponentially as t increases. For the default setting of the basic PAS scheme, the predicate pair used can be uniquely determined with high probability when $t = 2$, since $1 + ((25 \times 26)^2 - 1)/2^{2 \times 5 \times 2} \approx 1.4029 < 2$.

This leads to partial breaking of the password. To avoid information leakage from the observed responses, Bai et al. proposed to set $t_{max} = 1$. With this setting, on average one will get $1 + ((25 \times 26)^2 - 1)/2^{2 \times 5} \approx 413.6$ predicate pairs for each observed session. Since the predicate pairs used for different authentication sessions are different, it seems impossible to break any part of the password when $t_{max} = 1$.

In this section, we propose a probabilistic attack that is still able to partially break the password even when $t_{max} = 1$ is used. The key point is that the same set of cell indices appear in the p -predicate vectors used for different authentication sessions. This makes it possible to further exploit the correlation among different p -predicate vectors to get more information about the secret cell indices, which can then be used to further refine the set of candidate p -predicate vectors obtained from each observed authentication session. When the number of observed authentication sessions is sufficiently large, we may be able to uniquely determine the cell indices. The probabilistic nature of the attack allows us to guess the cell indices even when the number of observed authentication sessions is not high enough. After determining the cell indices, some secret characters may also be uniquely determined or there are only a few candidates left.

The success rate of the attack smoothly increases as the number of observed authentication sessions increases. For the default setting of the basic PAS scheme, experimental results show that only 7 observed authentication sessions are enough to achieve a success rate higher than 50%, which refutes the claim that the password can be used for at least 10 times before renewal. Even with only two observed authentication sessions, the success rate is not negligible – around 3.5%. The probabilistic attack is also computationally efficient. Its maximal complexity is always strictly smaller than the complexity of the brute force attack.

In the following part of this section, we describe how the attack works, and give some theoretical analyses on the probabilities involved and the computational complexity of the attack. Experimental results are given to demonstrate the feasibility of the proposed attack on the default setting of the basic PAS scheme. Finally, we show the consequence of breaking the secret cell indices is that the PAS scheme is downgraded to a challenge-response protocol working like a one-time password (OTP) system but with worse usability and security.

3.3.1 Description of the Attack

The probabilistic attack described next is a form of intersection attack sketched in Chapter 2. This is a good example to show how the intersection attack need not be equivalent to a brute force attack. To simplify the description of the probabilistic attack, we show how it works for the basic PAS scheme when the adversary/attacker knows the value of len . In this case, given $\hat{t} \geq 1$ observed authentication session(s), a step-by-step description of the probabilistic attack is as follows:

Attack: Probabilistic Attack.

Input: $\hat{t} \geq 1$ authentication sessions.

Output: Candidates for the secret characters.

- 1: **for all** observed authentication sessions **do**
- 2: obtain a set of p -predicate vectors agreeing with all the n_r challenge-response pairs. Denote all the \hat{t} sets by \mathbb{P}_i , $i = 1, \dots, \hat{t}$.
- 3: **for all** p -predicate vectors ($\text{pred}_1, \dots, \text{pred}_p$) in \mathbb{P}_i **do**
- 4: extract the cell-index part to get a p -tuple cell-index vector (c_1, \dots, c_p) . All the p -tuple cell-index vectors form a new set \mathbb{C}_i .
- 5: Calculate $\mathbb{C}^* = \bigcap_{i=1}^{\hat{t}} \mathbb{C}_i$.
- 6: Use \mathbb{C}^* to refine each set \mathbb{P}_i and get a new set as follows: $\mathbb{P}^* = \{x = (c_i, h_i) | x \in \mathbb{P} \wedge c_i \in \mathbb{C}^*\}$.
- 7: **if** $|\mathbb{C}^*| = 1$ **then**
- 8: all the p secret cell indices can be immediately determined, and thus some candidates of those secret characters in \mathbb{P}_i^* corresponding to the secret cell indices can also be obtained.
- 9: **else if** $|\mathbb{C}^*| > 1$ **then**
- 10: count the number of times each cell-index vector occurs in $\mathbb{P}_1^*, \dots, \mathbb{P}_{\hat{t}}^*$ and rank the cell-index vectors in order of their occurrence. All cell-index vectors that are ranked first are the candidates for the secret cell-index vector. All characters in $\mathbb{P}_1^*, \dots, \mathbb{P}_{\hat{t}}^*$ that correspond to these candidates cell-index vectors are then the candidates for the secret characters.

In the proposed attack, Steps 1 and 2 correspond to the brute force attack targeting each p -predicate vector, and Steps 3-6 exploit the correlation existing between different p -predicate vectors (i.e., the static cell-index vector). Steps 7-10 have two different cases, according to the cardinality of \mathbb{C}^* . The ranking based strategy in Step 10 is justified by the fact that the secret cell-index vector appears to occur most frequently, since it occurs at least once while others may never occur. A more detailed analysis on this ranking probability will be discussed in Section 3.3.2. Step 10 is the main part to make the attack work in a probabilistic manner.

The proposed probabilistic attack also works for the extended PAS scheme. It can be done by simply replacing c_i with $\{c_{i,1}, \dots, c_{i,k}\}$. However, due to the usability problem with the extended PAS scheme (recall Section 3.2.5), we will only discuss the basic PAS scheme in the following theoretical and experimental analysis on the performance of the probabilistic attack.

3.3.2 Theoretical Analysis

In this subsection, we show some theoretical analyses on Steps 7-10 of the attack.

Number of Observed Authentication Sessions to have $|\mathbb{C}^*| = 1$

First let us investigate how many observed authentication sessions will ensure that $|\mathbb{C}^*| = 1$ happens with high probability. According to our discussion in Section 3.2.1,

the probability that each incorrect p -predicate vector will remain in \mathbb{P}_i is $1/2^{ln_r}$. Then, we can derive:

$$\Pr[|\mathbb{P}_i| = a + 1] = \binom{N_1}{a} \left(\frac{1}{2^{ln_r}} \right)^a \left(1 - \frac{1}{2^{ln_r}} \right)^{N_1-a}$$

where $0 \leq a \leq N_1$ and $N_1 = (MH)^p - 1$. Note that the correct p -predicate vector is always in \mathbb{P}_i , so $|\mathbb{P}_i| \geq 1$.

Given a set \mathbb{P}_i of size $a + 1$, let us estimate the probability that an incorrect p -tuple cell-index vector (c_1, \dots, c_p) belongs to \mathbb{C}_i under the assumption that all incorrect p -predicate vectors appear in \mathbb{P}_i with equal probability. To facilitate the following discussion, denote the probability by $\rho_0(a)$. When $a > N_1 - H^p$, we can see $\rho_0(a) = 1$, since there can be a maximum of $N_1 - H^p$ p -predicate vectors with other cell-index vectors. When $a \leq N_1 - H^p$, the probability is:

$$\rho_0(a) = 1 - \frac{\binom{N_1-H^p}{a}}{\binom{N_1}{a}} = 1 - \prod_{i=0}^{a-1} \left(1 - \frac{H^p}{N_1 - i} \right)$$

Based on the above results, for a randomly generated set \mathbb{P}_i whose size is unknown, the probability that an incorrect cell-index vector (c_1, \dots, c_p) belongs to \mathbb{C}_i is as follows:

$$\begin{aligned} \rho &= \Pr[(c_1, \dots, c_p) \in \mathbb{C}_i] \\ &= \sum_{a=0}^{N_1} \rho_0(a) \cdot \Pr[|\mathbb{P}_i| = a + 1]. \end{aligned} \quad (3.2)$$

Assuming the above probability ρ does not depend on the subscript i , we get:

$$\Pr[(c_1, \dots, c_p) \in \mathbb{C}^*] = \prod_{i=1}^{\hat{t}} \Pr[(c_1, \dots, c_p) \in \mathbb{C}_i] = \rho^{\hat{t}}$$

Then, we can further derive the probability that $|\mathbb{C}^*| = 1$ as the probability that none of the $M^p - 1$ incorrect cell-index vectors is in \mathbb{C}^* : $\Pr[|\mathbb{C}^*| = 1] = \left(1 - \rho^{\hat{t}} \right)^{M^p-1}$. Let $\Pr[|\mathbb{C}^*| = 1] \geq q$, we can derive the following condition:

$$\hat{t} \geq \left\lceil \log_{\rho} \left(1 - q^{\frac{1}{M^p-1}} \right) \right\rceil$$

Once the parameters of the basic PAS scheme are all given, one can immediately estimate the value of ρ and then calculate the minimal value of \hat{t} corresponding to any threshold probability q . For the default parameters, we can calculate $\rho = 0.4834$. With this value of ρ , Table 3.5 shows the minimal value of \hat{t} that can ensure $|\mathbb{C}^*| = 1$ happens with different probabilities. We can see that on average 10 observed authentication sessions are enough to uniquely determine the secret cell indices with Steps 7 and 8.

TABLE 3.5: The minimal value of \hat{t} against q to ensure $\Pr[|\mathbb{C}^*| = 1] \geq q$.

q	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\hat{t} \geq$	7	8	8	9	9	9	10	10	11	11	12

Ranking Probability in Steps 9 and 10

The data in Table 3.5 show that Steps 7 and 8 are not able to effectively reduce the number of observed authentication sessions. When $q = 0.5$, we need 10 observed authentication sessions, which is the maximal number before password renewal. This does not make too much sense. Although we may also be able to break the password with 7 observed authentication sessions, the probability is a bit too low. Steps 9 and 10 can help the attack work with even less than 7 observed authentication sessions, and also with a nontrivial success rate.

To theoretically analyze the ranking probability problem involved in Step 10, we first need to estimate the size of \mathbb{P}_i^* . Assuming the number of incorrect p -predicate vectors in \mathbb{P}_i decreases with the same rate as the number of incorrect cell-index vectors in \mathbb{C} , i.e., $(|\mathbb{P}_i^*| - 1)/(|\mathbb{P}_i| - 1) = (|\mathbb{C}^*| - 1)/(|\mathbb{C}| - 1) = \rho^{\hat{t}}$, we can have $|\mathbb{P}_i^*| = 1 + \rho^{\hat{t}}(|\mathbb{P}_i| - 1)$. Let $E[.]$ denote expected value. Since $E[|\mathbb{P}_i|] = 1 + N_1/2^{ln_r}$, we get $E[|\mathbb{P}_i^*|] = 1 + \rho^{\hat{t}}N_1/2^{ln_r}$.

After we have the estimation of $|\mathbb{P}_i^*|$, we want to know the probability that in $\sum_{i=1}^{\hat{t}} |\mathbb{P}_i^*|$ p -predicate vectors the number of times the correct p -predicate vector occurs is not less than any of the incorrect ones. Thus, we need to solve the following mathematical problem:

There are $N = M^p$ types of objects. Type-1 objects occur with probability $q_1 = (H^p - 1)/N_1$, and all other objects occur with probability $q_0 = H^p/N_1$. Randomly pick $L = \sum_{i=1}^{\hat{t}} (|\mathbb{P}_i^*| - 1)$ objects with the above probabilities and add \hat{t} more type-1 object(s), what is the probability that the number of type-1 object(s) is not less than the number of objects of any other type?

Note that $q_1 + (N - 1)q_0 = 1$ for the above problem. To facilitate our discussion, denote the number of type- i objects in the L objects by $\#(O_i)$. It is not easy to get an explicit solution to the above problem. Now let us try to derive a practical lower bound of the probability. When $L \leq \hat{t}$, $\#(O_i) \leq L \leq \hat{t} \leq \#(O_1) + \hat{t}$ always holds, so $\Pr[\max_{i=2}^N (\#(O_i)) \leq \#(O_1) + \hat{t}] = 1$. When $L \geq \hat{t} + 1$, we have the following result:

$$\begin{aligned}
& \Pr \left[\max_{i=2}^N (\#(O_i)) \leq \#(O_1) + \hat{t} \right] \\
&= 1 - \Pr [\exists i \in \{2, \dots, N\}, \#(O_i) > \#(O_1) + \hat{t}] \\
&\geq 1 - \Pr [\exists i \in \{2, \dots, N\}, \#(O_i) \geq \hat{t} + 1] \\
&= 1 - \Pr \left[\bigvee_{i=2}^N (\#(O_i) \geq \hat{t} + 1) \right]
\end{aligned}$$

TABLE 3.6: Lower bounds of $\Pr [\max_{i=2}^N (\#(O_i)) \leq \#(O_1) + \hat{t}]$ against \hat{t} .

\hat{t}	1	2	3	4	5	6	7	8	9	10
The theoretical lower bound	0	0	0.9473	0.9997	1	1	1	1	1	1
Experimental result	0.0504	0.2915	0.9604	0.9999	1	1	1	1	1	1

$$\begin{aligned}
 &\geq 1 - \min \left(1, \sum_{i=2}^N \Pr [\#(O_i) \geq \hat{t} + 1] \right) \\
 &= 1 - \min \left(1, (N-1) \sum_{i=\hat{t}+1}^L \binom{L}{i} q_0^i (1-q_0)^{L-i} \right). \tag{3.3}
 \end{aligned}$$

When \hat{t} is close to 1, the above lower bound is generally equal to 0, which does not make much sense. But as \hat{t} becomes larger, the lower bound quickly converges to 1. Taking the default parameters of the basic PAS scheme and assuming:

$$L = E \left[\sum_{i=1}^{\hat{t}} (|\mathbb{P}^*| - 1) \right] = \hat{t} \rho^{\hat{t}} \frac{N_1}{2^{\ln r}}$$

we calculated the above lower bound for $\hat{t} = 1, \dots, 10$. For each value of \hat{t} , 10000 random experiments were also made to see how large the probabilities are. Table 3.6 shows the results.

Following a similar argument, we can also estimate the following inequality:

$$\begin{aligned}
 &\Pr [\exists i \in \{2, \dots, N\}, \#(O_i) \geq \#(O_1) + \hat{t}] \\
 &\leq \Pr [\exists i \in \{2, \dots, N\}, \#(O_i) \geq \hat{t}] \\
 &= \Pr \left[\bigvee_{i=2}^N (\#(O_i) \geq \hat{t}) \right] \\
 &\leq \min \left(1, \sum_{i=2}^N \Pr [\#(O_i) \geq \hat{t}] \right) \\
 &= \min \left(1, (N-1) \sum_{i=\hat{t}}^L \binom{L}{i} q_0^i (1-q_0)^{L-i} \right). \tag{3.4}
 \end{aligned}$$

Then, assuming there are N_{max} cell-index vectors occurring most often in $\mathbb{P}_i^*, \dots, \mathbb{P}_{\hat{t}}^*$, i.e., N_{max} is the cardinality of the set $\{i | \#(O_i) = \max_{j=1}^{M^p} \#(O_j)\}$, we can get an upper bound of its expected value as:

$$E[N_{max}] \leq 1 + (M^p - 1) \cdot \min \left(1, (N-1) \sum_{i=\hat{t}}^L \binom{L}{i} q_0^i (1-q_0)^{L-i} \right)$$

TABLE 3.7: Theoretical upper bounds of $E[N_{max}]$ and estimated values.

\hat{t}	1	2	3	4	5	6	7	8	9	10
The theoretical upper bound	625	625	607.1	6.842	1.012	1	1	1	1	1
Experimental result	3.6846	3.6184	1.7168	1.0086	1	1	1	1	1	1

For the default setting of the PAS scheme and $\hat{t} = 1, \dots, 10$, Table 3.7 shows the theoretical upper bound and the estimated value from 10000 random experiments. The data in Tables 3.6 and 3.7 imply that one can recover the secret cell-index vector with high probability with only 3 observed authentication sessions.

3.3.3 Time Complexity of the Attack

The computational complexity of the proposed probabilistic attack can be calculated by summing up the complexities of groups of steps. The complexity of Steps 1-2 is $\hat{t}(MH)^p$, which is the maximal number of p -predicate vectors one has to check for all the \hat{t} observed authentication sessions to get \mathbb{P}_i . After Step 1 (and 2) is finished, the average size of each \mathbb{P}_i is $1 + N_1/2^{ln_r}$, so the average complexity of Steps 3-6 is $\hat{t}(1 + N_1/2^{ln_r})$. The complexity of Steps 7-8 is very small, so it can be omitted. The ranking done in Steps 9-10 has complexity $\sum_{i=1}^{\hat{t}} |\mathbb{P}_i^*| = \hat{t}(1 + \rho^{\hat{t}} N_1/2^{ln_r})$. The worst-case complexities of Step 3-6 and 9-10 are always less than the complexity of Steps 1-2. As a whole, we can see the overall complexity of the attack is determined by Steps 1-2, which has an upper bound $\mathcal{O}(\hat{t}(MH)^p)$. For the default setting of the basic PAS scheme and $\hat{t} = 4$, the complexity is $\mathcal{O}(\hat{t}(MH)^p) \approx 2^{20.7}$.

Recalling the size of the password space of the basic PAS scheme, i.e., $len \cdot (MH)^p$, we can see the complexity of the probabilistic attack is always strictly smaller (although not by too much) than that of the brute force attack since $\hat{t} < len$ always holds. Note that when $\hat{t} = len$ one does not need to break the system, since all the predicate indices have been used up and the password has already been renewed.

3.3.4 Experimental Results

Based on the theoretical analysis and the complexity estimate of the probabilistic attack, we can see that the attack is feasible as long as $(MH)^{pk}$ is not too large (between 2^{70} to 2^{80}). This condition is satisfied for the default setting of the PAS scheme. In fact, it has to be so, because all the parameters involved (especially p and k) cannot be too large to ensure an acceptable level of usability.

We developed a MATLAB implementation of the basic PAS scheme with $p = 2$, and tested the real performance of the proposed probabilistic attack. On a computer equipped with a 2.4GHz Intel Core 2 Duo processor, and 2GB memory, one successful attack with \hat{t} observed authentication sessions consumes only around $5\hat{t}$ seconds.

TABLE 3.8: Success rate of finding the secret and estimated number of candidates.

\hat{t}	1	2	3	4	5	6	7	8	9	10
Success rate	0.012	0.035	0.071	0.13	0.24	0.41	0.60	0.76	0.86	0.94
Number of candidates	3.01	2.51	2.02	1.73	1.51	1.36	1.23	1.10	1.03	1.01

The statistical results of 1000 real attacks targeting the default setting of the basic PAS scheme are shown in Table 3.8. It turned out that the real performance is worse than the theoretical analysis obtained in Section 3.3.2. We attribute this to the deviation of real attacks from some of the theoretical assumptions we made in the theoretical analysis in Section 3.3.2. For instance, we calculate the values in Table 3.6 by assuming $E[|\mathbb{P}_i^*|] = 1 + \rho^{\hat{t}} N_1 / 2^{\ln r}$ and $L = \hat{t} \rho^{\hat{t}} N_1 / 2^{\ln r}$, but in practice their values vary in a wide range around the means. Despite the mismatch between Table 3.8 and Table 3.6, we can see the success rate of breaking the secret cell-index pair and the average number of candidates follow the same pattern as the data in Table 3.7. The experimental data in Table 3.8 clearly show that with 7 observed authentication sessions one can break the secret cell-index pair with probability greater than 50%. Even with only two observed authentication sessions, the success rate is high enough (3.5%) to threaten a considerable percentage of users. Recall that for sufficient security, we require the success probability to be less than one in a million.

3.3.5 Consequences of the Probabilistic Attack

Note that it is impossible and unnecessary to break the whole password with the probabilistic attack, since some secret characters will never occur until the last authentication session. In fact, the main consequence of breaking the secret cell indices is the following: the password becomes a set of len words $\{W_j^* = w_1[j] \cdots w_p[j]\}_{j=1}^{\text{len}}$, each of which is used for exactly one authentication session. After all the len words $\{W_j^*\}_{j=1}^{\text{len}}$ are used up, a new password (i.e., a new set of len words) have to be issued to the user. Clearly, this means the PAS scheme now works essentially like a one-time password (OTP) system, where each word W_j^* is the OTP used for each authentication session and expires immediately after being used.

The degradation of the PAS scheme to an OTP-like system has several consequences. First, this fact disqualifies the PAS scheme as a better solution over common OTP systems against the targeted passive adversaries. Second, the downgraded PAS scheme is still a challenge-response protocol, which asks the user to go through the same process as in the original PAS scheme. In comparison, common OTP systems are not based on a challenge-response structure² and the user is simply asked to input

²Although we can define an OTP as a challenge-response protocol, where the challenge is the prompt and the response is the password itself, what we essentially want to say here is that there is no real computation done by the user in an OTP except for recalling and typing the password.

the dynamic password in an input box, so the usability is much better. Third, the downgraded PAS scheme offers lower security against random guess attack. Recalling our analysis given in Section 3.2.3, we can derive that the success rate of randomly guessing the predicate (which is reduced to be the word W_j^*) is:

$$1 \cdot \frac{1}{\binom{MH+k-1}{k}^p} + \frac{1}{2^{ln_r}} \cdot \frac{\binom{MH+k-1}{k}^p - 1}{\binom{MH+k-1}{k}^p} = \frac{1}{2^{ln_r}} + \frac{2^{ln_r} - 1}{2^{ln_r} \binom{MH+k-1}{k}^p} \quad (3.5)$$

Comparing the above equation with Equation (3.1), we can see the success rate becomes larger due to the lack of M in the denominator of the second term. For the default setting of the PAS scheme, Equation (3.1) equals to $\frac{1}{2^{10}} + \frac{2^{10}-1}{2^{10} \times (25 \times 26)^2} \approx 9.7893 \times 10^{-4}$, but Equation (3.5) equals to $\frac{1}{2^{10}} + \frac{2^{10}-1}{2^{10} \times 26^2} \approx 2.4544 \times 10^{-3}$, nearly 2.5 times larger. To maintain the same level of security against random guess attack, the parameter values have to be increased accordingly, which will make usability even worse.

3.4 Conclusion

In this chapter, we have re-evaluated the security of the predicate-based authentication service (PAS) presented by Bai et al. [17]. PAS can be considered a human identification protocol according to our definition of such protocols in Chapter 2. PAS was designed such that it could be used for at least 10 observed authentication sessions (with default parameter values). Although, this number is already lower, we have shown that one can break the scheme with even fewer observed sessions. More specifically, with 7 observed sessions there is a 50% chance of obtaining the secret, and even with 2 observed sessions there is a 3.5% chance that the adversary can obtain the secret. The PAS scheme is insecure against both brute force attack and a probabilistic attack. Lack of security against brute force attack is due to the very small size of the secret space. The probabilistic attack can break part of the password even with a small number of observed authentication sessions, as already mentioned. The breaking of part of the password downgrades the PAS scheme to a one-time password (OTP) like system, thus nullifying its main advantages over common OTP systems. It is possible to enhance security of the PAS scheme by increasing the values of some parameters, unfortunately, which will definitely decrease the usability and make the system not useful as a practical solution.

This chapter demonstrates that without carefully analysing the underlying mathematical structure, a human identification protocol might succumb to simple yet innovative attacks, even if it is designed to be secure for a handful of authentication sessions. Perhaps the reason why much of the original analysis by Bai et al. [17] was inaccurate is because PAS does not seem to be based on a clearly defined underlying mathematical problem. Without such mathematical structure, it is not easy to comprehensively analyse the protocol's security. It is more desirable, then, to construct protocols that are constructed from some clearly defined mathematical problem, since then the security can be thoroughly analysed against the (conjectured) hardness of solving the problem. The next chapter discusses the security of a protocol from Sobrado and Birget [9, 21]

based on an apparently hard to solve geometric problem. At first glance, it appears that the protocol might resist all forms of intersection attacks. But as we shall see, due to the structure of the geometric problem, information about the secret is leaked, and part of the secret can be revealed after a handful of observed identification sessions.

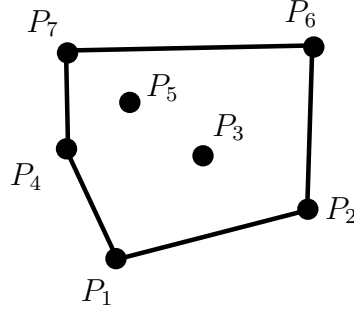
4

Security Analysis of CHC (Convex Hull Click)

Sobrado and Birget recently proposed a convex hull based human identification protocol [21], whose steps can be performed by humans without additional aid. The main part of the protocol involves the user mentally forming a convex hull of secret icons in a set of graphical icons and then clicking randomly within this convex hull. While some rudimentary security issues of this protocol have been discussed, a comprehensive security analysis has been lacking. In this chapter we analyze the security of this convex hull based protocol. In particular, we show two probabilistic attacks which reveal the user's secret after the observation of only a handful of authentication sessions. These attacks can be efficiently implemented as their time and space complexities are considerably less than the brute force attack. We show that while the first attack can be mitigated through appropriately chosen values of system parameters, the second attack succeeds with a non-negligible probability even with large system parameter values which make the protocol unusable.

In [9] Wiedenbeck et al. gave a detailed description of the protocol from [21], with a usability analysis employing human participants. Since the work reported in [9] is more comprehensive, we will adhere to the protocol described therein for our security analysis in this chapter. Following the term used in [9], we call the protocol Convex Hull Click or CHC in short. The protocol can be roughly described as follows: in the setup phase, the user and the server share a subset of graphical icons as a secret. As in all human identification protocols, the setup phase is assumed to take place in a secure setting, outside the reach of any adversaries. In an identification session, the server shows a screen of randomly placed graphical icons. The user mentally forms a convex hull of the secret graphical icons and then clicks a random point inside this convex hull.

This chapter is organized as follows. In Section 4.1, we give a detailed account

FIGURE 4.1: The convex hull of a set of points Π .

of the protocol. Section 4.2 describes first of the two attacks. We digress in Section 4.3 to explore the geometric structure of convex hulls in an attempt to estimate the information leakage of the secret once a challenge-response pair has been observed. Section 4.4 describes, in detail, the main attack on CHC which is followed by concluding remarks in the last section.

4.1 The CHC Human Identification Protocol

We begin with the definitions of polygons and convex hulls [36].

Definition 5 (Polygon). *A polygon is a piece-wise linear, closed curve in a plane. The straight line segments forming the closed curve are called the sides of the polygon. A point joining two consecutive sides is called a vertex. A polygon is simple if it does not cross itself.*

Definition 6 (Interior, Exterior and Boundary). *The set of points in the plane that lie outside a simple polygon is called its exterior; the set of points lying on the polygon form its boundary and the set of points inside the boundary of the polygon is called its interior. If a point P lies on the boundary or in the interior of a polygon, we say that the polygon contains P or P is contained in the polygon.*

Definition 7 (Convex Polygon). *A simple polygon is convex if all points on the line segment joining any two points in its boundary or interior are contained in the polygon.*

Definition 8 (Convex Hull). *The convex hull of a set of points Π , is the smallest convex polygon for which every point in Π is contained in the polygon.*

Figure 4.1 shows the convex hull of the set of points $\Pi = \{P_1, P_2, \dots, P_7\}$. We shall denote the convex hull of a set of points Π by $\text{ch}(\Pi)$. We denote the membership relation “contains” by \in . For instance, in Figure 4.1, $P_i \in \text{ch}(\Pi)$, for $1 \leq i \leq 7$. The convex hull of 3 points is a triangle. Hence, we will use the terms convex hull and triangle interchangeably for the case of 3 points. We describe the CHC human identification protocol next.

4.1.1 The Protocol

In the CHC human identification protocol, initially, \mathcal{H} and \mathcal{C} choose k graphical icons from a set of n . These k icons constitute the shared secret between the two parties. As an example, k can be 5 and n can be 100. This is called the setup phase. When \mathcal{H} wants to prove his identity to \mathcal{C} , the following protocol is carried out.

Protocol: Convex Hull Click (CHC).

Setup: \mathcal{H} and \mathcal{C} share a set of k graphical icons out of n as a secret.

- 1: \mathcal{C} randomly samples a set of m graphical icons out of n . Here, m is a random positive integer between n and some lower bound m_{\min} . \mathcal{C} ensures that at least 3 of the k secret icons are included in these m graphical icons. These icons are distributed randomly on the screen of the user's computer terminal within a rectangular frame and aligned in a grid.
- 2: \mathcal{H} mentally computes the convex hull of any 3 secret icons displayed on the screen and randomly clicks a point contained in this convex hull. \mathcal{H} does not need to click on the icons themselves. \mathcal{H} can click anywhere on the screen in the interior or boundary of this convex hull. Notice that this is equivalent to clicking on the convex hull of all the secret icons present in the screen.
- 3: \mathcal{C} repeats the process a certain number of times and accepts or rejects \mathcal{H} if all replies are correct.

¹For the ease of analysis, we make some assumptions as follows.

- Instead of choosing m randomly each time, we assume it to be fixed. In fact, we will later see that once n and k are fixed, we do not have much freedom in choosing m , if a certain attack is to be avoided.
- We replace graphical icons by non-negative integer lattice points on a real plane, enclosed within a rectangular area. The lattice points are identified by a unique integer label from the set $\{1, 2, \dots, n\}$. Notice that, graphical icons are displayed for the ease of humans. Thus, from an analytical point of view, the two representations are equivalent. Throughout this text, we will use the terms, icons and labels, interchangeably.
- One round of the protocol will thus constitute the positive quadrant of the real plane. The m graphical icons are replaced by randomly placed integer lattice points on this quadrant, each one having a unique label. The user's set of secret icons is thus also a set of integer labels; see Figure 4.2. We shall call the area enclosed in the rectangle as the rectangular lattice area or simply the rectangle.

¹There can be a slight difference in the real setting when it comes to clicking on the boundary of a geometric object. It is arguably hard for a user to click on the boundary of convex hulls with accuracy. It is easy to see that a convex hull of more than 3 secret icons contains the convex hull of

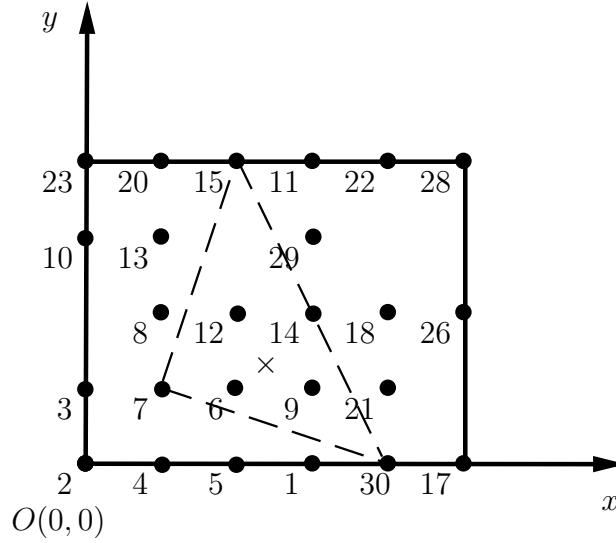


FIGURE 4.2: One iteration of the Convex Hull Click protocol.

Example 1. Suppose $n = 30$ and $m = 25$. Further, suppose $k = 4$, and \mathcal{H} and \mathcal{C} share the secret $\{7, 15, 27, 30\}$. Figure 4.2 shows one iteration of the protocol. Since the challenge only contains 7, 15 and 30 from the set of secret labels, \mathcal{H} forms the convex hull of the points corresponding to these labels and outputs a random point contained in this convex hull. This point is depicted by the symbol \times in the figure. Note that this point is not necessarily a point on the lattice. We can consider this to be a point in the real plane. That is, it belongs to \mathbb{R}^2 . \square

4.1.2 Mitigating Random Guesses

Recalling our description of challenge-response protocols, the challenge in the CHC protocol is a screen full of graphical icons (or labels) and the corresponding response is a point $P \in \mathbb{R}^2$. The number of challenge-response pairs in an authentication session is chosen such that the probability of the adversary \mathcal{A} impersonating \mathcal{H} with random clicks is very small (random guess attack). We denote this fixed number by r_0 (the number of rounds or iterations of the CHC protocol). For example, Wiedenbeck et al. used $r_0 = 10$ [9, §6, pp. 183]. They also mentioned that the implementation of their protocol ensures that convex hulls of secret icons occupying more than half the screen are rare [9, §3, pp. 180].² Thus, we can assume that the average probability of success

any 3 secret icons. Therefore, the boundary points of the latter convex hull might be in the interior of the former. Thus, these boundary points will be easier for the user to click in the former case. To simplify our analysis in this paper, we ignore the difference between the two settings.

²If the challenge is generated as mentioned in the protocol description, then there is a chance that the probability that a random click is contained in the convex hull of secret icons is greater than $1/2$. However, through our experimental results, we found that for the values of system parameters used

of \mathcal{A} in impersonating \mathcal{H} , through random clicks, is less than $(\frac{1}{2})^{r_0}$.

4.2 Attack 1: Difference in Distributions

Our first observation is that \mathcal{C} has to ensure that at least 3 out of k secret labels are displayed on the screen. There is no such restriction on the non-secret labels. Naturally, this may lead to two different probabilities for the secret and non-secret labels. The probabilities depend on how the random challenges are generated. There are several possible ways to generate random challenges. To simplify our discussion, in this paper we consider the following approach: first generate a random number $l \in \{3, \dots, k\}$, then randomly select l secret labels and $m - l$ non-secret labels to form the challenge. Note that this approach was also the one adopted in the implementation from [9].

We now calculate the probability of generating a secret label and compare it with the probability of generating a non-secret label. Let N denote the set of all labels and let K denote the set of secret labels. Thus, $|N| = n$ and $|K| = k$. Let \bar{K} denote the set of non-secret labels. Thus, $K \cup \bar{K} = N$. We assume that the adversary \mathcal{A} has observed $r \geq 1$ challenges sent from \mathcal{C} to \mathcal{H} . For each of these challenges, we denote the set of secret labels appearing in the challenge by K_j and the set of non-secret labels by \bar{K}_j , for $1 \leq j \leq r$. Notice that $|K_j \cup \bar{K}_j| = m$ for all j . In this attack, we do not even require the responses to these challenges. For $1 \leq i \leq n$ and $1 \leq j \leq r$, define the following indicator random variables:

$$S_{i,j} = \begin{cases} 1 & \text{if label } i \text{ appears in challenge } j \\ 0 & \text{otherwise} \end{cases}$$

Then, for any $i \in K$ and any $j \in \{1, \dots, r\}$, we have:

$$\begin{aligned} \Pr[S_{i,j} = 1, i \in K] &= \sum_{l=3}^k \Pr[S_{i,j} = 1 | |K_j| = l] \Pr[|K_j| = l] \\ &= \frac{3}{k} \frac{1}{k-2} + \frac{4}{k} \frac{1}{k-2} + \dots + \frac{k}{k} \frac{1}{k-2} \\ &= \frac{1}{k(k-2)} \left(\frac{k(k+1)}{2} - 3 \right) \end{aligned} \quad (4.1)$$

And for any $i \in \bar{K}$, we have:

$$\begin{aligned} \Pr[S_{i,j} = 1, i \in \bar{K}] &= \sum_{l=3}^k \Pr[S_{i,j} = 1 | |\bar{K}_j| = l] \Pr[|\bar{K}_j| = l] \\ &= \frac{m-3}{n-k} \frac{1}{k-2} + \frac{m-4}{n-k} \frac{1}{k-2} + \dots + \frac{m-k}{n-k} \frac{1}{k-2} \\ &= \frac{1}{k-2} \frac{1}{n-k} (m-3 + m-4 + \dots + m-k) \\ &= \frac{1}{k-2} \frac{1}{n-k} \left(m(k-2) - \frac{k(k+1)}{2} + 3 \right) \end{aligned} \quad (4.2)$$

in this paper, the probability is less than $1/2$. Thus, we do not require any modifications to ensure that this probability is less than $1/2$.

Now, let $S_i^{(r)}$ denote the number of times label i appears in r challenges. Then:

$$E[S_i^{(r)}] = \sum_{j=1}^r E[S_{i,j}]$$

Thus, for $i \in K$, we have:

$$E[S_i^{(r)}, i \in K] = r \Pr[S_{i,j} = 1, i \in K]$$

And for $i \in \overline{K}$, we get:

$$E[S_i^{(r)}, i \in \overline{K}] = r \Pr[S_{i,j} = 1, i \in \overline{K}]$$

Thus, the two expected values will be different, provided the two probabilities in Equations (4.1) and (4.2) are different. For instance, when $n = 112, m = 70, k = 5$ and $r = 100$, we get:

$$E[S_i^{(r)}, i \in K] = (100)(0.8) = 80$$

and:

$$E[S_i^{(r)}, i \in \overline{K}] = (100)(0.6168) = 61.68$$

Hence, in 100 randomly generated challenges, we expect the secret labels to appear around 80 times each and the non-secret labels to appear around 62 times each. This observation immediately leads to the following probabilistic attack.

Attack: Attack 1.

Input: r challenges.

Output: k labels.

- 1: Count the number of times each label appears in the r challenges.
- 2: Output the top k most frequently occurring labels.

The above algorithm has a high success rate provided the two aforementioned probabilities differ considerably. We ran simulations for two different sets of system parameter values and the results are shown in the first two rows of Table 4.1. For each set of values, a total number of 1000 simulated attacks were performed. As can be seen, the algorithm, on average, outputs almost all the secret labels even with only 100 given challenges. And, in both sets of values, the probability of obtaining all k secret labels as the output of Attack 1 is higher than 0.5. Since each identification session contains $r_0 = 10$ challenges, this implies only 10 identification sessions. The set of labels thus obtained can be verified against a few responses corresponding to these challenges. Once the secret labels are obtained, it is trivial for \mathcal{A} to impersonate \mathcal{H} . To avoid this attack, the two probabilities should be equal. This gives the following lemma:

Lemma 1. *If $\Pr[S_{i,j} = 1, i \in K] = \Pr[S_{i,j} = 1, i \in \overline{K}]$ for $j \in [1, r]$, then $n = \frac{2km}{k+3}$.*

TABLE 4.1: Simulation results for Attack 1.

n	m	k	r	Average Number of Secret Labels	Probability of Finding all k Secret Labels
112	70	5	100	4.6	0.622
500	200	12	100	11.4	0.554
112	90	5	100	0.1	0.000
500	313	12	100	0.2	0.000

Proof. From Equations (4.1) and (4.2), we get:

$$\begin{aligned}
& \frac{1}{k-2} \frac{1}{n-k} \left(m(k-2) - \frac{k(k+1)}{2} + 3 \right) = \frac{1}{k(k-2)} \left(\frac{k(k+1)}{2} - 3 \right) \\
\Rightarrow & \frac{1}{n-k} \left(m(k-2) - \frac{k(k+1)}{2} + 3 \right) = \frac{1}{k} \left(\frac{k(k+1)}{2} - 3 \right) \\
\Rightarrow & km(k-2) - \frac{k^2(k+1)}{2} + 3k = \frac{nk(k+1)}{2} - 3n - \frac{k^2(k+1)}{2} + 3k
\end{aligned}$$

This implies that:

$$\begin{aligned}
& km(k-2) = \frac{nk(k+1)}{2} - 3n \\
\Rightarrow & 2km(k-2) = nk^2 + nk - 6n \\
\Rightarrow & 2km(k-2) = n(k^2 + k - 6) \\
\Rightarrow & 2km(k-2) = n(k+3)(k-2) \\
\Rightarrow & \frac{2km}{k+3} = n
\end{aligned}$$

□

The last two rows of Table 4.1 show the results of the simulations with the value of m calculated according to Lemma 1. The results are what we expect if m out of n objects are sampled at random. Thus, this fix prevents this type of attack. Notice that, if the value of m is chosen according to the equation in Lemma 1, the probability of any label appearing in a challenge is m/n . That is, all labels are equally likely to appear in a challenge. This can be verified by direct substitution. Of course the above formula does not always give an integral solution. In that case, the nearest integer value of n or m can be chosen. The resulting probability difference would be statistically small, requiring a huge number of challenges to differentiate. Alternatively, we can only look for integral solutions to the equation, for instance $n = 120, m = 90$ and $k = 6$. In this case, Attack 1 will not work no matter how many challenges are observed.

Readjusted values of Parameters.

Wiedenbeck et al. used the values $n = 112$ and $k = 5$ for their user study. The value of m was dynamic, ranging from 43 to 112 giving an average value of 83 [9, §4.1, pp. 181]. In light of Lemma 1, for $n = 112$ and $k = 5$, we suggest $m = 90$ instead. It should be noted that this only guarantees that the system will be secure against Attack 1, since for such small values brute force attack is feasible. For high security, Wiedenbeck et al. suggest $n = 500$, $m = 200$ and $k = 12$. However, for $m = 200$ and $k = 12$, the value $n = 320$ should be used; and for $n = 500$ and $k = 12$, the value $m = 312.5 \approx 313$ should be chosen. This last value of m can become prohibitive, since it will most probably be hard for an average human user to find secret icons among a pool of icons as large as 300. Thus, Lemma 1 limits the values of system parameters that can be used.

4.3 Number of Candidates Satisfying a Challenge-Response Pair

Before we proceed to the description of our second attack, we would like to try to answer the following question of theoretical interest: given one challenge-response pair, how many convex hulls of three labels contain the response point P ? For simplicity, we assume the response point P to be in \mathbb{R}^2 . As before, if P is contained in the convex hull of the points in S , we denote it by $P \in \text{ch}(S)$.

We see that each convex hull of three lattice points is a 3-combination of labels. Also, only m out of a total of n labels occur in one challenge. Therefore the number of convex hulls of three labels that contain the point P is less than or equal to $\binom{m}{3}$. We assume that all $\binom{m}{3}$ possible 3-combinations of labels are enumerated and let $\Gamma_1, \dots, \Gamma_{\binom{m}{3}}$ denote these 3-combinations. Thus each 3-combination is a set of 3 labels. We define the indicator random variable corresponding to Γ_i by C_i , which is 1 if $P \in \text{ch}(\Gamma_i)$. Let $C = \{\Gamma_i | P \in \text{ch}(\Gamma_i), 1 \leq i \leq \binom{m}{3}\}$. Then, we have that:

$$E[|C||P] = \sum_{i=1}^{\binom{m}{3}} E[C_i|P]$$

And,

$$E[|C|] = \int_R E[|C||P] f_P(P) dP = \int_R \left(\sum_{i=1}^{\binom{m}{3}} E[C_i|P] \right) f_P(P) dP$$

where R denotes the rectangle and $f_P(P)$ is the probability density function of the point P . We assume that the bottom-left corner of the rectangle coincides with the origin of the xy -coordinate system. Let $(a, 0)$ and $(0, b)$ be the coordinates of the bottom-right and top-left corners of the rectangle, respectively. The area of the rectangle is therefore ab . If we assume P to be uniformly distributed over the rectangle, we get:

$$E[|C|] = \frac{1}{ab} \int_R \left(\sum_{i=1}^{\binom{m}{3}} E[C_i|P] \right) dP = \frac{1}{ab} \sum_{i=1}^{\binom{m}{3}} A_i$$

where A_i is the area of $\text{ch}(\Gamma_i)$. Now, let γ be the fraction of the number of convex hulls containing the point P . Then, it can be obtained as:

$$\gamma = \frac{E[|C|]}{\binom{m}{3}}$$

There are two things wrong with this approach. First, the bounding rectangle is chosen such that the number of lattice points it can accommodate is considerably higher than m . Thus $(a+1)(b+1) > m$. This means that the placement of m labels will be different in different challenges, thus giving a different value of γ each time. This feature is included in the CHC protocol to enable humans to conveniently locate the secret icons. Secondly, and more importantly, the distribution of P is not uniform over the rectangle. This is true even if we assume the user to select a point uniformly at random, contained in the convex hull of the secret labels; the points around the boundary of the rectangle have a much lower probability of being chosen, as they are contained in the least number of convex hulls. Figure 4.3 shows the distribution of the point P in a simulation of 10,000 challenges. We chose the parameters: $n = m = 16$, $a = 3$, $b = 3$ and $k = 3$. The point P is generated as a uniform random point contained in the convex hull of any three secret labels. We used Turk's method to compute a random point within a triangle [37] (see Appendix A.1). As the figure shows, the

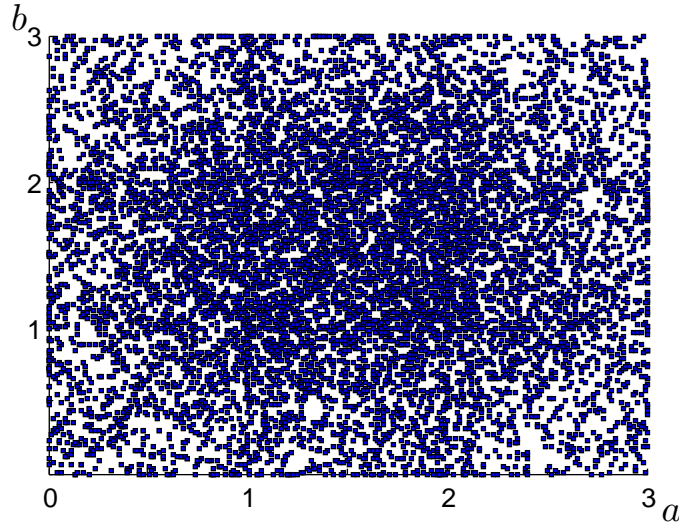


FIGURE 4.3: The distribution of P .

density is lower along the boundaries as compared to the center. For these reasons, we use an experimental approach to find the value of γ . We ran the following algorithm to find an approximate value of γ . We randomly select k out of n labels as a secret to calculate the value of γ .

TABLE 4.2: Values of γ .

Runs	t	n	m	k	a	b	Average γ
10	100	112	90	5	13	13	0.1460
5	100	160	100	12	13	13	0.1479

Algorithm: Find γ .

Input: Parameters n , m , a , b and k ; k secret labels and a precision value t (say $= 100$).

Output: Approximate value of γ .

- 1: **for** $i = 1$ to t **do**
- 2: Generate a random challenge as in the convex hull protocol.
- 3: Form a convex hull of any 3 secret labels.
- 4: Sample a random point P contained in this convex hull.
- 5: Initialize $C \leftarrow 0$.
- 6: **for all** $1 \leq j \leq \binom{m}{3}$ **do**
- 7: check whether the point P is contained in $\text{ch}(\Gamma_j)$. If yes, increment C by 1.
- 8: $\gamma_i \leftarrow C / \binom{m}{3}$.
- 9: Output $\gamma = \frac{1}{t} \sum_{i=1}^t \gamma_i$.

Table 4.2 shows the values obtained for γ for two different sets of parameters. This gives us the result that with these parameter values, we expect approximately 15 percent of the convex hulls of 3 labels in a challenge-response pair to contain the response point. The 3-combinations making up these convex hulls are possible candidates for the secret 3-combination. While the average is around 0.15 for these choices of parameters, there was substantial deviation found in individual values with some values being as low as ≈ 0.016 and some being as high as ≈ 0.25 . This is largely because different challenges have different sizes of convex hulls. As a result, in order to get accurate results, we compute the value of γ for each challenge-response pair separately. Thus $\gamma_1, \gamma_2, \dots$, will now denote the values of γ for challenge-response pairs 1, 2, \dots , respectively.

4.4 Attack 2

One attack mentioned by the authors in [9] is to find all k -combinations of the set of n labels whose convex hull does not satisfy a challenge-response pair (does not contain the response point). Initially, the list contains all k -combinations of n labels. After the observation of each challenge-response pair, the k -combinations of labels, whose convex hull does not contain the response point, are discarded from the list. The sole

remaining k -combination is then the k secret labels of \mathcal{H} . The attack's time and space complexity is $\mathcal{O}(\binom{n}{k})$. Thus, with the values of $n = 320$, $m = 200$ and $k = 12$, the time and space complexity of this attack is roughly 2^{70} , which can be intractable especially in terms of memory resources.

We notice that the user only has to form a convex hull of 3 labels. Thus, in theory, there could possibly be an attack of complexity $\mathcal{O}(\binom{m}{3})$. Our second attack runs within this bound and outputs one of the k secret labels with high probability. The basic idea of the attack is as follows. We first find all candidate 3-combinations, i.e., all 3-combinations of labels whose convex hull contains the point P corresponding to a challenge-response pair. Next, we construct a frequency list that maintains the record of the number of candidate 3-combinations in which each label appears. We update the frequency list by including more challenge-response pairs and seeing if the candidate 3-combinations also satisfy these challenge-response pairs. Finally, the label that appears with the highest frequency is the output of the attack. Section 4.4.3 explains in detail why the output is one of the secret labels with high probability. Once one or more secret labels are obtained, the adversary can impersonate \mathcal{H} . Notice that this can be done even with less than k secret labels. Section 4.4.4 describes how this is achieved. Note that human users tend to remember multiple icons with some hints. It is quite likely that they will select all secret icons that belong to the same category, e.g., icons of software or national flags. In this case, revealing part of the secret will lead to a better guess of the whole set of secret icons.

4.4.1 The Attack

We now describe the attack formally, and do a preliminary analysis followed by a detailed description of why the attack works. As before, we assume that all $\binom{m}{3}$ possible 3-combinations of labels are enumerated and let $\Gamma_1, \dots, \Gamma_{\binom{m}{3}}$ denote these 3-combinations. Thus, each 3-combination is a set of 3 labels.

Attack 2.

Input: r challenge-response pairs with response points P_1, \dots, P_r , respectively, and a threshold τ .

Output: Label(s) with maximum frequency.

- 1: *Test Set.* Initialize $C \leftarrow \phi$. For $1 \leq i \leq \binom{m}{3}$, if $P_1 \in \text{ch}(\Gamma_i)$, then $C \leftarrow C \cup \{\Gamma_i\}$.
- 2: *Frequency List.* For each $\Gamma \in C$, initialize $\text{freq}(\Gamma) \leftarrow 1$.
- 3: **for** $i = 2$ to r **do**
- 4: For each $\Gamma \in C$, if $P_i \in \text{ch}(\Gamma)$, then $\text{freq}(\Gamma) \leftarrow \text{freq}(\Gamma) + 1$.
- 5: *Thresholded Subset.* $C^{(\tau)} \leftarrow \{\Gamma \in C \mid \text{freq}(\Gamma) > \tau\}$.
- 6: *Frequency of labels.* for each distinct label l in $C^{(\tau)}$ compute:

$$\text{freq}(l) \leftarrow \sum_{\Gamma \in C^{(\tau)} \mid l \in \Gamma} \text{freq}(\Gamma)$$

7: output all labels l' such that $\text{freq}(l') = \max_{l \in C^{(\tau)}} \{\text{freq}(l)\}$.

The time complexity of the above attack is $\mathcal{O}(\binom{m}{3})$ or $\mathcal{O}(m^3)$. The space complexity is $\mathcal{O}(\gamma \binom{m}{3})$. Continuing with our theoretical treatment in the previous section, we would like to first analyze the expected sizes of the frequency lists before we detail the simulation results of Attack 2 and the reasons for its high success probability.

Let $F^{(i)} = \sum_{\Gamma \in C} \text{freq}(\Gamma)$, denote the cumulative frequency after the i th challenge-response pair. We have seen earlier that:

$$E[F^{(1)}] = E[|C|] = \gamma_1 \binom{m}{3}$$

That is, the expected size of the Test Set is as above. Also, let:

$$L^{(i)} = \sum_{\Gamma \in C | l \in \Gamma} \text{freq}(\Gamma)$$

denote the frequency of a label after the i th challenge-response pair. For $L^{(1)}$, we see that each label can occur with $\binom{m-1}{2}$ combinations of the remaining labels. Assuming all these combinations to be uniformly distributed, each combination will have a probability γ_1 of being in C . Thus:

$$E[L^{(1)}] = \gamma_1 \binom{m-1}{2}$$

The above two results can also be obtained differently. Consider the indicator random variable $Y_{i,j}$ which is 1 if $P_i \in \text{ch}(\Gamma_j)$. Also, let $X_{i,j}$ be the indicator random variable which is 1 if Γ_j exists in challenge i (Since $m \leq n$, the j th combination might not even exist in challenge i). Then, we can see that:

$$\begin{aligned} E[Y_{1,j}] &= \Pr[Y_{1,j} = 1] = \Pr[Y_{1,j} = 1 | X_{1,j} = 1] \Pr[X_{1,j} = 1] \\ &+ \Pr[Y_{1,j} = 1 | X_{1,j} = 0] \Pr[X_{1,j} = 0] \\ &= \Pr[Y_{1,j} = 1 | X_{1,j} = 1] \frac{m}{n} \frac{m-1}{n-1} \frac{m-2}{n-2} = \gamma_1 \frac{m}{n} \frac{m-1}{n-1} \frac{m-2}{n-2} \\ &= \gamma_1 \frac{\binom{m}{3}}{\binom{n}{3}} \end{aligned}$$

The above result is true since we are assuming that m is chosen according to Lemma 1. From this, it follows that:

$$E[F^{(1)}] = E\left[\sum_{j=1}^{\binom{n}{3}} Y_{i,1}\right] = \sum_{j=1}^{\binom{n}{3}} E[Y_{i,1}] = \binom{n}{3} \gamma_1 \frac{\binom{m}{3}}{\binom{n}{3}} = \gamma_1 \binom{m}{3}$$

TABLE 4.3: Expected and actual values of the number of combinations and labels.

Simulation	Number of labels		Number of combinations		Secrets
	Actual	Theoretical	Actual	Theoretical	
1	2815.20	2843.80	84457.00	85315.00	3245.00
2	2101.30	2047.80	63040.00	61434.00	2646.00
3	1156.20	1076.50	34687.00	32295.00	2705.30
4	1073.40	1028.90	32201.00	30867.00	2279.00
5	1327.20	1273.70	39816.00	38210.00	2845.70
6	2466.20	2525.00	73985.00	75751.00	2801.00
7	1885.20	1893.10	56555.00	56794.00	3241.70
8	1892.10	1934.70	56764.00	58042.00	2900.70
9	917.57	930.42	27527.00	27913.00	2147.70
10	2539.20	2534.80	76175.00	76044.00	3483.70
Average	1817.36	1808.87	54520.70	54266.50	2829.58

Which is the same as the result obtained above. Now, since each combination contains 3 labels and we assume all the labels to be uniformly distributed over the combinations, we get that:

$$E[L^{(1)}] = \frac{3}{m} \gamma_1 \binom{m}{3} = \gamma_1 \binom{m-1}{2}$$

We now attempt to find the expected number of Γ 's in C such that $P_i \in \text{ch}(\Gamma)$, when $i > 1$. We have:

$$\begin{aligned} E\left[\sum_{j=1}^{\binom{n}{3}} Y_{1,j} Y_{i,j}\right] &= \sum_{j=1}^{\binom{n}{3}} E[Y_{1,j}] E[Y_{i,j}] \\ &= \binom{n}{3} \gamma_1 \gamma_i \frac{\binom{m}{3}^2}{\binom{n}{3}^2} = \gamma_1 \gamma_i \frac{\binom{m}{3}^2}{\binom{n}{3}} \end{aligned}$$

So, after r challenge-response pairs, we have that:

$$\begin{aligned} E[F^{(r)}] &= \gamma_1 \binom{m}{3} + \gamma_1 \gamma_2 \frac{\binom{m}{3}^2}{\binom{n}{3}} + \cdots + \gamma_1 \gamma_r \frac{\binom{m}{3}^2}{\binom{n}{3}} \\ &= \gamma_1 \binom{m}{3} \left(1 + \frac{\binom{m}{3}}{\binom{n}{3}} \sum_{i=2}^r \gamma_i\right) \end{aligned}$$

And:

$$E[L^{(r)}] = \frac{3}{m} E[F^{(r)}] = \frac{3}{m} \gamma_1 \binom{m}{3} \left(1 + \frac{\binom{m}{3}}{\binom{n}{3}} \sum_{i=2}^r \gamma_i\right)$$

Thus, if we run the attack on a set of r challenge-response pairs, we would expect the number of times each combination and each label to appear according to the above

equations. We ran a simulation to compare the theoretical expected values against actual mean values. The simulation was run with the following parameters: $n = 112$, $m = 90$, $k = 5$ and $r = 21$. Table 4.3 shows the results. As can be seen, the theoretical values match well with the experimental results. For each challenge-response pair i , let S_i denote the set of 3 secret labels used by the user to form a convex hull. The last column in the table shows the average of the frequency of occurrence of each label in S_1 (corresponding to the Test Set), after 21 challenge-response pairs. Notice that this value is always higher than the average for all labels. By the pigeonhole principle, this means that at least one of the secret labels occurs with a frequency higher than the expected frequency for all labels. This is the motivation behind Attack 2. Since, at least one of the secret labels appears with a frequency higher than the average, there is a non-trivial chance that it will occur with the highest frequency as the output of Attack 2. We give the simulation results for Attack 2 next, following which we attempt to explain why at least one of the secret labels occurs with an above-average frequency.

4.4.2 Simulation Results for Attack 2

The simulation results for Attack 2 are shown in Table 4.4. The column labeled “pairs” shows the number of challenge-response pairs used. The column labeled “Secret Appeared” shows the number of times one of the secret labels is the output of Attack 2, in 100 runs. Thus, this corresponds to the probability of success of Attack 2. As can be seen, with a non-trivial probability at least one of the secret labels appears with the highest frequency, i.e., the output of Attack 2. The value of τ , or the threshold, is chosen such that the size of $C^{(\tau)}$ is at least 50. There is no particular reason for this choice of τ , except to ensure that the size of $C^{(\tau)}$ is reasonably large. As τ is dynamic over different runs, only its average value is shown.

In all the simulation runs the bounding rectangle had end coordinates:

$$(0, 0), (13, 0), (0, 13), (13, 13)$$

We used Turk’s method to compute a random point within a triangle [37] (See Appendix A.1 for Turk’s algorithm). Our simulation results suggest that increasing k makes the probability of success lower. However, the probability is still higher than k/n , the success probability of random guess, which is 0.0446 when $n = 112$ and $k = 5$, and 0.075 when $n = 160$ and $k = 12$. It should be noted that increasing k does not increase the time and space complexity of Attack 2, which is always $\mathcal{O}(\binom{m}{3})$, although it does affect the probability of success. Our experimental results also indicate that the success probability increases with more challenge-response pairs. Thus, the probability of success of Attack 2 is a function of n , m , k and r .

4.4.3 Why does Attack 2 Work

In this section, we give a qualitative explanation for the success of Attack 2. That is, we explain why one of the secret label appearing in C has the highest frequency with high probability. We show this in two steps. First, we show that relative to any point

TABLE 4.4: Output of Attack 2.

Simulation Number	n	m	k	pairs	Secret Appeared	Average Threshold
1	112	90	5	20	64/100 = 0.64	6.4
2				30	76/100 = 0.76	7.8
3				50	88/100 = 0.88	10.9
4	160	100	12	20	35/100 = 0.35	4.8
5				30	40/100 = 0.40	5.6
6				50	48/100 = 0.48	7.2

P clicked by the user, there are regions in the rectangle where labels of lattice points have low and high frequencies. Secondly, we reason that the secret labels have a higher probability of being in the high frequency region as compared to non-secret labels.

We assume that the rectangle has coordinates $(0, 0)$, $(a, 0)$, $(0, b)$ and (a, b) for some positive integers a and b . For simplicity, we assume that $(a + 1)(b + 1) = m$. That is, the number of possible lattice points that can be contained in the rectangle is exactly m . Assume that we are given a response point $P \in \mathbb{R}^2$. Let $C = \{\Gamma_i | P \in \text{ch}(\Gamma_i), 1 \leq i \leq \binom{m}{3}\}$. Also, for a label l , define:

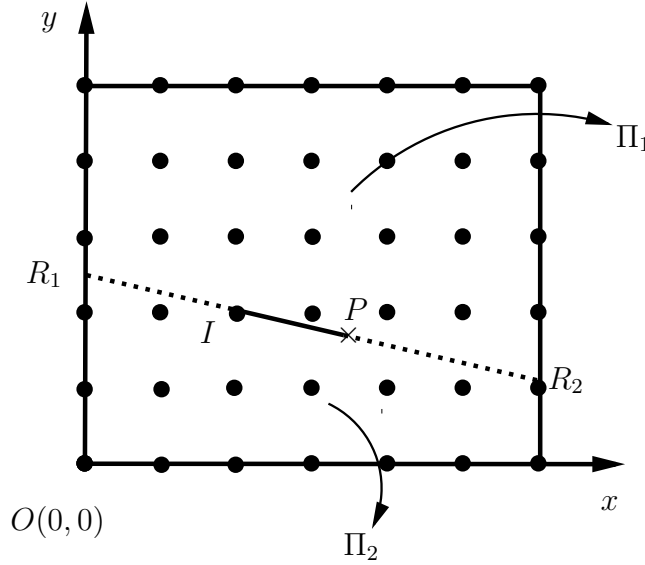
$$\text{freq}(l) = \sum_{\Gamma \in C | l \in \Gamma} \text{freq}(\Gamma)$$

For a lattice point I in the rectangle, let $\text{lab}(I)$ denote the label of I . Of interest is the question that which region of the rectangle, relative to P , contains the lattice points whose labels have higher values of $\text{freq}(\cdot)$.

Abusing notation, we shall denote $\text{freq}(\text{lab}(I))$ by $\text{freq}(I)$, when considering a generic label. We can see that $\text{freq}(I) \leq \binom{m-1}{2}$. And it is not hard to see that if $I = P$, then $\text{freq}(I) = \binom{m-1}{2}$. We now consider the case when $P \neq I$. Consider the line segment \overline{IP} . Extend this line segment in both directions such that it intersects the boundaries of the rectangle at points R_1 and R_2 as shown in Figure 4.4. If any lattice point lies on the line segment $\overline{PR_2}$, then all its 3-combinations with I will be in C . We next consider the case when no lattice point except I , lies on the line $\overline{R_1R_2}$. $\overline{R_1R_2}$ thus divides the rectangle into 2 partitions. Denote the set of lattice points in these two partitions by Π_1 and Π_2 . Thus, $|\Pi_1| + |\Pi_2| = m - 1$. Now consider two lattice points I_1, I_2 both different from I . A necessary condition for the triangle ΔII_1I_2 to contain the point P , is for I_1 and I_2 to be in different partitions.³ Thus, in this case, $\text{freq}(I) \leq |\Pi_1||\Pi_2|$. We wish to find when this product produces the maximum value. We know that $|\Pi_1| = m - 1 - |\Pi_2|$. Thus:

$$|\Pi_1||\Pi_2| = (m - 1 - |\Pi_2|)|\Pi_2|$$

³For suppose that is not the case and both I_1 and I_2 are in Π_1 , then all the sides of the triangle ΔII_1I_2 never intersect the line segment \overline{IP} , except at point I . Hence, the point P cannot be contained in the triangle unless $I = P$. But we have already assumed that not to be true.

FIGURE 4.4: The 2 partitions Π_1 and Π_2 .

Differentiating the right hand side with respect to $|\Pi_2|$ and equating it to 0, we see that the above product has a maximum value when $|\Pi_2| = (m - 1)/2$. This implies that $|\Pi_1| = (m - 1)/2$. Thus, the product above will be maximised if the 2 partitions are *equal*.

However, not all the pairs in $\Pi_1 \times \Pi_2$ will form a convex hull with I containing P . Consider the points $I_1 \in \Pi_1$ and $I_2 \in \Pi_2$. The triangle $\Delta I I_1 I_2$ will contain the point P , if the side $\overline{I_1 I_2}$ intersects the segment $\overline{P R_2}$. Similarly, if $\overline{I_1 I_2}$ intersects $\overline{R_1 I}$ or $\overline{I P}$ (except at the point P), then the corresponding triangle with I does not contain the point P . Thus, the longer the segment $\overline{P R_2}$, the higher will be the number of pairs from $\Pi_1 \times \Pi_2$ intersecting it. This gives us the following result:

freq(I) will be maximised if: (1) the line $\overline{R_1 R_2}$ divides the rectangle into two partitions with an almost equal number of lattice points, (2) the length of the line segment $\overline{P R_2}$ is close to the length of $\overline{R_1 R_2}$.

These two observations give us the following informal result:

Result 1. *Let $P \in \mathbb{R}^2$. Draw a line $\overline{R_1 R_2}$ that intersects P and divides the rectangular lattice area into 2 partitions such that the two contain an almost equal number of lattice points. Suppose $\overline{R_1 P}$ is shorter than $\overline{R_2 P}$. Then the labels of the lattice points around the vicinity of $\overline{R_1 P}$ will have higher values of $\text{freq}(\cdot)$. Furthermore, the labels of the lattice points around the vicinity of $\overline{R_2 P}$ will have lower values of $\text{freq}(\cdot)$.*

The terms “almost” and “vicinity” used in the above result hold their natural meanings and we do not attempt to rigorously define them. We call the region around

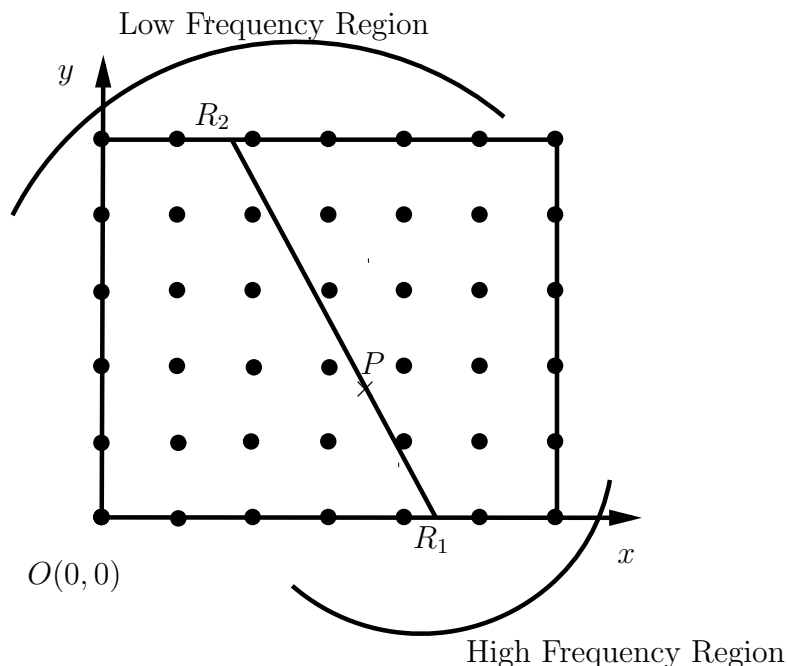


FIGURE 4.5: The high and low frequency regions.

the shorter line segment, $\overline{R_1P}$, the high frequency region, and the region around the longer line segment $\overline{PR_2}$, the low frequency region. Sandwiched between these two will be the region containing the lattice points whose labels have mid-range values of $\text{freq}(\cdot)$. Figure 4.5 illustrates this result. We insist that the boundaries of these regions are fuzzy. This analysis is correct except for some degenerate cases; such as when P is at the center of the rectangle. This is not covered by the above result, because $\overline{R_1P} < \overline{R_2P}$ is a necessary condition, which does not hold if P is at the center of the rectangle. In this case, there is an infinite number of ways to partition the rectangle into equal areas. However, apart from these exceptions, we expect the behavior to be similar most of the time. The following theorem proves that given any point P , not at the center of the rectangle, there is always a unique way to partition the rectangle into two equal areas by a line through P and the center of the rectangle.

Theorem 1. *Let R be a rectangle in the xy -plane of real numbers, with vertices $(0,0)$, $(a,0)$, $(0,b)$ and (a,b) . Let $C(a/2, b/2)$ be the center of the rectangle R . Let $P \in \mathbb{R}^2$ be a point contained in this rectangle with coordinates (x_P, y_P) . Suppose $P \neq C$. Then, the line \overline{PC} is the unique line that divides R into two polygons of equal areas.*

Proof. Any line through C divides the rectangle into equal areas. Therefore, given a point P , the line \overline{PC} will divide the rectangle into equal areas. Now, suppose there is another line L , that goes through P and not through C and divides the rectangle into two polygons of equal areas. Consider the line L' that is parallel to L and goes through point C . Thus L' also divides the rectangle into two polygons of equal areas.

However, L' is different from L since it does not go through P . But this implies that one of the two polygons resulting from L is contained in one of the polygons resulting from L' . This means that the area of the polygon is smaller than half. A contradiction. Therefore, the line \overline{PC} is the unique line dividing a rectangle into equal areas.

The equation of the line \overline{PC} is given by:

$$\begin{aligned}
 \frac{y - y_P}{x - x_P} &= \frac{b/2 - y_P}{a/2 - x_P} \\
 \Rightarrow (a/2 - x_P)(y - y_P) &= (x - x_P)(b/2 - y_P) \\
 \Rightarrow (a - 2x_P)(y - y_P) &= (x - x_P)(b - 2y_P) \\
 \Rightarrow ay - ay_P - 2x_P y + 2x_P y_P &= bx - 2x_P y_P - bx_P + 2x_P y_P \\
 \Rightarrow (a - 2x_P)y &= (b - 2y_P)x + ay_P - bx_P
 \end{aligned}$$

which holds if $x_P \neq a/2$. If $x_P = a/2$, then the equation of the line \overline{PC} is $y = a/2$. \square

Theorem 1 allows us to construct the line partitioning the rectangle into two equal parts through the point P . Figure 4.6 shows the line through the point P during a simulation run. The parameters used were $a = b = 9$, $n = 125$, $m = 100$ and $k = 5$. The triangle shown is the convex hull of the 3 secret labels chosen at random by the user (simulated). As the figure illustrates, the lattice points with the highest values of $\text{freq}(\cdot)$ are populated around the shorter line segment $\overline{R_1 P}$ and the lattice points with the lowest values of $\text{freq}(\cdot)$ are populated around the longer line segment $\overline{R_2 P}$. Incidentally, the figure also shows one of the secret labels (label not shown) appearing in the high frequency region. The white lattice point in the figure represents the secret label appearing in the high frequency region. When $m < (a+1)(b+1)$, there are some “holes” in the rectangle, however the above results still give us a good approximation.

Equipped with this knowledge, we can finally give a reason for the success of Attack 2. Let $S \in C$ be the 3-combination selected by the user to form the convex hull containing P . Let the 3 secret labels in S be l_1, l_2 and l_3 . Since $\text{ch}(S)$ is a triangle, at least one of l_1, l_2 and l_3 will be in the high frequency region with high probability. To see why this is true, we see that the only way for this not to be true is for one element of S to be in the low frequency region and the other two in the average frequency region. But compared to all possible triangles, the number of such triangles is small. Thus with high probability, at least one of the secrets will be in the high frequency region. Suppose that point is l_1 . This means that $\text{freq}(l_1)$ will have a high value relative to most labels. This in turn means that l_1 will be in a high number of 3-combinations in C . Since, the number of such 3-combinations is high, given r challenge-response pairs, these 3-combinations will have a higher value of $\text{freq}(\cdot)$ with high probability, which implies that the frequency of l_1 will be high. It should be noted that l_1 may not appear in some challenges or it could be in the low frequency region in some challenges (because one of the other two secret labels is in the high frequency region). However, on average, a secret label always has a higher probability to be in the high frequency region than a non-secret label. This makes the most frequent label a secret label with high probability. This explains why Attack 2 is successful with high probability.

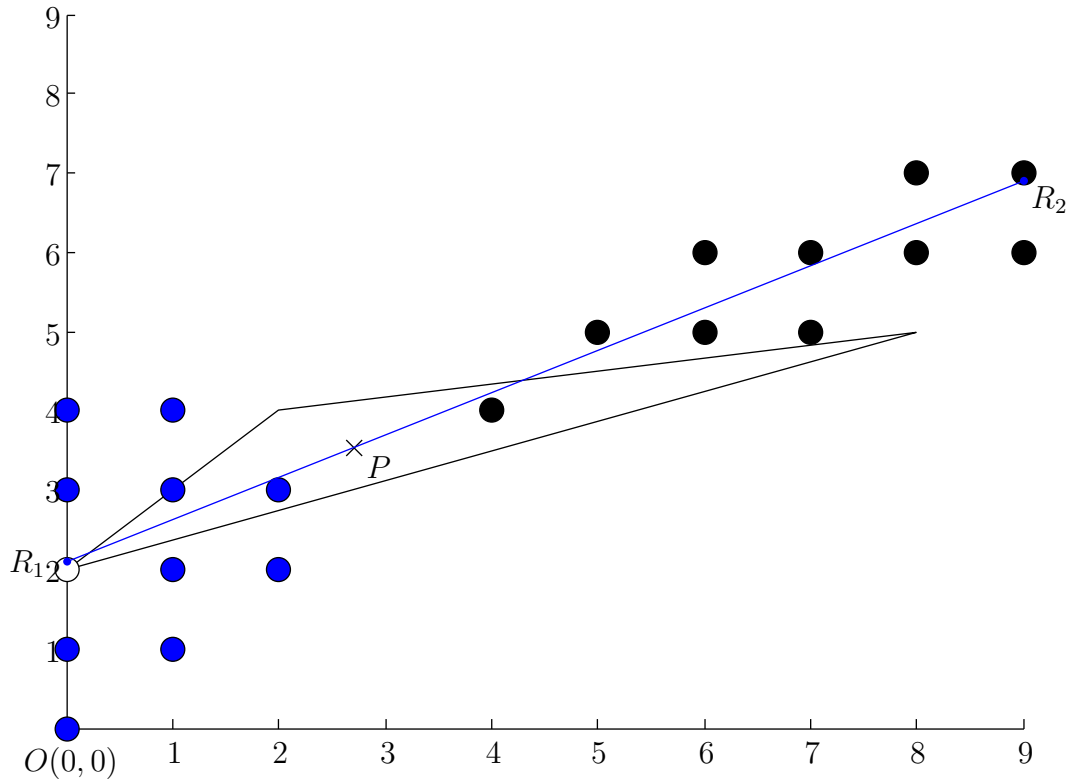


FIGURE 4.6: A simulation run showing the low and high frequency regions.

Improved Variant of Attack 2.

The above analysis gives us an interesting way to improve Attack 2. Given r challenge-response pairs, we choose the pair as the Test Set, which has P closest to the boundary of the rectangle. This will mean that with high probability, one of the secret labels will be near the edge and hence will have a high value of $\text{freq}(\cdot)$. Our test results show that indeed this increases the success probability of the attack. The results are shown in Table 4.5. See in contrast the results obtained in Table 4.4. We call this variant of Attack 2, the Chosen Test Set Attack.

4.4.4 Impersonation using Attack 2

Attack 2 (and its variant), outputs one of the secret labels with a non-negligible probability, say $p(n, m, k, r)$ or $p(m, k, r)$, since n is dependent on m and k . We abbreviate this probability as p . One can run the attack multiple times to obtain the whole set of secrets. But that requires in the order of kr challenge-response pairs. While this number is not huge, the adversary can still impersonate \mathcal{H} with high probability even after observing fewer challenge-response pairs. We see that \mathcal{A} does not need to find all the k secrets in order to impersonate \mathcal{H} . The impersonation process is described below.

TABLE 4.5: Output of the Chosen Test Set Attack.

Simulation Number	n	m	k	pairs	Secret Appeared	Sessions
1	112	90	5	20	77/100 = 0.77	10
2				30	83/100 = 0.83	14
3				50	95/100 = 0.95	20
4	160	100	12	20	50/100 = 0.50	77
5				30	67/100 = 0.67	86
6				50	78/100 = 0.78	123
7	320	200	12	20	46/100 = 0.46	83
8				30	46/100 = 0.46	125
9				50	59/100 = 0.59	163
10	357	200	25	20	35/100 = 0.35	330

Algorithm: Impersonate \mathcal{H} .

Input: t sets of r challenge-response pairs.

Output: 1 if successful, 0 if unsuccessful.

- 1: *Obtain Secrets.* Run Chosen Test Set Attack on each set of r challenge-response pairs, to obtain the set of labels $L = \{l_1, l_2, \dots, l_t\}$.
- 2: *Impersonate \mathcal{H} .* Initiate an identification session with \mathcal{C} .
- 3: **for** each of the r_0 challenges sent by \mathcal{C} **do**
- 4: **if** only one label from L is in the challenge **then**
- 5: Click on the lattice point of that label.
- 6: **else if** two labels from L are in the challenge **then**
- 7: Randomly click any point on the line connecting the two corresponding lattice points.
- 8: **else if** three or more labels from L are in the challenge **then**
- 9: Click a random point contained in the convex hull of the corresponding lattice points.
- 10: **else if** no label from L is in the challenge **then**
- 11: Click a random point contained in the rectangle.
- 12: **if** \mathcal{C} outputs **accept**, then output 1, else output 0.

The probability that “Impersonate \mathcal{H} ” outputs 1, depends in part on the success probability of the Chosen Test Set Attack. This also suggests that once $k - 2$ secret labels are obtained, they are enough to impersonate \mathcal{H} with probability 1. This is true since every challenge will contain at least one of the $k - 2$ secret labels, and then the above impersonation process can be used to impersonate \mathcal{H} .⁴ Thus, if $k = 5$, only 3

⁴There is a small probability that the attacker may fail, due to an inaccurate click. A human user cannot always exactly click the center of an icon or on a line, which may render the clicked point out of the convex hull.

secret labels are required, and if $k = 12$, only 10 secret labels are enough. Thus the effective security of the protocol is $k - 2$ secret labels.

Even if the number of secret labels obtained is less than $k - 2$, impersonation can still be successful with high probability. For instance, the probability that the t labels in L are all distinct secret labels and the adversary is successful in impersonating \mathcal{H} is:

$$\begin{aligned}
& \frac{k-1}{k} \frac{k-2}{k} \cdots \frac{k-t+1}{k} \left(\Pr[|L| = 0] \cdot \frac{1}{2} + (1 - \Pr[|L| = 0]) \cdot 1 \right)^{r_0} p^t \\
&= \frac{p^t}{k^{t-1}} (k-1) \cdots (k-t+1) \left(1 - \frac{1}{2} \Pr[|L| = 0] \right)^{r_0} \\
&= \frac{p^t}{k^{t-1}} (k-1) \cdots (k-t+1) \left(1 - \frac{1}{2} (1 - (m/n))^t \right)^{r_0} \\
&= \left(\frac{p}{k} \right)^t \frac{k!}{(k-t)!} \left(1 - \frac{1}{2} (1 - (m/n))^t \right)^{r_0} \tag{4.3}
\end{aligned}$$

when $t < k - 2$ and:

$$\left(\frac{p}{k} \right)^t \frac{k!}{(k-t)!}$$

when $t \geq k - 2$. Here, we have assumed that the probability of success of a random click is $1/2$. In actual, it can be considerably less than $1/2$. But that does not result in any substantial change in the overall success probability of impersonation. Let us consider the parameter values $k = 5$, $n = 112$ and $m = 90$, and assume that $r_0 = 10$ and $r = 30$. From Table 4.5, we get the approximate probability of success of the Chosen Test Set Attack as $p = 0.95$. For these values, the above probability has the peak value of 0.59 at $t = 2$. This implies that even after observing only $tr/r_0 = 6$ identification sessions, the adversary has a 60 percent chance of getting $t = 2$ distinct secret labels and successfully impersonating \mathcal{H} . For $k = 12$, $n = 160$, $m = 100$ and $r = 50$, we get the approximate value of $p = 0.78$ from Table 4.5. These values give a peak probability value of 0.27 at $t = 3$. This means only 15 observed identification sessions. Similarly, for $n = 320$, $m = 200$, $k = 12$ and $r = 50$, we get the peak probability 0.15 at $t = 2$. These probabilities are non-negligible. Notice that the probability of success through random clicks is less than $(1/2)^{10} \approx 0.00098$.

The output of t trials of Attack 2 can be modeled as following a binomial distribution, where the probability that a secret label is the output is p . Since each secret label is equally likely to be the output, we can assume that the probability of each one being the output is p/k . Let X denote the number of trials required before $k - 2$ distinct labels are obtained, given that each trial is a success. Then [38, §7.2, p. 334]:

$$E[X] = 1 + \frac{k}{k-1} + \frac{k}{k-2} + \cdots + \frac{k}{3}$$

And since the trials are distributed binomially, we get:

$$\begin{aligned}
tp &= 1 + \frac{k}{k-1} + \frac{k}{k-2} + \cdots + \frac{k}{3} \\
\Rightarrow t &= \frac{1}{p} \left(1 + \frac{k}{k-1} + \frac{k}{k-2} + \cdots + \frac{k}{3} \right)
\end{aligned}$$

Thus, the expression above gives the expected number of trials of Attack 2 required to get $k - 2$ distinct labels. Each trial takes r challenge-response pairs and there are r_0 challenge-response pairs in each identification session. Thus, under Attack 2, the protocol can only be used for rt/r_0 sessions before the adversary has $k - 2$ secret labels to impersonate \mathcal{H} with probability 1. The last column in Table 4.5, under the heading “Sessions”, shows the values of rt/r_0 for the corresponding parameters, where t is obtained from the expression above. These values should be seen with caution, as they only give a rough estimate of the number of sessions a particular secret can be used under the attacks mentioned in this paper. We again stress that the adversary can still impersonate \mathcal{H} with a non-negligible probability even with fewer sessions.

The weakness exploited in Attack 2 seems to be an inherent problem for convex hull based protocols. Even if the user is asked to form a convex hull of more than 3 secret icons, the attack can still be applied. This is true since the point clicked by the user will be contained in at least one of the possible 3-combinations of the secret icons.

4.4.5 Discussion

A shortcoming of the analysis in this chapter is the lack of an explicit expression for $p(m, k, r)$, i.e., the success probability of Attack 2 (or its variant). Unfortunately, there does not seem to be a straightforward way of obtaining such an expression. Still, the numerical values of $p(m, k, r)$ obtained indicate that the protocol is insecure even for system parameter values recommended by Wiedenbeck et al. for high security, such as $n = 320$, $m = 200$ and $k = 12$. For example, there is approximately a 15 percent chance that the adversary can impersonate a user after observing only 10 identification sessions with these parameter values. By observing more sessions, the probability can be improved. It is not clear whether Attack 2 can be modified to obtain secrets with a number of challenge-response pairs less than kr . So far, we only know how to obtain the complete set of secret icons by a sequential application of Attack 2.

To mitigate impersonation using Attack 2, we can increase r_0 . For instance, for the aforementioned values and $r_0 = 20$, the peak success probability of impersonation is 0.09. However, increasing r_0 increases the number of challenge-response pairs per session. This in turn implies that the number of identification sessions observed, before the adversary can obtain the secrets, decreases. The usability of the system also decreases with increasing r_0 . Another way is to increase k . This does not necessarily imply an increase in identification time, since the user has to form a convex hull of only 3 secret labels. The last row of Table 4.5 shows the probability of success of the Chosen Test Set attack, when $k = 25$ and $m = 200$. While, the success probability is still non-negligible, with these parameter values, Equation (4.3) indicates that the user can be authenticated for about 330 sessions before secret renewal based on the attacks mentioned in this paper. However, increasing k raises some usability issues. First, remembering 25 graphical icons might not be easy for humans. Secondly, with $k = 25$, an averagely larger number of secret icons are to be displayed on the screen. This means that the convex hull of these icons can occupy a large area of the screen. This makes the probability of success of the random click attack higher.

4.5 Conclusion

The Convex Hull Click (CHC) graphical human identification protocol is an interesting alternative to other proposed protocols in literature. The scheme is easy to execute for humans and is apparently more secure as compared to some of the previous approaches. The security of the underlying problem was not extensively analyzed previously. This is partly due to the complex structure of the problem. This work is the first attempt to extensively analyze the protocol. We have shown two attacks on the CHC protocol. The first attack outputs the secret icons with high probability after observing a few authentication sessions. We have proposed a formula which allows to find values of system parameters for which this attack can be avoided. The second attack outputs a secret icon with high probability after observing only a handful of identification sessions. The attack can be improved and then can be used to impersonate the user with a non-trivial probability.

Our approach in this chapter has been as mathematically rigorous as possible. However, the problem is not easy to tackle analytically and computer simulations were needed to supplement the theoretical work. While in its current form, the protocol does seem to have significant weaknesses, research can be done to find some variants of the protocol that are easy for humans to compute while being secure at the same time. An interesting future line of research is to find new geometric problems for human identification protocols. Some other existing examples of human identification protocols based on geometric problems appear in [21, 39], but the exact security of these remains unexplored. Interestingly, the aim of the CHC protocol is shoulder-surfing resistance [9]. However, as we have seen, from an analytical point of view we can consider this as a human identification protocol according to the definition stated in Chapter 2. An automated shoulder-surfer can gain the same knowledge as a passive adversary in Matsumoto and Imai's threat model. Thus both the analysis of CHC protocol in this chapter and the Predicate-based Authentication Service (PAS) in the previous chapter, justify our classification of these authentication methods under the general definition of human identification protocols.

We have seen that both PAS and CHC are insecure in the sense that the secret can be found after observing a number of sessions less than originally claimed. In the case of PAS, this was probably down to the fact that the underlying problem was not rigorously defined, and as a result we found many security flaws. On the other hand, the underlying mathematical structure was obvious in CHC. All we needed to do is to analyze the properties and geometry of convex hulls to see if the protocol can be used for a large number of authentication sessions. However, unfortunately, due to the very structure of convex hull of three points, information about the secret is revealed after a handful of authentication sessions. Thus, even though the approach used in the latter protocol is better, we have not improved much on the number of sessions a protocol can be used before secret renewal. Next, we show our first protocol construction. We conjecture that the protocol can be used for a large number of authentication sessions. We back this claim with a detailed security analysis.

5

Protocol Construction 1: Kangaroo Hopping

In this chapter we propose a new human identification protocol which we call the Kangaroo Hopping Protocol. The protocol is designed so that the underlying problem cannot be straightforwardly described as a system of linear equations, as in the case of the Example Protocol from Chapter 2 or Matsumoto’s protocols from [19]. The name of the protocol alludes to the fact that the prover \mathcal{H} essentially jumps over locations in a challenge based on random integers. This name is inspired by Pollard’s kangaroo algorithm for solving discrete logarithms [40], although there is no other similarity between his algorithm and our protocol.

We also propose a new generic passive attack on human identification protocols. The attack is a variant of the meet-in-the-middle (time-memory tradeoff) attack described by Hopper and Blum in [8, §6], briefly discussed in Chapter 2. The main component of the attack is Coppersmith’s baby-step giant-step algorithm which has its application in solving the restricted Hamming weight discrete logarithm problem [41]. It performs better than the attack mentioned in [8] on the two proposed protocols in that paper, namely HB and Sum of k Mins, further reducing their security. Our protocol shows better security under both time-memory tradeoff attacks. We also rigorously analyze other possible attacks to demonstrate their efficacy, or lack thereof, in breaking our protocol.

The organization of this chapter is as follows. Section 5.1 describes the Kangaroo Hopping Protocol together with a description of a user friendly implementation. Section 5.2 contains the bulk of the results described in this chapter, and corresponds to the security analysis of the protocol. It also describes the new attack on human identification protocols based on Coppersmith’s baby-step giant-step algorithm and Hopper and Blum’s meet-in-the-middle algorithm. A brief usability analysis is described in Section 5.3, and concluding remarks are present in Section 5.4.

5.1 Proposed Protocol

We shall refer to a vector of n elements or an n -tuple as an ordered list of n elements. If \mathbf{c} is a vector of n -elements, then $\mathbf{c}[i]$ denotes the i th element of \mathbf{c} , for $0 \leq i \leq n-1$;¹ the index i is called the i th location in \mathbf{c} . Let n be such that $n = ab$ for positive integers a and b . We call a the *jump constant* for reasons that will be clear later. Let k and $d \geq 2$ be positive integers. Let \mathbf{c} be a vector of n integers drawn uniformly at random from the set $\{0, 1, \dots, d-1\}$.

We first describe the protocol formally and then show an implementation that is human friendly. Sections 5.1.1 and 5.1.2 give a less technical description of the protocol with example values for the parameters.

Protocol: Kangaroo Hopping.

Setup: Let n , a , b , μ , k and d be public parameters, with $n = ab$. \mathcal{C} and \mathcal{H} choose $k+1$ locations in \mathbf{c} . These locations are essentially a set of integers: s_0, s_1, \dots, s_k , where $0 \leq s_i \leq n-1$. s_0 is called the starting location. All these locations constitute the secret.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} generates the vector \mathbf{c} and sends it to \mathcal{H} .
- 3: \mathcal{H} assigns $t \leftarrow s_0$.
- 4: \mathcal{H} updates $t \leftarrow (t + a \cdot \mathbf{c}[s_0]) \bmod n$.
- 5: **for** $1 \leq j \leq k$ **do**
- 6: \mathcal{H} updates $t \leftarrow (t + \mathbf{c}[s_j]) \bmod n$.
- 7: \mathcal{H} sends $\mathbf{c}[t]$ to \mathcal{C} .
- 8: \mathcal{C} outputs **accept** if all the answers are correct, otherwise \mathcal{C} outputs **reject**.

Lemma 2. Let S_1 and S_2 be two sets of secret locations in \mathbf{c} . Then, Protocol 1 is an identification protocol with:

$$\Pr[\langle \mathcal{H}(S_1), \mathcal{C}(S_2) \rangle = \text{accept}] \leq d^{-\mu}$$

if $S_1 \neq S_2$ and 1 otherwise.

Proof. Since the protocol is deterministic, if $S_1 = S_2$, \mathcal{C} will accept \mathcal{H} with probability 1. For $S_1 \neq S_2$, we see that since each element of \mathbf{c} is generated uniformly at random from the integers $\{0, 1, \dots, d-1\}$, the probability that two different sets of locations generate the same output for \mathbf{c} is at most $1/d$. The result follows for μ iterations. \square

5.1.1 User Friendly Implementations

Both graphical and textual implementations are possible for our protocol. In a graphical implementation, we can use n graphical objects such as software icons just as in

¹There is a slight difference in notation in this chapter. Instead of defining the index from 1 to n , we define it from 0 to $n-1$, where index 0 is the first index and index $n-1$ is the n th index. This shall prove convenient when analyzing the security of the protocol.

the case of the Example Protocol's implementation described in Chapter 2. Clearly, in this case the user's secret is a set of k icons out of n . Since a graphical implementation was already described in Chapter 2, here we illustrate an example text-based implementation.

We represent the secret space, i.e. the locations in \mathbf{c} by an alphabet. For instance, the English alphabet is a candidate. The vector \mathbf{c} is presented to \mathcal{H} in the form of a grid with $a \times b$ cells, where $b = n/a$ (a is chosen such that it divides n). Each location in \mathbf{c} is mapped to a unique character in the secret space alphabet. Each cell in the grid contains a unique character from the secret space, below which is the corresponding random digit from \mathbf{c} . The secret is then a string from this alphabet, instead of a set of integers of vector locations. Thus, the starting location is also a character from this alphabet.

Given a challenge "grid", \mathcal{H} locates the cell containing the starting character. This corresponds to the location s_0 in the formal description. \mathcal{H} then looks at the digit corresponding to this cell. Let the digit be d_0 . \mathcal{H} moves d_0 steps vertically downwards, with wraparound if necessary, thus reaching a new character location in the same grid. Call this location l_0 . \mathcal{H} then looks at the digit in the cell containing s_1 . Let d_1 be the digit. It now moves horizontally to the right of the location l_0 and moving to the start of the next row if the end of the row is reached. This results in the new location l_1 . \mathcal{H} continues to move horizontally according to the digits corresponding to the rest of its secret locations. If the bottom right corner of the grid is reached, \mathcal{H} moves to the top left corner, thus moving in a cycle. At the end of this procedure, \mathcal{H} simply outputs the digit corresponding to the final character location thus reached. Figure 5.1 shows an example. Here $a = 12$ and $b = 6$, which implies that $n = 72$. In this example and also through most of the chapter, we will choose $d = 10$, as this is a common base for humans. The alphabet is composed of the characters $a, \dots, z, A, \dots, Z, 0, \dots, 9$ and special characters: $!, @, \#, \$, \%, \wedge, \&, *, (,)$. Notice that the order of the characters remains the same for all challenges and only the digits corresponding to these characters change for different challenges. For the graphical implementation, we simply replace the alphabet with a corresponding set of graphical objects. While the text-based implementation is shown here as an illustration of our protocol, as before, we recommend graphical implementation as it is more user-friendly and has less security issues, such as resistance to dictionary attacks.

5.1.2 Different Ways of Computation

The above mentioned procedure is one way the human user can perform the protocol steps. There are a number of other ways in which the protocol can be executed. The user can choose any method he or she prefers.

For instance, a different and probably more efficient way is sketched here.

1. Ignore the starting location and add all the digits corresponding to the remaining k secret locations.
2. From the starting location, move d_0 steps vertically downwards (continuing from

a	b	c	d	e	f	g	h	i	j	k	l
3	2	6	9	2	1	7	5	4	4	6	8
m	n	o	p	q	r	s	t	u	v	w	x
1	6	7	4	9	7	5	3	2	7	6	1
y	z	A	B	C	D	E	F	G	H	I	J
3	2	5	1	5	2	9	6	6	8	6	0
K	L	M	N	O	P	Q	R	S	T	U	V
3	1	7	4	9	7	5	3	4	7	6	1
W	X	Y	Z	0	1	2	3	4	5	6	7
6	3	8	2	6	8	3	2	9	5	8	0
8	9	!	@	#	\$	%	^	&	*	()
4	7	2	0	9	1	5	3	6	3	4	9

FIGURE 5.1: An example challenge grid.

the top if the bottom of the grid is reached), where d_0 is the digit corresponding to the starting location.

3. Divide the sum obtained in Step 1 by the jump constant a to get a quotient and a remainder. The quotient is the number of vertical steps and the remainder is the number of horizontal steps to be taken. The user can then follow these steps from the location reached in Step 2 and finally output the digit corresponding to the location thus reached.

If the jump constant a is a multiple of 10 then the above division can be performed by most humans mentally. Thus to make this method easy for most users, n and a can be chosen to be 200 and 20 respectively. These parameters are easy for humans to use. k can be chosen somewhere between 10 and 15.

5.2 Security Analysis

Recall that for security, we consider the computationally bounded passive adversary of Definition 4. The adversary can view every challenge-response pair. The adversary knows the description of the Kangaroo Hopping Protocol and the public parameters specified in that protocol. The only thing hidden from the adversary is the set of secret locations shared by \mathcal{C} and \mathcal{H} . We assume that the secret locations are chosen uniformly at random from the set of all possible secret locations.

Given m challenge-response pairs $(\mathbf{c}_1, r_1), \dots, (\mathbf{c}_m, r_m)$, the goal of \mathcal{A} is to impersonate \mathcal{H} either by partially or completely learning the secret locations or by impersonating \mathcal{H} by guessing the answers without knowledge of the secret. We assume that all these challenge-response pairs correspond to successful authentication sessions between \mathcal{H}

and \mathcal{C} . We first look at random guess and brute force attacks, following which we shall show more sophisticated attacks.

5.2.1 Some Obvious Attacks

Recalling the discussion on the security of the Example protocol in Chapter 2, it is important to know the probabilities of success of the random guesses and the time complexity of the brute force attack so that values of system parameters can be chosen accordingly.

Random Guess 1

The most obvious impersonation attack is to randomly guess the response when given a challenge. Since the output is in the range $\{0, 1, \dots, d-1\}$, a random guess from this set will be successful with probability $1/d$ for each challenge. For μ rounds (iterations), this probability is $d^{-\mu}$.

Random Guess 2

Another method for impersonation is to guess the secret and then answer a challenge by following the steps of Protocol 1 correctly. We see that there are a total of n^{k+1} possible ways of choosing $k+1$ locations in \mathbf{c} . However, due to the commutativity of the addition operation, any permutation of a given set of locations will generate the same output. Thus we are looking at the number of ways of choosing the starting location times the number of ways of choosing k locations from a set of n locations with replacement and without order, which is exactly: $n \binom{n+k-1}{k}$. Thus the probability of success in guessing the correct secret is $(n \binom{n+k-1}{k})^{-1}$.

Brute Force

The brute force attack has complexity $\mathcal{O}(n \binom{n+k-1}{k})$. It works by trying all possible secret locations satisfying m challenge-response pairs. In the next section, we shall see that after observing m challenge-response pairs, the expected number of candidates for the secret is: $n \binom{n+k-1}{k} / d^m$. To reduce this number to a unique secret, \mathcal{A} needs on average: $n \binom{n+k-1}{k} / d^m \approx 1 \Rightarrow m_{\text{lb}} \approx \log_d(n \binom{n+k-1}{k})$ challenge-response pairs (the information theoretic bound on m). Thus $\mathcal{O}(\log_d(n \binom{n+k-1}{k}))$ challenge-response pairs are enough to find the secret uniquely. With a lower number of challenge-response pairs, there are multiple candidates and even a computationally unbounded adversary cannot distinguish between them. For concrete values, we see that if $n = 200$ and $k = 15$, the brute force attack has complexity roughly 2^{83} . An attack with this complexity is generally considered intractable.

5.2.2 Algebraic Interpretation

Given m challenge-response pairs $(\mathbf{c}_1, r_1), \dots, (\mathbf{c}_m, r_m)$, we now consider the problem of finding the secret locations s_0, \dots, s_k . We attempt to describe this problem algebraically. For any challenge-response pair (\mathbf{c}, r) , a location \hat{r} satisfying $\mathbf{c}[\hat{r}] = r$ is called a *satisfying location* for (\mathbf{c}, r) . For $1 \leq i \leq m$, let \mathfrak{R}_i be the set of all satisfying locations for (\mathbf{c}_i, r_i) . Let $\mathfrak{R}^m = \mathfrak{R}_1 \times \dots \times \mathfrak{R}_m$ be the m -ary cartesian product over these m sets. We represent the elements of \mathfrak{R}^m as m -element vectors, $\hat{\mathbf{r}} = [\hat{r}_1 \ \dots \ \hat{r}_m]^T$, in an obvious way. Recall that for any vector \mathbf{x} , $\text{wt}(\mathbf{x})$ denotes its Hamming weight. Define the matrix C as:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{bmatrix}$$

where $c_{i,j} = \mathbf{c}_i[j]$. Then, for each $\hat{\mathbf{r}} \in \mathfrak{R}^m$ we have:

$$\begin{bmatrix} aC & C \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \equiv \hat{\mathbf{r}} \pmod{n} \quad (5.1)$$

where \mathbf{x} and \mathbf{y} are n -element vectors with $\text{wt}(\mathbf{x}) = 1$ and $\text{wt}(\mathbf{y}) = k$. Clearly, \mathbf{x} corresponds to the starting secret location and \mathbf{y} corresponds to the k remaining secret locations. Notice that \mathbf{y} need not be a binary vector since each location can be chosen more than once. There are $|\mathfrak{R}^m|$ such equations and we cannot write this as a system of fewer equations as for each satisfying location \hat{r} for a challenge-response pair (\mathbf{c}, r) , r is independent of \hat{r} . This is true since each element of \mathbf{c} is generated uniformly at random from $\{0, 1, \dots, d-1\}$.

We first estimate $|\mathfrak{R}^m|$.

Lemma 3. For $1 \leq i \leq m$, $E[|\mathfrak{R}_i|] = n/d$. And $E[|\mathfrak{R}^m|] = (n/d)^m$.

Proof. Let $i \in [1, m]$. In each pair (\mathbf{c}_i, r_i) , if every element of \mathbf{c}_i is generated uniformly at random from $\{0, 1, \dots, d\}$, then the expected number of times r_i occurs in \mathbf{c}_i is n/d . This is exactly the expected number of satisfying locations for r_i . The result for $|\mathfrak{R}^m|$ follows from its definition. \square

There is exactly one element in \mathfrak{R}^m that contains the satisfying locations corresponding to \mathcal{H} 's secret. We denote this by $\hat{\mathbf{r}}_s$. For more compact notation, define $\mathbf{s} = [\mathbf{x} \ \mathbf{y}]^T$ and $C' = [aC \ C]$. Finding a unique \mathbf{s} satisfying $C'\mathbf{s} \equiv \hat{\mathbf{r}}_s \pmod{n}$ thus translates into finding the secret locations of \mathcal{H} . There are $2n$ unknowns in \mathbf{s} . If m is small, the number of solutions for \mathbf{s} is very large. On the other hand, higher m means a higher value of $E[|\mathfrak{R}^m|] = (n/d)^m$, which in turn means more equations to be solved, since \mathcal{A} has no way to distinguish $\hat{\mathbf{r}}_s$ from other elements in \mathfrak{R}^m .

Let *Solve Equations* be an algorithm that finds solutions (possibly multiple) to the linear system defined in Equation (5.1). Further, let $\tau(n, k, m, d)$ be the time complexity of this algorithm. We have the following probabilistic attack for finding \mathbf{s} .

Attack: Attack 5.1.

Input: The matrix C' and the set \mathfrak{R}^m .

Output: A candidate secret.

- 1: Initialize an empty set S .
- 2: **for** each $\hat{\mathbf{r}} \in \mathfrak{R}^m$ **do**
- 3: Run Solve Equations on input $C'\mathbf{s} = \hat{\mathbf{r}} \bmod n$. If any solutions are found, assign them to the set S .
- 4: Output an element uniformly at random from S .

This attack will perform well probabilistically if $|S|$ is small. As we have found before, the expected size of \mathfrak{R}^m is $(n/d)^m$. There are a total of n^m possible different output vectors for $C'\mathbf{s} \bmod n$. Therefore, the probability that an \mathbf{s} , different from \mathcal{H} 's secret, is in the set S can be estimated as $\frac{(n/d)^m}{n^m} = 1/d^m$. Thus this probability becomes lower as m increases. Therefore, we can assume that the performance of this attack is good. The expected time complexity of Attack 5.1 is:

$$\mathcal{O}\left(\tau(n, k, m, d) \left(\frac{n}{d}\right)^m\right)$$

If Gaussian elimination is used as the Solve Equations algorithm, we require $m \geq 2n$, but this means that the expected size of \mathfrak{R}^m will be greater than or equal to $(n/d)^{2n}$. We can see the complexity of this attack with concrete values. Let $n = 100$, $d = 10$ and $k = 16$. Then, we have $m \geq 2n = 200$. Gaussian elimination takes time $\mathcal{O}(n^3)$ giving a total approximate time 2^{687} , which is surely infeasible.

Since the weights of \mathbf{x} and \mathbf{y} are restricted, we might still be able to use other methods with a smaller value of m . To this end, given m , we find the number of possible solutions of the following equation:

$$C'\mathbf{s} \equiv \hat{\mathbf{r}} \bmod n \tag{5.2}$$

where $\hat{\mathbf{r}} \in \mathfrak{R}^m$. Since $\text{wt}(\mathbf{x}) = 1$ and $\text{wt}(\mathbf{y}) = k$, with $m = 0$ there are a total of $n \binom{n+k-1}{k}$ possible choices for \mathbf{s} . With $m = 1$, we expect a $1/n$ fraction of these choices to satisfy the above equation. Continuing on this way, we see that the expected number of possible choices for \mathbf{s} are:

$$\frac{n \binom{n+k-1}{k}}{n^m} = \frac{\binom{n+k-1}{k}}{n^{m-1}}$$

Since the expected size of \mathfrak{R}^m is $(n/d)^m$, we see that the combined expected number of solutions are:

$$\frac{n \binom{n+k-1}{k}}{d^m}$$

Equating the above expression to 1, and denoting $m_{\text{lb}} = m$ we get:

$$m_{\text{lb}} = \log_d n + \log_d \binom{n+k-1}{k} \tag{5.3}$$

Thus $m_{\text{lb}} \geq \log_d n + \log_d \binom{n+k-1}{k}$ is required on average to find a unique value of \mathbf{s} in Equation (5.2). This m_{lb} is again the information theoretic bound on m to obtain a unique secret. By using the concrete values as above, we find that the resulting value of m_{lb} from Equation (5.3) is approximately 29. Thus in theory, we can have an algorithm that solves the problem with $m \geq 29$. However, this value of m implies that $(n/d)^m \approx 2^{96}$. Thus whether or not a Solve Equations algorithm that works for smaller values of m can be found, the overall time complexity of Attack 5.1 is still very high. Thus it is not possible to improve this complexity without a different approach. Informally speaking, the main reason for this interesting result is that in our protocol, the computations are also done on the location indices and not just the digits corresponding to these locations as for instance in Matsumoto's protocols from [19]. Since the digits are generated uniformly at random, the final answer is not linearly dependent on the location indices. Next, we present a time-space tradeoff algorithm that utilizes fewer challenge-response pairs and has better time complexity.

5.2.3 Time-Memory Tradeoff

In [8, §6], Hopper and Blum sketched a meet-in-the-middle algorithm on k -out-of- n protocols which has an average-case time complexity of:

$$\mathcal{O}\left(n^{k(1-\frac{\ln d}{2 \ln Q})}\right) \quad (5.4)$$

By k -out-of- n protocols, we mean those protocols that use a shared secret which is a set of k objects out of n . Here Q is an intermediate result, which in our protocol corresponds to the range of the intermediate locations during the computation of the protocol. Thus in our protocol, $Q = n$. The Kangaroo Hopping Protocol and the protocols from [8] as well as many other protocols in literature loosely fall in the category of k -out-of- n protocols (the only difference in our protocol is that we have a starting location that is computed differently from the k remaining locations). For the HB protocol, the average-case time complexity of this attack is: $\mathcal{O}(\binom{n}{k/2})$ [8, §3.1, pp. 58]. This is true due to two reasons. First the protocol uses the addition operation, which is commutative, and the user has to choose k *unique* locations as a secret. Therefore, the number of possible secrets are $\binom{n}{k}$ instead of n^k . Secondly, Q equals d in their protocol. Similarly, for the Sum of k Mins Protocol the average-case time complexity of this attack is $\mathcal{O}(\binom{n(n-1)/2}{k/2})$ [8, §3.2, pp. 59]. However, since the size of the secret is exactly twice than in the HB protocol, the comparative time complexity is $\mathcal{O}(\binom{n(n-1)/2}{k/4})$.

This attack is essentially a time-memory tradeoff. The time-memory tradeoff attack that we present here employs a deterministic *baby-step giant-step* algorithm by Coppersmith summarised in [2, pp. 109] and detailed in [41, §2.1]. On human identification protocols that use a shared secret of k objects out of n , the resulting attack has average-case time complexity of:

$$\mathcal{O}\left(\frac{n^{k(1-\frac{\ln d}{2 \ln Q})} n^{\frac{\ln d}{\ln Q}}}{2^{\frac{k}{2} \frac{\ln d}{\ln Q}}}\right)$$

which is better than the former if $2^{k/2} < n$ or $k < 2 \log_2 n$. The space complexities of the two attacks are the same. While the time complexity is comparable to the previously mentioned meet-in-the-middle attack for generic human identification protocols, our attack however, performs much better on the two protocols in [8]. The average-case time complexity of our algorithm on the HB protocol is $\mathcal{O}(\binom{n/2}{k/2})$ and on the Sum of k Mins Protocol is $\mathcal{O}(\binom{n(n-1)/4}{k/4})$. This is substantially smaller than the previous result.

The original application of Coppersmith's algorithm is to solve the restricted Hamming weight discrete logarithm problem [42]. But since this algorithm essentially utilizes the knowledge of the restricted Hamming weight, we can modify it to solve our problem. We notice that there are several other deterministic algorithms that perform asymptotically better than Coppersmith's algorithm like the one proposed by Stinson of time complexity $\mathcal{O}\left(k^{3/2}(\ln n)\binom{n/2}{k/2}\right)$ [41]. However, for the choice of parameter values used in this chapter, the performance is comparable to Coppersmith's algorithm if not worse. There are also some probabilistic variants, but which cannot be applied here since the discrete logarithm is always unique and this is not necessarily the case with the candidate locations in our problem for small values of m . For larger values of m time-memory tradeoff algorithms become infeasible. We now describe the attack on our protocol and derive its time complexity. The derivations of the other results mentioned above are similar.

For simplicity, we assume n and k to be even integers. For arbitrary n and k , the attack can be carried out with minor differences [41, §5]. For $0 \leq i \leq n-1$, define \mathbf{b}_i to be a vector of length n such that $\mathbf{b}_i[l+1] = 1$ whenever, $l \equiv i+j \pmod n$, for $0 \leq j \leq n/2-1$, and 0 otherwise. Clearly, for all i , \mathbf{b}_i is a binary vector with $\text{wt}(\mathbf{b}_i) = n/2$. Let $\mathfrak{B} = \{\mathbf{b}_i : 0 \leq i \leq n/2-1\}$. Now, let Y be the set of all n -element vectors. From [41, §2.1], we see that for all $\mathbf{y} \in Y$ with $\text{wt}(\mathbf{y}) = k$, there exists a $\mathbf{b} \in \mathfrak{B}$, such that:

$$\mathbf{y} \cdot \mathbf{b} = \frac{k}{2}$$

For any $\mathbf{y}_1, \mathbf{y}_2 \in Y$, \mathbf{y}_2 is called the *sub* of \mathbf{y}_1 , denoted $\mathbf{y}_2 \prec \mathbf{y}_1$, if $\mathbf{y}_1[l] = 0 \Rightarrow \mathbf{y}_2[l] = 0$ for $1 \leq l \leq n$. Let $\mathbf{1}$ denote the binary vector of weight n . Let $\mathbf{y}_{=k/2}$ denote a vector whose weight is $k/2$. We divide \mathbf{s} into two parts: $\mathbf{s}_1 = [\mathbf{x} \ \mathbf{y}_{=k/2}]^T$ and $\mathbf{s}_2 = [\mathbf{0} \ \mathbf{y}_{=k/2}]^T$. We assume there to be a hash table, initially empty, which will be used as a data structure in this attack.

Attack: Attack 5.2.

Input: The set \mathfrak{B} and m challenge-response pairs.

Output: A set S of candidates for the secret \mathbf{s} .

- 1: Initialize an empty set S .
- 2: **for** $0 \leq i \leq n/2-1$ **do**
- 3: **for all** possible vector $\mathbf{s}_1 = [\mathbf{x} \ \mathbf{y}_{=k/2}]^T$ such that $\mathbf{y}_{=k/2} \prec \mathbf{b}_i$ **do**
- 4: Compute the string $q \leftarrow \mathbf{c}_1 \cdot \mathbf{s}_1 \pmod n \parallel \dots \parallel \mathbf{c}_m \cdot \mathbf{s}_1 \pmod n$.
- 5: Insert this m -digit string q along with \mathbf{s}_1 in the hash table.
- 6: **for all** vector $\mathbf{s}_2 = [\mathbf{0} \ \mathbf{y}_{=k/2}]^T$ such that $\mathbf{y}_{=k/2} \prec \mathbf{1} - \mathbf{b}_i$ **do**

```

7:      for  $1 \leq i \leq m$  do
8:          Initialize an empty set  $\mathfrak{Q}_i$ .
9:          for  $1 \leq j \leq n$ , if  $r_i \equiv (j + \mathbf{c}_i \cdot \mathbf{s}_2) \bmod n$ , update  $\mathfrak{Q}_i \leftarrow \{j\} \cup \mathfrak{Q}_i$ .
10:         Insert each string  $q \in \mathfrak{Q}_1 \times \cdots \times \mathfrak{Q}_m$  in the hash table along with  $\mathbf{s}_2$ .
11:     for each collision in the hash table, construct  $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$  and update  $S \leftarrow \{\mathbf{s}\} \cup S$ .
12:     Output  $S$ .

```

Once Attack 5.2 is executed, we need another attack to uniquely determine the vector in S that satisfies $m \geq \log_d(n \binom{n+k-1}{k/2})$ challenge-response pairs.

Attack: Attack 5.3.

Input: The set S and $m \geq \log_d(n \binom{n+k-1}{k/2})$ challenge-response pairs.

Output: An $\mathbf{s} \in S$.

```

1: for all  $\mathbf{s} \in S$  do
2:     if  $\mathbf{c}_i \cdot \mathbf{s} \equiv r_i \bmod n$  for  $1 \leq i \leq m$ , output  $\mathbf{s}$  and halt.

```

The memory requirement of Attack 5.2 is $\mathcal{O}(n \binom{n+k/2-1}{k/2})$. The combined running time of Attacks 5.2 and 5.3 is:

$$\mathcal{O}\left(n \binom{n/2 + k/2 - 1}{k/2} + \left(\frac{n}{d}\right)^m \binom{n/2 + k/2 - 1}{k/2} + \log_d \left(n \binom{n+k-1}{k} \right) \frac{\binom{n+k-1}{k}}{d^m} \right)$$

Neglecting the logarithmic terms as well as the quadratic term in n , we get:

$$\mathcal{O}\left(\frac{n^{m+1}}{d^m} \binom{n/2 + k/2 - 1}{k/2} + \frac{\binom{n+k-1}{k}}{d^m}\right)$$

Following a similar procedure to that of [8, §6], in Appendix A.2, we show that to minimize this quantity the optimum value of m for Attack 5.2 is:

$$m = \frac{\ln \left(\frac{\binom{n+k-1}{k} \ln d}{\binom{n/2+k/2-1}{k/2} \ln(n/d)} \right)}{\ln n} - 1$$

And this value of m gives the running time:

$$\mathcal{O}\left(\left(\binom{n+k-1}{k}\right)^{1-\ln d/\ln n} \binom{n/2+k/2-1}{k/2}^{\ln d/\ln n}\right)$$

In contrast, if we use the meet-in-the-middle algorithm from [8], we get a running time of:

$$\mathcal{O}\left(\left(\binom{n+k-1}{k}\right)^{1-\ln d/\ln n} \binom{n+k/2-1}{k/2}^{\ln d/\ln n}\right)$$

TABLE 5.1: The time and space complexity of the time-memory tradeoff attack.

n	k	τ_{tm}	μ_{tm}	n	k	τ_{tm}	μ_{tm}	n	k	τ_{tm}	μ_{tm}
100	10	2^{33}	2^{33}	120	10	2^{36}	2^{35}	150	10	2^{39}	2^{37}
	20	2^{55}	2^{52}		20	2^{60}	2^{55}		20	2^{65}	2^{58}
	30	2^{72}	2^{67}		30	2^{79}	2^{71}		30	2^{86}	2^{76}
200	10	2^{43}	2^{39}	300	10	2^{48}	2^{43}	500	10	2^{55}	2^{47}
	20	2^{72}	2^{63}		20	2^{83}	2^{69}		20	2^{96}	2^{77}
	30	2^{97}	2^{83}		30	2^{112}	2^{92}		30	2^{132}	2^{104}

which is considerably larger in the second term. Table 5.1 shows various choices of the parameters n and k and the resulting combined time and space complexity of Attacks 5.2 and 5.3. We assume $d = 10$ and the time and space complexities are represented by the symbols τ_{tm} and μ_{tm} respectively.

5.2.4 Comparative Time Complexities

The main motivation behind our protocol was to increase Q relative to d in Equation 5.4 without compromising too much on usability. If Q roughly equals the square of d , then we can choose smaller values of k , as the time complexity of the attack will increase. This is not straightforwardly possible in the HB protocol and Sum of k Mins. For instance, if $Q = 100 = 10^2 = d^2$, these protocols will require k additions of 2 digit numbers in a single round. This is prohibitively difficult for most humans since the additions have to be performed mentally. The Kangaroo Hopping Protocol achieves this by shifting the computations to the locations rather than the values of those locations. At each step, the user only has to add a 2 or 3 digit number to a single digit number. As a result, usability is preserved while the time-memory tradeoff attacks perform worse in our case. Table 5.2 shows a direct comparison of the three protocols in terms of the time-complexity of our attack. The time complexity of the meet-in-the-middle attack from [8] is labeled “Old”, whereas our attack is labeled “New”. As can be seen, our attack is efficient than the previous attack by a few orders of magnitude. Time-memory tradeoff attacks is one way to attack our protocol. The next section looks at a different way to attack the protocol.

5.2.5 Significance of the Jump Constant a

Let \hat{r} denote a location. Clearly it is an integer modulo n . We first attempt to find the probability distribution of obtaining \hat{r} as a sum of the values of k locations, ignoring the starting location and hence the jump constant a . To this end, let $p(k, \hat{r})$ be the probability that \hat{r} is the final location after the sum of the values of k locations as in our protocol. In other words, it denotes the probability that \hat{r} is the sum of k integers

TABLE 5.2: Time complexity of time-memory tradeoff attacks on three protocols.

n	k	k -weight HB		Sum of k Mins		Kangaroo Hopping	
		Old	New	Old	New	Old	New
100	8	2^{22}	2^{18}	2^{24}	2^{22}	2^{30}	2^{28}
	12	2^{30}	2^{24}	2^{34}	2^{31}	2^{41}	2^{38}
	16	2^{37}	2^{29}	2^{45}	2^{41}	2^{51}	2^{47}
200	8	2^{26}	2^{22}	2^{28}	2^{26}	2^{37}	2^{36}
	12	2^{36}	2^{30}	2^{40}	2^{37}	2^{52}	2^{49}
	16	2^{46}	2^{37}	2^{53}	2^{49}	2^{65}	2^{61}
300	8	2^{28}	2^{24}	2^{30}	2^{28}	2^{42}	2^{40}
	12	2^{40}	2^{34}	2^{44}	2^{41}	2^{58}	2^{56}
	16	2^{50}	2^{42}	2^{57}	2^{53}	2^{73}	2^{70}

(not necessarily unique): $s_1, \dots, s_k \in \mathbb{Z}_d$. Clearly, $p(1, \hat{r}) = 1/d$ for $0 \leq \hat{r} \leq d-1$ and $p(1, \hat{r}) = 0$ for $d \leq \hat{r} \leq n-1$. For any subsequent k , we see that the probability $p(k, \hat{r})$ can be obtained by:

$$p(k, \hat{r}) = \sum_{i=0}^{n-1} p(k-1, \hat{r} - i \bmod n) p(1, i)$$

This is similar to [8, §3.2]. The following dynamic programming algorithm of time complexity $\mathcal{O}(kn^2)$ then computes these probabilities:

Algorithm: Algorithm 5.1.

Input: n , k and d .

Output: The probabilities $p(k, \hat{r})$.

- 1: Assign $p(1, \hat{r}) \leftarrow 1/d$ for $0 \leq \hat{r} \leq d-1$ and $p(1, \hat{r}) \leftarrow 0$ for $d \leq \hat{r} \leq n-1$.
- 2: **for** $2 \leq i \leq k$ **do**
- 3: **for** $1 \leq j \leq n$ **do**
- 4: $p \leftarrow 0$.
- 5: **for** $1 \leq l \leq n$ **do**
- 6: $p \leftarrow p + p(i-1, j-l \bmod n) p(1, l)$.
- 7: $p(i, j) \leftarrow p$.
- 8: Output $p(k, \hat{r})$ for $0 \leq \hat{r} \leq n-1$.

Now, let $q(a, k, \hat{r})$ denote the probability of obtaining the location \hat{r} as the sum of k integers and the starting location, thus including the jump constant a . Then, we can see that:

$$q(a, k, \hat{r}) = \frac{1}{d} \sum_{i=0}^{d-1} p(k, \hat{r} + ia \bmod n)$$

Let U denote the uniform distribution over \mathbb{Z}_n . Let Q denote the distribution of the $q(a, k, \hat{r})$'s. $\Delta(Q, U) = \frac{1}{2} \sum_{i=0}^{n-1} |q(a, k, i) - \frac{1}{n}|$ is defined as the statistical distance between the two probability distributions.

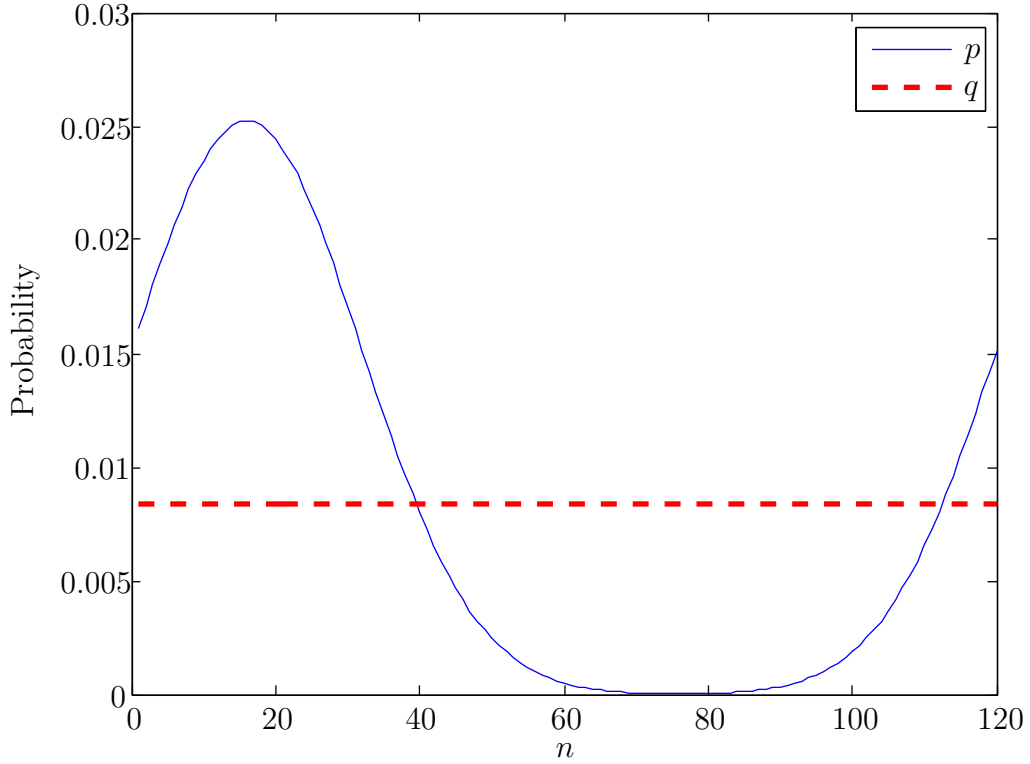


FIGURE 5.2: The jump constant a makes the distribution nearly uniform over n .

Lemma 4. Fix a k . Suppose d divides n . Then $\Delta(Q, U)$ is minimum if $a = \frac{n}{d}$.

Proof. a divides n into d blocks, each of length a . Since the starting location is uniform over \mathbb{Z}_d , its product with a will then be uniformly distributed over n . If $a \neq \frac{n}{d}$, then either $ad < n$ or $ad > n$. In both cases, the numbers between ad and n will have different probabilities of occurrence than the remaining numbers. \square

Figure 5.2 shows the distribution of $q(a, k, \hat{r})$ against $p(a, k, \hat{r})$ with $n = 120$, $k = 30$ and $a = 12$. Table 5.3 shows $\Delta(Q, U)$ with different values of n and k (a is chosen such that $n = ab$). Note that greater value of n requires larger value of k to make $\Delta(Q, U)$ small. Based on these results, we see that if the statistical distance is small, an adversary can distinguish from the uniform distribution after observing $\frac{1}{\Delta(Q, U)}$ challenge-response pairs on average. The adversary can then (possibly) use some statistical methods to guess the starting location and then use the following attack to guess the answer to a challenge probabilistically:

Attack: Algorithm 5.4.

Input: n , k , d and a challenge c .

Output: A guessed response.

TABLE 5.3: The statistical distance $\Delta(Q, U)$ against n and k .

n	k	$\Delta(Q, U)$	n	k	$\Delta(Q, U)$	n	k	$\Delta(Q, U)$
100	10	3.5×10^{-16}	120	10	4.9×10^{-8}	150	10	1.0×10^{-4}
	20	7.0×10^{-16}		20	3.3×10^{-15}		20	1.6×10^{-8}
	30	9.9×10^{-16}		30	1.0×10^{-15}		30	2.5×10^{-12}
200	10	7.2×10^{-3}	300	10	9.7×10^{-2}	500	10	3.3×10^{-1}
	20	8.2×10^{-5}		20	1.5×10^{-2}		20	1.7×10^{-1}
	30	9.3×10^{-7}		30	2.3×10^{-3}		30	8.8×10^{-2}

- 1: Run Algorithm 5.4 to obtain an interval of locations, $\delta \subseteq [0, n - 1]$, such that $\sum_{\hat{r} \in \delta} p(k, \hat{r}) = p(\delta)$, for some probability $p(\delta)$.
- 2: Guess a starting location and *shift* the interval δ accordingly.
- 3: Given a challenge \mathbf{c} , pick a location \hat{r} uniformly at random from δ and output $\mathbf{c}[\hat{r}]$.

Then the success probability of this attack is:

$$\frac{p(\delta)}{n\delta} + \frac{(1 - p(\delta))}{n^2}$$

Considering $p(\delta)$ to be high, we can see that the quantity $1/n\delta$ is less than $1/n^2$. If the adversary knows the starting location, then the probability of success becomes $1/\delta$. As an example, for the parameters used in Figure 5.2, if $\delta = [0, 49] \cup [100, 119]$, then $p(\delta) = 0.93$ and the success probability of the attack is: 1.34×10^{-4} which is considerably better than the naive guess which succeeds with probability 7.0×10^{-5} .

Notice that each session in our protocol consists of μ iterations. Therefore, in light of the discussion above, we mandate the use of our protocol for $\frac{1}{\mu\Delta(Q, U)}$ sessions only, before secret renewal.

5.3 Usability Analysis

To demonstrate comparable usability, we use similar parameters as used in the experiment in [8]. We use $n = 200$, $k = 15$ and $\mu = 6$. With these parameters, the time and space complexity of our time-memory tradeoff attack is proportional to 2^{61} and 2^{54} respectively. The statistical distance $\Delta(Q, U)$ is 4.9×10^{-4} , which means that the quantity $\frac{(\Delta(Q, U))^{-1}}{\mu} \approx 340$. Thus, with these parameters our scheme can be used securely for at least 340 authentication sessions.

With $n = 200$, $k = 15$ and $\mu = 7$, the experiment done for the usability of HB Protocol reported in [8] gave an average time of 166 seconds. Notice that there are $\mu = 7$ rounds instead of 6. This is important to add noise into the answer. The user sends the wrong answer to one of the challenges. To compare with our protocol, we

can see that apart from the starting location, the user has to add two numbers for each secret location. One of these numbers is in the range $[0, 199]$ and the other is in the range $[0, 9]$. Thus arguably, adding a single digit number to a number in the range $[0, 199]$ will take approximately the same time, as we are well versed with doing such computations in our heads. For the starting location, we see that the user can move vertically (with wraparound if necessary) according to the digit corresponding to the starting location. The user reaches to a new location this way. The result of the remaining $k = 15$ locations can then be divided by 20 to get a quotient and a remainder. The quotient is the number of vertical steps and the remainder is the number of horizontal steps to be taken. The user can then follow these steps and output the digit corresponding to the location thus reached. Thus while this last step takes more time than the other steps, it can surely be done within half the time required for the computation of the $k = 15$ other secret locations. Now, one iteration of the HB Protocol takes $166/7 \approx 23.7$ seconds on average. This implies that according to our argument, the computation of the last part takes ≈ 12 seconds. Thus, conjecturing that the calculations for the remaining $k = 15$ locations amounts to time 142.2 seconds, we can see that for $\mu = 6$ rounds, this amounts to a total time of ≈ 213 seconds.

From this discussion, we can say that for low values of α and β , our protocol is approximately $(\alpha, \beta, 213)$ -human executable. While it takes slightly more time than the HB Protocol, it is more usable as the user does not have to send a wrong answer with probability $1/7$, which is not possible for most humans. Furthermore, for these parameter choices, the time complexity of the time-memory tradeoff attack is proportional to 2^{37} in the case of the HB Protocol. In our case, the complexity is 2^{61} . The comparative time-complexity of the attack on the Sum of k Mins Protocol with similar parameters is 2^{49} . Again, lower than the time-complexity for our protocol. To increase usability, one can further reduce μ from 6 to 4 and get a time of approximately 143 seconds. Some of the common authentication mechanisms, such as PIN-based authentication, use 4-digit numbers for security. We acknowledge the absence of actual experiments on users.

5.3.1 Handling Errors

Whenever humans are involved in performing computations, errors are unavoidable. With the default setting of the protocol, the legitimate user can be rejected if he/she does one mistake in any of the μ rounds of the protocol. We can handle this by requiring that the user's answers be correct in most of the rounds. For instance, if $\mu = 6$, then the server accepts the user if 5 or more of the answers are correct. Although, the probability of success of a random guess attack will increase, it will still be considerably low. An interesting area of research would be to devise a protocol that handles user errors by using error correcting codes.

5.3.2 Suggested Values of Parameters

Table 5.4 shows choices of parameter values for different security requirements and the resulting parameterized security against different attacks. In the table, μ stands for the

TABLE 5.4: Suggested parameter values for the Kangaroo Hopping Protocol.

Security	n	k	m	p_{rg}	τ_{bf}	$\tau_{\text{tm}}/\mu_{\text{tm}}$	$\Delta(Q, U)$	Sessions
Low	200	12	4	10^{-4}	2^{71}	$2^{49}/2^{44}$	2.9×10^{-3}	85
Medium	200	16	4	10^{-4}	2^{87}	$2^{61}/2^{54}$	4.9×10^{-4}	500
High	200	20	6	10^{-6}	2^{101}	$2^{72}/2^{63}$	8.2×10^{-5}	2,000
Paranoid	200	24	6	10^{-6}	2^{114}	$2^{83}/2^{71}$	1.4×10^{-5}	12,000

number of iterations (rounds) in one authentication session. p_{rg} stands for the success probability of the random guess attack. τ_{bf} stands for the complexity of the brute force attack. $\tau_{\text{tm}}/\mu_{\text{tm}}$ shows the time/space complexity of the time-memory tradeoff attack. Finally, *Sessions*, represents the number of authentication sessions a particular secret can be used. It is obtained as $\frac{(\Delta(Q, U))^{-1}}{\mu}$. Notice that, space complexity can be a severe limitation as well ($2^{63}/8 \approx 10^{18}$, i.e. about 1 eta byte). For low and medium level security, the restriction on the number of sessions can be relaxed. Notice that in comparison with some other protocols found in literature, the number of sessions is quite high. For instance, the cognitive authentication scheme of Weinshall [10] can only be used for approximately 40 sessions even when the size of the user’s secret is as large as 150 [29, §4.1]. For more practical sizes of the secret, the number of allowable sessions is even lower. Similarly, Bai et al.’s scheme can be used for 10 authentication sessions only [17].² For these reasons, we have restricted our comparison to the two protocols in [8].

5.4 Conclusion

In comparison to protocols analyzed in Chapters 3 and 4, the Kangaroo Hopping Protocol described in this chapter can be used for a larger number of authentication sessions before security renewal. Our confidence in this claim is backed by a detailed security analysis in which we have shown that it is not possible to represent the challenge-response pairs as a system of linear equations with a non-negligible probability. Thus, the main design goal of the protocol was protection from the use of Gaussian elimination. As discussed before, the two protocols from [8] also attempt to do so; by the introduction of noise in the case of the HB Protocol, and by introducing pairs of minimums in the case of the Sum of k Mins Protocol. These “tricks” help in preventing the use of Gaussian elimination to find the secret after the observation of only $\mathcal{O}(n)$ authentication sessions.

Some of the protocols in literature have ignored the impact of time-memory trade-off attacks. We attempted to construct a protocol with security against time-memory tradeoff attacks in mind. The resulting protocol offers reasonable usability and good

²In fact, less than 7 according to our analysis in Chapter 3.

security. We acknowledge that the protocol can not be used frequently, as authentication seems to require about 2-3 minutes time. However, it can be used under certain circumstances such as when the user is using an insecure computer. An interesting question is whether improvements can be made to find a solution that achieves better security with progressively smaller values of parameters such as the size of the secret. Another area of interest is to find other variants of time-memory tradeoff attacks that can be applied to human identification protocols.

Perhaps the main drawback of the Kangaroo Hopping Protocol is that despite a detailed account of its security, it is not clear whether it is based on a hard to solve mathematical problem. This is unlike the HB Protocol, which is shown to be based on the NP-Hard problem of learning parity in the presence of noise. However, the HB Protocol actually uses a variant of the problem of learning parity in the presence of noise. In particular, it uses an unknown secret vector of low Hamming weight k . We will show later that this variant is hard in the sense of fixed-parameter intractability. We next show the construction of a protocol that is based on a hard mathematical problem in the sense of fixed-parameter intractability. Furthermore, we show that fixed-parameter intractability, in and of itself, does not guarantee that the protocol can be used for more than $\mathcal{O}(n)$ authentication sessions. A formal treatment of fixed-parameter intractability is the topic of Chapter 7. However, we shall give an intuitive introduction in the next chapter, which will help in understanding the reason for the choice of the particular problem to construct the protocol, called the Counting Edges Protocol.

6

Protocol Construction 2: Counting Edges

So far, we have seen that there does not appear to be a clear logical way to design human identification protocols. A major hurdle is that mathematical problems used in mainstream cryptography to build secure protocols cannot be directly used to construct human identification protocols. Discrete logarithm problem, factorization problem, etc., are only known to be hard with input sizes so huge, that it is beyond the capability of humans to do any computations on them mentally. In the absence of such hard problems, researchers have attempted to build their protocols on mathematical problems whose hardness is justified on the basis of infeasibility of all *known* attacks. Often, this approach has backfired with some protocols completely succumbing to simple innovative attacks. An example is the Predicate-based Authentication Service (PAS) from Bai et al. [17] discussed in Chapter 3. We have shown that PAS can remain secure for a much smaller number of authentication sessions than originally claimed by Bai et al. In short, systematic research in the area of human identification protocols should use known hard mathematical problems much like mainstream cryptography, but most problems from mainstream cryptography yield protocols that require computations beyond the capabilities of most humans.

In this chapter, we propose a new way of designing human identification protocols, i.e., based on fixed-parameter intractable problems. These problems are the subject of parameterized complexity theory, a relatively new area of complexity theory mainly introduced by Downey and Fellows [43]. We shall give a very brief yet intuitive explanation of a fixed-parameter intractable problem in this chapter. Detailed introduction to the area of fixed-parameter intractability together with reductions showing the fixed-parameter intractability of many problems used in human identification protocols is deferred to the next chapter. The main focus of this chapter is the construction of a protocol based on a fixed-parameter intractable problem. Our approach of constructing the protocol is as follows. We first construct a basic version that is based on a fixed-parameter intractable problem in a straightforward manner. We then alter the

protocol to mitigate security and usability problems. The protocol thus constructed can be considered as based on a “sparse” version of the underlying fixed-parameter intractable problem. By carefully choosing values of system parameters, we ensure that the adversary cannot gain advantage from the knowledge of sparsity of the problem.

This chapter is organized as follows. We first give a brief but intuitive explanation of what is meant by a fixed-parameter intractable problem in Section 6.1. We then proceed to describe the first construction of the new protocol in Section 6.2. On the basis of subsequent security analysis, we describe the weaknesses of the first construction in Section 6.4. In Section 6.5 we describe the main construction of the protocol, followed by a thorough security analysis. The last section contains concluding remarks.

6.1 Fixed-Parameter Intractable Problems

The HB Protocol from [8] is based on an NP-Hard problem, called LEARNING PARITY WITH NOISE or LPN in short.¹ Recall that the protocol is similar to the description of the Example Protocol in Chapter 2, with the major difference that \mathcal{H} sends a wrong response bit with probability $\epsilon < 0.5$. The adversary \mathcal{A} who observes challenge-response pairs from the HB Protocol, needs to find the secret vector \mathbf{x} given the challenges and the responses (dot products of the challenges and the secret vector). The adversary also knows that with a known probability $\epsilon < 0.5$, the responses are incorrect. In other words, \mathcal{A} has to solve the NP-Hard problem, LPN. NP-Hardness of the LPN problem implies that the best known algorithms to solve the problem run in time exponential in the size of the input, i.e., n . However, the underlying problem used in HB is slightly different from the LPN problem.

Recall that Hopper and Blum suggested restricting the Hamming weight of the secret to k to make it easier for humans to memorize the secret, where k can be small, say around 15. This can be easier for humans since they only need to memorize the locations of the 1’s. Thus, the underlying problem has an additional parameter, k . Let us call this the k -LPN problem. If it is shown that k -LPN is NP-Hard, it only tells us that the best known algorithms to solve this problem require time exponential in n or k . It is possible that there is an algorithm that is only exponential in k . Consider for instance the VERTEX COVER problem which is known to be NP-Complete. Yet, there is an algorithm that solves this in time $\mathcal{O}(kn + (\frac{4}{3})^k k^2)$ [44]. Even though this is an exponential algorithm in the size of the input (k and/or n), since k has to be small in our case, this yields an efficient algorithm in practice. We are instead more interested in knowing if the best possible algorithms to solve a particular problem run in time $n^{f(k)}$ for some function f . If n is large and k is modest, which is the case with human identification protocols, these algorithms can be prohibitive in practice. Consider for example the exhaustive search algorithm of time complexity $\mathcal{O}(\binom{n}{k})$. With $n = 200$ and $k = 15$, the run time of this algorithm is about 2^{73} . Through the theory of fixed-parameter intractability, which treats k as a parameter separate from the input, it can be shown that a problem is intractable in the sense that the run-time of algorithms

¹From now onwards we shall denote complexity theoretic problems in small capitals.

to solve it will have k in the exponent of n . Such problems are called fixed-parameter intractable.

We shall next show a protocol, called Counting Edges Protocol, which is based on a fixed-parameter intractable problem. As mentioned before, we shall not describe the reduction showing the fixed-parameter intractability of the underlying problem. We defer this to the next chapter, where we will give a more technical background to the theory of fixed-parameter intractability and show that many problems used in human identification protocols are in fact based on fixed-parameter intractable problems, without the inventors of these protocols realizing the link between these two areas.

A Note of Caution

A hardness result showing that a human identification protocol is based on a fixed-parameter intractable problem should not be taken as a certificate for the security of the protocol. There are two main reasons for this skepticism. First, the reduction only guarantees that the problem is NP-Hard and/or fixed-parameter intractable in the worst case. However, the instances of this problem generated in most of these protocol are random. It might be the case that random instances of the problem are not as hard as the worst case instances. We will show an example of this in our first protocol, described next. Second, the protocol might have other weaknesses that can be exploited to impersonate \mathcal{H} , without having to find the secret by solving the hard problem. For instance, Li and Shum found several weaknesses in the HB protocol (based on the LPN problem) in [14, §4.3.1, p. 23]. They noticed, for instance, that since the Hamming weight of the secret is small, the dot product of the secret and the challenge vectors is more likely to be 0 as compared to 1, if the number of 1's in the challenge vector are small. With this knowledge, the adversary has a higher probability of impersonating \mathcal{H} . Notice that this does not mean that the adversary can guess the secret. Instead, it shows that the adversary still has a good chance of impersonating \mathcal{H} without having to find the secret vector \mathbf{x} ; merely the knowledge of $\text{wt}(\mathbf{x}) = k$ is enough. Thus, while the hardness results in this thesis show that the mathematical primitives used to construct protocols are hard to solve, they should not be solely taken as a guarantee of security of the protocols. A protocol might have other weaknesses that can be exploited to impersonate \mathcal{H} without having to solve the underlying hard mathematical problem itself. A thorough security analysis is, therefore, still necessary.

6.2 The Counting Edges Protocol: First Construction

As described in the introduction, we first give a basic construction of a protocol based on a fixed-parameter intractable problem, and then extend it to a full protocol to resolve security and usability issues. The main reason for involving a graph is the goal of prolonging the lifespan of the protocol to $\mathcal{O}(n^2)$ authentication sessions before secret renewal; thus mitigating the drawback mentioned in the Example Protocol in Chapter

2. The protocol is based on counting edges in a small subset of vertices in a graph. Appendix A.3 contains the terminology of graphs used here.

Protocol: Counting Edges (Basic Construction).

Setup: Let μ , n and $k \leq n$ be publicly known positive integers. Let $V = \{1, \dots, n\}$ be a set of vertices. \mathcal{H} and \mathcal{C} share a k -element (or k -vertex) subset K of V as a secret.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} creates an undirected graph $G_i = (V, E_i)$, where E_i is chosen randomly.
- 3: \mathcal{H} counts the number of edges in the induced subgraph, $G_i[K]$, and sends the result to \mathcal{C} .
- 4: **if** all μ responses are correct, \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

6.3 Security Analysis

To analyze the security of the basic protocol, we look at the protocol from an adversarial perspective and attempt to find ways to attack the protocol. In the course of analyzing the protocol, we shall also show that the underlying problem is fixed-parameter intractable. Unfortunately, as we shall see later, this does not guarantee the security of the protocol for a sufficiently large number of observed sessions.

6.3.1 Impersonation without the knowledge of K

As always we shall first see how \mathcal{A} can impersonate \mathcal{H} without finding the secret set of vertices K , through random guesses. The number of edges in an induced graph of k vertices can range between 0 and $\binom{k}{2}$.² Thus, \mathcal{A} can simply sample an integer in the range $[0, k(k-1)/2]$ according to a certain probability. As we already know, this is known as the random guess attack. The probability of generating an integer from $[0, k(k-1)/2]$ depends on the way the graphs G_i are generated. Denote by G , the graph generated by \mathcal{C} in an arbitrary challenge. G is a random graph on V , where the randomness stems from the way the edge set E is generated. There are a total of $\binom{n}{2}$ possible edges. Assume that each edge is generated with probability p . Then, a graph G with e edges is generated with probability:

$$p^e(1-p)^{\binom{n}{2}-e}.$$

If $p = 1/2$ then all possible $2^{\binom{n}{2}}$ graphs are generated with equal probability [45]. We assume that this is the probability with which each edge is generated. This implies that the sum of edges in any induced k -subgraph is binomially distributed. More specifically, let $\kappa = \binom{k}{2}$. Then, the probability that $G[K]$ has i edges in a randomly generated graph G is:

$$\frac{\binom{\kappa}{i}}{2^\kappa}$$

²A 0-edge induced subgraph is an edgeless graph.

for $0 \leq i \leq \kappa$. Thus, an adversary can sample an integer in the range $[0, \frac{k(k-1)}{2}]$ which has the maximum probability in the binomial distribution. Specifically, he can reply with the integer $\lfloor k(k-1)/4 \rfloor = \lfloor \frac{\kappa}{2} \rfloor$. Then, the probability that the adversary is successful is:

$$\frac{\binom{\kappa}{\lfloor \frac{\kappa}{2} \rfloor}}{2^\kappa}$$

Denote the term above by $p(\kappa)$. The protocol can be iterated enough times so that the probability of a random guess is low. In other words, choose a μ such that the probability $p(\kappa)^\mu$ is small. We can choose values of these parameters such that this probability is as low as 10^{-6} , which as mentioned earlier will suffice to be an acceptable level of security [8]. For instance, if $k = 10$, we can choose $\mu = 6$ or 7 . In Chapter 2, we called this attack Random Guess 1. That is, random guess on the response space.

6.3.2 Randomly Guessing the Secret

The other form of random guess attack, Random Guess 2 as mentioned in Chapter 2, targets the secret space. More specifically, the adversary randomly chooses a valid secret and then correctly follows the steps of the protocol using the randomly chosen secret. As we have seen, the success probability of this attack depends on the size of the secret space. Since the secret space is of size $\binom{n}{k}$ in our protocol, the probability of success of correctly guessing the secret is $\binom{n}{k}^{-1}$ for our protocol. For $n = 200$ and $k = 10$, the success probability is about 2^{-54} , which is very low. Due to this reason, unless otherwise specified, whenever we mention random guess attack here onwards, we shall mean the random guess attack on the response space.

6.3.3 Finding K

Now, let us assume that \mathcal{A} is given m challenge-response pairs. Then an intersection attack similar to the one mentioned in Chapter 2 can be used to find the secret K as follows. Denote the m challenge-response pairs by $(G_1, r_1), \dots, (G_m, r_m)$.

Attack: Intersection Attack.

Input: m challenge response pairs $(G_1, r_1), \dots, (G_m, r_m)$.

Output: An $s \in S$, where S is the set of candidates for the secret.

- 1: Choose the pair (G_1, r_1) and find k -vertex subsets in G_1 whose induced subgraphs in G_1 have r_1 edges (One can in fact choose any of the m pairs). Let S be the set containing all such k -vertex subsets.
- 2: **for** $i = 2$ to m **do**
- 3: **if** an element (k -vertex set) in S , does not have an induced subgraph in G_i with r_i edges **then** discard the element.
- 4: **if** S contains only one element **then** output the element **else** sample an element randomly from S as the output (Note again that an element of S is a set of k vertices).

The main part of the attack is Step 1, in which all k -vertex subsets of G_1 (or equivalently V) have to be found whose induced subgraphs in G_1 have r_1 edges. Note that finding one such subset does not suffice. To see this, first note that the probability that the induced subgraph of any k -vertex subset of G_1 has r_1 edges is given by:

$$\sum_{i=0}^{\kappa} \left(\frac{\binom{\kappa}{i}}{2^{\kappa}} \frac{\binom{\kappa}{i}}{2^{\kappa}} \right) = \sum_{i=0}^{\kappa} \left(\frac{\binom{\kappa}{i}}{2^{\kappa}} \right)^2 = \frac{1}{2^{2\kappa}} \sum_{i=0}^{\kappa} \binom{\kappa}{i}^2 = \frac{\binom{2\kappa}{\kappa}}{2^{2\kappa}}$$

Where we have used the identity [46, p. 135]:

$$\sum_{i=0}^{\kappa} \binom{\kappa}{i}^2 = \binom{2\kappa}{\kappa}$$

Thus, on average there are:

$$\frac{\binom{2\kappa}{\kappa}}{2^{2\kappa}} \binom{n}{k}$$

possible k -vertex subsets of V whose induced subgraphs in G_1 have r_1 edges. If we put $n = 200$ and $k = 10$, this amounts to about 2^{51} 10-vertex subsets. Thus, if an algorithm outputs one k -vertex subset of V whose induced subgraph in G_1 has k_1 edges, the probability that this is the secret K is very low. So, a better strategy is to keep the list of all “satisfying” k -vertex subsets and further refining the list with the knowledge of more challenge-response pairs. One can find all such sets of k vertices with a brute force attack of complexity $\mathcal{O}(\binom{k}{2} \binom{n}{k})$. The question is, can one do better?

Another way of executing Step 1 is to count all r_1 -edge induced subgraphs in G_1 , with exactly k vertices. One way to do so is to solve the k -EDGE INDUCED SUBGRAPH problem. Given a graph G , this problem asks to decide whether G contains an induced subgraph of r edges. Note that a solution to this problem does not necessarily imply that the resulting induced subgraph has exactly k vertices. Nevertheless, this problem has recently been shown to be fixed-parameter tractable [47]. In other words, there exists an algorithm that solves the problem in time exponential in k but polynomial in n . On the other hand, the counting version of the problem, which asks to count the number of r -edge induced subgraphs in G is fixed-parameter intractable [47]. Denote this problem by k -#EDGE INDUCED SUBGRAPH.³ Note that induced subgraphs can have isolated vertices too. As counting all r_1 -edge induced subgraphs in G_1 is fixed-parameter intractable, this approach does not seem to produce an efficient attack (in the sense of fixed-parameter tractability). However, as mentioned before, the problem presented to the adversary is a little different from p -#EDGE INDUCED SUBGRAPH. The adversary at least knows that the induced subgraph of r_1 edges has exactly k vertices. The exact problem can then be stated as follows:

Given m pairs $(G_1, r_1), \dots, (G_m, r_m)$, where G_i is a graph on a vertex set V of n vertices, find a k -element subset K of V such that the induced subgraphs $G_i[K]$ have r_i edges, for $1 \leq i \leq m$.

³For counting problems, we use the symbol # following convention. More on this appears in the next chapter.

As mentioned before, when $m = 1$ there are many candidates for the secret set K . The problem is to find a candidate when m is such that a unique k -vertex subset is expected (information theoretic bound). We can estimate this as:

$$\begin{aligned}
& \left(\frac{\binom{2\kappa}{\kappa}}{2^{2\kappa}} \right)^m \binom{n}{k} < 1 \\
\Rightarrow & \binom{n}{k} < \left(\frac{\binom{2\kappa}{\kappa}}{2^{2\kappa}} \right)^{-m} \\
\Rightarrow & \log_2 \binom{n}{k} < -m \left(\log_2 \binom{2\kappa}{\kappa} - \log_2 2^{2\kappa} \right) \\
\Rightarrow & m > -\frac{\log_2 \binom{n}{k}}{\left(\log_2 \binom{2\kappa}{\kappa} - \log_2 2^{2\kappa} \right)}
\end{aligned}$$

Recall that $\kappa = \binom{k}{2}$. Denote this lower bound on m by m_{lb} . For instance, when $n = 200$ and $k = 10$, $m > 16$ challenge-response pairs are expected to give a unique solution, which will be the secret K . For any $m > 1$, we call this problem **COMMON VERTEX SUBSET**. In Section 7.5 of the next chapter, we show that this problem is fixed-parameter intractable. Of course, as mentioned earlier, this result only shows that the problem is fixed-parameter intractable in the worst case. Since, the graphs generated in the protocol are random, the result does not necessarily imply hardness of the problem in this case. Consider for instance a randomized probabilistic algorithm that when given the pair (G_1, r_1) simply outputs a random k -vertex subset of V . If run $2^{2\kappa} / \binom{2\kappa}{\kappa}$ times, the algorithm is expected to output a k -vertex subset of V whose induced subgraph contains r_1 edges with high probability. Of course, this algorithm will not perform well with $m > m_{\text{lb}}$ pairs, as now we expect a unique k -vertex subset to satisfy all m pairs.

6.4 Drawbacks of the Basic Protocol

Unfortunately, there is an efficient way of attacking the protocol with relatively small values of m ; around 300 to 400. To understand this, denote by $x_{i,j}$ a variable that is 1 if vertices v_i, v_j are both in K and 0 otherwise. Let A be a binary matrix whose i th row corresponds to the “stretched out” adjacency matrix of graph G_i . That is, let A_i denote the adjacency matrix of G_i . Then we create a row vector from this matrix of length $\eta = \binom{n}{2}$, with the entries of A_i in this order: $a_{1,2}, a_{1,3}, \dots, a_{1,n}, a_{2,3}, \dots, a_{n-1,n}$. This row vector is then the i th row of A . Let \mathbf{r} denote the vector of m elements whose entries are the m responses, i.e., m edge-counts. Then, we have the following system of linear equations:

$$A\mathbf{x} = \mathbf{r} \tag{6.1}$$

where \mathbf{x} is the unknown vector of the $x_{i,j}$ ’s with $\text{wt}(\mathbf{x}) = \kappa$. Figure 6.1 shows two graphs G_1 and G_2 on the vertex set $V = \{1, 2, 3, 4\}$. For these two graphs the $m \times \eta = 2 \times 6$

matrix A is as follows,

$$\begin{matrix} & 1,2 & 1,3 & 1,4 & 2,3 & 2,4 & 3,4 \\ G_1 & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \\ G_2 & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{matrix}$$

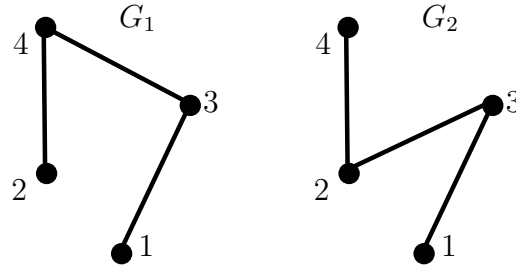


FIGURE 6.1: Two graphs G_1 and G_2 on the vertex set $V = \{1, 2, 3, 4\}$.

If there is no efficient algorithm for solving the system of Equation (6.1) for $m < \eta$, then the protocol can be used until $\eta = \binom{n}{2}$ observed challenge-response pairs, since Gaussian elimination cannot be used for $\eta < \binom{n}{2}$ to uniquely solve for the secret. In other words, potentially, the protocol can be used for $\mathcal{O}(n^2)$ number of challenge-response pairs, instead of just $\mathcal{O}(n)$ in the case of the Example Protocol mentioned in Chapter 2. This is the main idea behind the design of this protocol. Unfortunately, there is an efficient way to solve this system of equations with a far smaller value of m through linear programming, by considering Equation (6.1) as an integer linear program. It is important to note that an accurate representation of the problem in terms of an integer linear program should also include other constraints, such as when $x_{i,j} = 1$ and $x_{j,k} = 1$ then $x_{i,k} = 1$, for all $i, j, k \in V$. However, with a sufficiently large m , the adversary does not need to be concerned about these constraints as then a unique solution to the above system of equations will obey such constraints. If $\kappa \ll \eta$, we can solve this problem with a linear programming relaxation. We relax the integer restriction, i.e., we assume $\mathbf{x} \in \mathbb{R}$, and then attempt to solve the following linear program:

$$\text{minimize } \mathbf{1} \cdot \mathbf{x}^T \text{ subject to } A\mathbf{x} = \mathbf{r}, \mathbf{x} \geq \mathbf{0}$$

It turns out that with an appropriate matrix A and sufficiently large m , the solution returned by the linear program is in fact the solution of the original problem with high probability [48]. With $n = 200$, $k = 10$ and $m = 400$, we ran the linear programming solver `lp_solve`⁴ using MATLAB on randomly generated instances of A and \mathbf{x} . The solver returned the correct secret in all 10 trials with an average time of about 100 seconds. On the other hand, with $m = 200$ the solver did not output the correct secret, even after running it a number of times. The probability of finding the correct

⁴<http://lpsolve.sourceforge.net/5.5/>

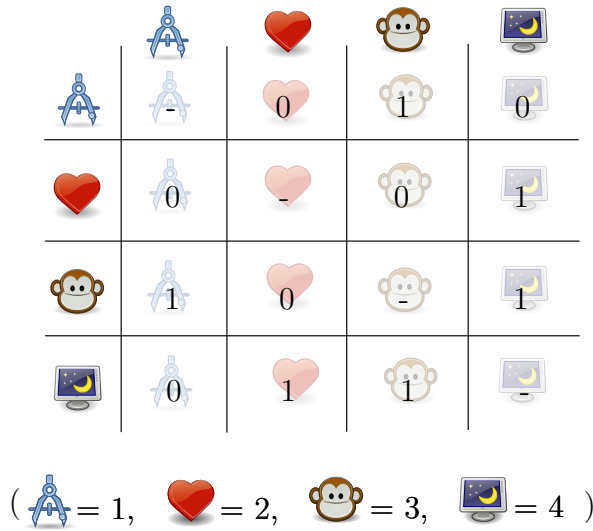


FIGURE 6.2: An example implementation of the basic Counting Edges Protocol.

secret was also very low for $m = 300$. Thus, it is evident that for some m in the range $[300, 400]$ with high probability the linear program above gives a solution that is also the solution of the original problem with $\text{wt}(\mathbf{x}) = \kappa$. Although, this program takes an m greater than m_{lb} , the value of m that can be safely used is not sufficient for practical purposes. This will only allow m/μ sessions before the secret needs to be renewed. We can choose μ to be 6 so that the chance of success of the random guess attack is around one in a million. With $m = 200$, this means only about 33 sessions after which the secret needs to be renewed.

Another drawback relates to the implementation of the protocol. We can implement the protocol using software icons similar to the one for the Example Protocol described in Chapter 2. A challenge can then be displayed as a grid of software icons connected through random edges. However, this way could be cumbersome for the user as n grows. A better way is to display the adjacency matrix of the graph. The borders of the matrix can be identified by software icons. In this case, the user will have to find entries in the matrix corresponding to his secret set of icons, and sum the entries (which will be binary). See Figure 6.2. The figure shows the adjacency matrix of the graph G_1 from Figure 6.1. The user only remembers the secret icons, and counts the positions corresponding to pairs of secret icons. The transparent icons in the background of each “cell” make it easier for the user to locate the correct entry. Still however, the drawback of this protocol is displaying an adjacency matrix of size up to 200×200 , which will definitely depend on the size of the display. We consider these two points and construct the revised Counting Edges Protocol next.

6.5 The Counting Edges Protocol: Main Construction

We make two main changes in the revised version of the protocol. First, the response is now an integer modulo a positive integer $d > 1$. This is to prevent the problem from being represented as a linear or integer program, and hence mitigating the attack mentioned before. Secondly, we now only show a subset V' of nodes from V , in a challenge. This makes it easier for the user to locate the secret vertices among the reduced set of vertices shown on the user's terminal. We also limit the maximum number of non-secret vertices displayed to $k' < k$. This reduces the computation time per challenge. Once again, the idea is to be able to use the protocol for $\mathcal{O}(\binom{n}{2})$ authentication sessions before secret renewal.

Protocol: Counting Edges (Main Construction).

Setup: Let μ , n and $k \leq n$ be publicly known positive integers. Let $V = \{1, \dots, n\}$ be a set of vertices. \mathcal{H} and \mathcal{C} share a k -element (or k -vertex) subset K of V as a secret. Also, let $n' \leq n$, $k' \leq k$ and $d > 1$ be (publicly known) non-negative integers.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} creates an undirected graph $G_i = (V', E'_i)$, where V' is a random n' -vertex subset of V containing k' vertices from K , and E'_i is chosen randomly.
- 3: \mathcal{H} counts the number of edges in the induced subgraph, $G_i[K']$, where K' is a k' -vertex subset of K , i.e., it contains those secret vertices that are present in G' . \mathcal{H} sends the result modulo d to \mathcal{C} .
- 4: **if** all μ responses are correct, \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

Note that the problem is now different from the COMMON VERTEX SET problem, as we now return an edge count modulo an integer d . However, we shall also show in the next chapter (cf. Section 7.5) that this variant is also fixed-parameter intractable.

6.6 Security Analysis

Note that we have used the modulus to avoid the problem being represented as a linear program such that the protocol can be used for $\mathcal{O}(\binom{n}{2})$ authentication sessions before secret renewal. In spite of this, and the fact that the underlying problem is fixed-parameter intractable, we still need to consider the exact running time of attacks so that we can choose values of system parameters accordingly. This will allow us, for instance, to more accurately calculate the number of authentication sessions the protocol can be used before secret renewal. Specifically, we conjecture that on the basis of the evidence from the following analysis, in light of Definition 4, the Edge Counting Protocol is $(d^{-\mu}, \binom{n}{2})$ -secure against passive adversaries. In the following, we let $\eta = \binom{n}{2}$, $\eta' = \binom{n'}{2}$, $\kappa = \binom{k}{2}$ and $\kappa' = \binom{k'}{2}$.

6.6.1 Fine-tuning Protocol Parameters

Since now the secret and non-secret vertices are chosen by \mathcal{C} differently, the two probabilities should be same to ensure that an attacker cannot find the secret simply by examining the frequency with which a vertex appears in challenges. For instance, if the secret vertices are more likely to appear in a challenge, the attacker can simply guess that the vertices that appear more often in a given set of challenges are the secret vertices. Now, the probability with which a secret vertex appears is k'/k . The same probability for a non-secret vertex is $(n' - k')/(n - k)$. If both these probabilities are equal, we get:

$$\begin{aligned}
 \frac{k'}{k} &= \frac{n' - k'}{n - k} \\
 \Rightarrow \frac{k'}{(n - k)} &= \frac{k}{(n' - k')} \\
 \Rightarrow k'n - k'k &= kn' - kk' \\
 \Rightarrow k'n &= kn' \\
 \Rightarrow \frac{n}{k} &= \frac{n'}{k'} \tag{6.2}
 \end{aligned}$$

This gives us the following theorem,

Theorem 2. *For any challenge in the Counting Edges Protocol, the probability of appearance of a secret vertex is the same as the probability of appearance of a non-secret vertex, if and only if $\frac{n}{k} = \frac{n'}{k'}$.*

In other words, according to Equation (6.2), the ratio of n' to k' should be equal to the ratio of n to k . Thus, if we choose $n = 200$ and $k = 20$, we can use $n' = 50$ and $k' = 5$. Another consideration is regarding the distribution of the responses. Since a response is now an integer modulo d , we need to ensure that the reply is almost uniformly distributed in $\{0, \dots, d - 1\}$. Let $\kappa' = \binom{k'}{2}$. Then, as we have seen before, the probability that the induced graph $G'[K']$ has i edges is:

$$\frac{\binom{\kappa'}{i}}{2^{\kappa'}}$$

where $0 \leq i \leq \kappa'$. Let $p(j)$ be the probability of obtaining the integer j as the reply to a challenge, where $j \in \{0, \dots, d - 1\}$. Then, we can see that:

$$p(j) = \sum_{i \equiv j \pmod{d}} \frac{\binom{\kappa'}{i}}{2^{\kappa'}}$$

Thus, we should choose a $d > 1$, such that with a given k' , the probabilities $p(j)$ are almost the same. For instance, with $k' = 5$ and $d = 3$, we get:

$$\begin{aligned}
 p(0) &= 0.3330078125, \\
 p(1) &= 0.3330078125, \\
 p(2) &= 0.333984375.
 \end{aligned}$$

Unfortunately, for other choices of d , the difference in probabilities is considerably larger, meaning that μ has to be increased to make the success probability of random guess equivalent to the case when $d = 3$.

6.6.2 Meet-in-the-middle Attack

The meet-in-the-middle attack, described in great detail in the previous chapter, is a general algorithm to search a space of $\binom{n}{k}$ objects. This has been applied on certain counting problems in graphs [49], and also as attacks on human identification protocols [8]. In a nutshell, the meet-in-the-middle strategy attempts to reduce the search space of $\mathcal{O}(\binom{n}{k})$ possibilities, to $\mathcal{O}(\binom{n}{k/2})$, by storing the intermediate results in a table, and finding if there are any collisions. Of course, for this strategy to be successful, it should be possible to decompose the search space into half, without having to do further computations when the two halves are joined. In the COMMON VERTEX SUBSET problem, halving the search space does not seem to work. This is so because counting edges in two induced subgraphs of $k/2$ disjoint vertices does not give us the result when considering the induced subgraph of the union of the two $k/2$ -vertex sets, since the vertices from the first vertex-set might be connected to vertices in the second set.

This is not surprising, since to the best of our knowledge there is no meet-in-the-middle attack for a related problem, k -CLIQUE (cf. Section 7.5 of Chapter 7). However, we can still deploy this strategy on the adjacency matrix formulation of the problem as described in the previous section. More specifically, the problem can now be described as follows:

$$A\mathbf{x} \equiv \mathbf{r} \pmod{d}. \quad (6.3)$$

However, since \mathbf{x} is a vector of $\eta = \binom{n}{2}$ unknowns with weight $\kappa = \binom{k}{2}$, the complexity of the meet-in-the-middle attack on this system is quite large. Recall that in Chapter 5 we proposed an improved variant of the meet-in-the-middle attack on such systems of linear equations. The attack from [8] and the one mentioned in Chapter 5 have the same space complexity, roughly $\binom{\eta}{\kappa/2}$, where η is the number of unknowns and κ is the Hamming weight of the unknown vector. The time complexity of the attack from [8] on this protocol is $\binom{\eta}{\kappa/2}$, whereas the time complexity of the attack from Chapter 5 is $\binom{\eta/2}{\kappa/2}$. For $n = 200$ and $k = 20$, the first meet-in-the-middle attack on the above system of linear equations has time complexity in the order of 2^{864} , which is huge. The meet-in-the-middle attack from Chapter 5 has a complexity of about 2^{769} , which is still enormous. Notice that the brute force attack has complexity 2^{90} , with these parameters. Of course, after obtaining $\binom{n}{2}$ challenge-response pairs, the adversary can find the secret with high probability using Gaussian elimination. Thus, we limit the use of the protocol to $\binom{n}{2}$ challenge-response pairs before secret renewal.

6.6.3 Attacks from Coding Theory

Another way of looking at Equation (6.3) is from a coding theory perspective (see Appendix A.4 for a brief introduction to coding theory). If we consider A to be the

parity check matrix, and the vector \mathbf{r} to be the syndrome of an unknown codeword, then finding an \mathbf{x} satisfying Equation (6.3) is the same as finding an error vector of weight $\kappa = \binom{k}{2}$. Although A is a random matrix, with large enough n , it is highly likely that all m rows of A are linearly independent. In fact, given a random matrix A with elements from \mathbb{Z}_3 , the probability that it has the full rank n , converges to 0.56012 as n grows [22]. From a security perspective, therefore, it is safer to assume that all m rows are linearly independent. If not, the attacker can still choose m linearly independent rows from $\mathcal{O}(m)$ observations. Thus, assuming A to be of full rank, when $m > m_{\text{lb}} = \log_d \binom{\eta}{\kappa}$ we expect to obtain a unique codeword of weight κ through decoding.⁵ We will see in the next chapter that with $d = 2$ this is the MAXIMUM LIKELIHOOD DECODING problem, which is both NP-Complete and fixed-parameter intractable. It can be shown that the problem is NP-Complete with any d [50, Appendix A].

Still, if the Hamming weight of \mathbf{x} is very low, we can use information set decoding algorithms to find the secret \mathbf{x} [51]. The main idea behind information set decoding is to find an information set. That is, find a set of m columns of A such that they correspond to all the non-zero entries in \mathbf{x} . If the resulting $m \times m$ submatrix of A is invertible, then Gaussian elimination can be used to recover \mathbf{x} . Niebuhr et al. [50] generalize some of the information set decoding algorithms for finite fields \mathbb{F}_d other than \mathbb{F}_2 . Translated to our formulation, they give lower bounds on the best known information set decoding algorithms for different values of d , n , m and k . To understand the complexity of the attacks, we consider concrete values of parameters, that are suggested in Table 6.1. Note that $\eta' = \binom{n'}{2}$ and $\kappa' = \binom{k'}{2}$. With these values, the system in Equation (6.3) has 19900 unknowns, with the Hamming weight of the unknown \mathbf{x} being 190 and the modulus $d = 3$. We see that these values of parameters are higher than those evaluated in [50, §3, p. 7]. Although, $d = 3$ is not explicitly used, it is evident, that for much smaller values of η' and κ' these algorithms are infeasible. Since the matrix A is generated at random, it is reasonable to assume that information set decoding algorithms are the best form of attack on the system from a coding theory perspective, since information set decoding algorithms can be used even if the underlying code structure is unknown.

One could also use the knowledge of sparsity of the matrix A , and use Low Density Parity Check (LDPC) decoding to obtain a solution [52]. LDPC codes use a sparse parity check matrix. In our case A is sparse, with the average row weight being, $\frac{\eta'}{\eta} \cdot \frac{1}{2} \cdot \eta = \frac{\eta'}{2}$ and average column weight being, $\frac{m}{\eta} \cdot \frac{\eta'}{2}$. With the specified set of system values, and say for $m = \frac{\eta}{2}$, we get an average row weight of 612.5 and average column weight of 306.5. Thus, we can consider the problem as maximum likelihood decoding of irregular LDPC codes.⁶ However, we can see that for comparable values of parameters, LDPC decoding using information set decoding algorithms takes in order of 2^{92} operations [53, p. 8]. However, in the case considered in [53] the matrix A is hidden. It remains an open problem to evaluate the complexity of LDPC decoding over \mathbb{Z}_3 with our suggested values of parameters.

⁵Note that we are using the same notation m_{lb} for a different lower bound.

⁶Regular LDPC codes have fixed row and column weights.

6.6.4 Coskun and Herley's Divide-and-Conquer Attack

We briefly discussed Coskun and Herley's divide-and-conquer attack in Section 2.5 of Chapter 2. To recall, Coskun and Herley's attack is a clever brute force strategy to find the secret in protocols which use a subset of the secret to compute responses to challenges [27]. Since the Counting Edges Protocol uses $k' < k$ secret vertices in each challenge, their attack is applicable to this protocol. However, Coskun and Herley's attack was described on a "hypothetical" human identification protocol; one that uses an N -bit secret, out of which U bits are used to construct a response to a challenge, and secrets can differ from each other by 1 bit. In the Counting Edges Protocol, and many other human identification protocols, the secret is a set of k objects from a total of n . Thus, their attack needs to be generalized on such " k -out-of- n " protocols. For instance, in k -out-of- n protocols, it makes no sense to consider secrets that are different in one bit. We have to instead look at secrets that differ from each other in one or more objects.

In Appendix A.5, we extend this attack on the Counting Edges Protocol, which requires substantial revisions of some of the results from [27]. We also implemented the attack on the Counting Edges Protocol, and the simulation results show that the attack is infeasible in practice, on the Counting Edges Protocol. This is probably due to the fact that the number of bits of the secret used to respond to each challenge is substantially higher in the Counting Edges Protocol, than the numbers evaluated by Coskun and Herley in [27]. See Appendix A.5 for more details.

6.7 Usability Analysis

Table 6.1 summarizes the suggested values for the system parameters. The values of the system parameters are chosen according to the time and/or space complexity of the best known attacks, and the graphic presentation of the protocol. μ is chosen such that the success probability of random guess attack is less than 10^{-6} . The "Sessions" column indicates the number of sessions allowed before secret renewal. It is chosen according to $\binom{n}{2}/\mu$. The total number of edges in a subgraph of $k' = 5$ nodes can be at most $\kappa' = \binom{k'}{2} = 10$. Thus, the user has to count at most 10 edges in a graph of $n' = 50$ nodes, which is arguably practical. As far as implementation of the protocol is concerned, we can use either a graph based implementation or an adjacency matrix based implementation. We can use an implementation based on software icons similar to the one shown in Figure 6.2. The difference now is that the position of the icons is randomly permuted such that the adversary cannot gain the knowledge of the secret by merely checking the location of appearance of icons. This random shuffle is bound to increase execution time of the protocol as users will have to locate 5 secret icons in a total of 50 cells in the display screen. If we assume that it takes 2 seconds for the user to locate a secret icon, and a further 1 second to count each edge, then each iteration of the protocol can be done in $10 + 10/2 = 15$ seconds on average (since on average there will be $10/2$ edges present). For $\mu = 13$ iterations, this amounts to 195 seconds.

By comparison, the average authentication time for the HB Protocol was shown to

TABLE 6.1: Suggested values of parameters for the Counting Edges Protocol.

n	k	n'	k'	d	η	κ	η'	κ'	μ	Sessions
200	20	50	5	3	19900	190	1225	10	13	1530

be about 160 seconds [8]. However, it should be noted that their usability study was done with values of system parameters that are not large enough for sufficient security. The specific values used for their usability study were $n = 200$ and $k = 15$. The best known (passive) attack on the HB Protocol is the meet-in-the-middle attack mentioned before, of space complexity $\mathcal{O}(\binom{n}{k/2})$. With these values of system parameters the space complexity is about 2^{46} , which cannot be considered intractable. If we raise this to around 2^{60} then the HB Protocol is bound to take more time per authentication session. Of course, there are some other drawbacks of the HB Protocol as well, such as the requirement to send the wrong response with probability $\epsilon < 0.5$. Thus, the Counting Edges Protocol appears to be better in terms of the balance between usability and security. To conclude, we conjecture that for low values of α and β , the Counting Edges Protocol is $(\alpha, \beta, 195)$ -human executable. We acknowledge the lack of empirical evidence to back this claim.

6.8 Conclusion

The Counting Edges Protocol proposed in this chapter has been constructed from a hard mathematical problem, specifically a fixed-parameter intractable problem. However, the variant of the protocol that we propose for use in practice is essentially based on a sparse version of the original problem. We use parameters $n' < n$ and $k' < k$, to ensure that there are not too many icons displayed on the user's screen in the implementation of the protocol. The security of the protocol is based on the assumption that there are no substantially better algorithms to solve the sparse variant of the original problem. We have also analyzed the ways in which the sparsity of the underlying problem can be used to attack the protocol. We do not claim that the discussion is comprehensive. For instance, in the main Counting Edges Protocol, there might be a way to solve the problem of finding a k -vertex subset having a specified number of edges in a sparse graph, without having to resort to the much harder adjacency matrix formulation of the problem. To the best of our knowledge, there does not appear to be an easy way to solve this problem. Indeed, it is known that the closely related DENSEST k -SUBGRAPH problem (See Section 7.5) remains NP-Hard even when the graph G has maximum degree 3 [54].

The approach used in this chapter is to find fixed-parameter intractable problems to construct human identification protocols. Problems from this area of computational complexity theory are appealing since they are “hard” with a fixed parameter, k . This

fixed parameter can be naturally mapped to the size of the secret in human identification protocols. However, as we have seen, mere fixed-parameter intractability is not enough to guarantee the security of the protocol. We have needed to further analyze security employing the randomness of the instances generated by the Counting Edges Protocol. And, where there are efficient algorithms for random instances, we have discarded the use of the underlying problem (cf. the basic Counting Edges Protocol). Secondly, in order to obtain better usability we needed to generate sparse instances of the underlying problem. If an efficient generic algorithm to attack the sparse variants of fixed-parameter intractable problems exists, then perhaps we cannot hope for a practical human identification protocol.

The underlying problem of the Counting Edges Protocol, namely COMMON VERTEX SUBSET, is not the only fixed-parameter intractable problem used in the construction of human identification protocol. As we show in the next chapter, many existing human identification protocols are actually based on fixed-parameter intractable problems, without the inventors of these protocols realizing the link. These include, the HB Protocol and the Sum of k Mins Protocol from [8], the Foxtail Protocol from [14], and even the Example Protocol from Chapter 2. However, mere fixed-parameter intractability does not tell us how many authentication sessions a protocol can be used. This shall be clear next, where we give a background on the theory of fixed-parameter intractability, and show that the above mentioned protocols as well as the Counting Edges Protocol are based on fixed-parameter intractable problems.

7

Fixed-Parameter Intractable Problems in Human Identification Protocols

As mentioned in the prelude to the previous chapter, to date only one human identification protocol is known to be based on a hard mathematical problem. This protocol is the HB protocol [8], named after its inventors Hopper and Blum. The protocol is based on the problem of LEARNING PARITY WITH NOISE (LPN in short), which is an NP-Hard problem. Moreover, there is some evidence which suggests that it is hard even in the random case [8, 55]. Apart from this protocol, there is no known human identification protocol in literature based on an NP-Hard or NP-Complete problem. However, the problems used in the construction of other protocols have often been conjectured to be hard. For instance, the other protocol from [8] is based on the so-called SUM OF k MINS problem, which Hopper and Blum claim to be some sort of “sparse subset sum problem” [8, §3.2]. However, without a formal reduction we can only speculate on the hardness of a problem. Indeed, it is known that the SUBSET SUM problem is solvable in pseudo-polynomial time [56, p. 247]. Thus, it is desirable to know the hardness of problems using formal complexity theoretic reductions.

Furthermore, as we discussed in the last chapter, mere NP-Hardness of a problem does not suffice in case of human identification protocols. Such protocols use problems that have another explicit parameter, k . This is necessary so that humans can easily remember a secret of k objects in a large pool of n objects. Thus, even though LPN is NP-Hard, the HB protocol proposed for human identification in [8] is based on a variant of LPN; one that employs a restricted Hamming weight. To show the hardness of problems that have an additional parameter k , we use the theory of fixed-parameter intractability, introduced mainly by Downey and Fellows [43], and the notion of fixed-parameter intractable problems from this theory.

This Chapter is organized as follows. Just as in the previous chapter, we begin by giving a motivating example using the HB Protocol in Section 7.1; only this time our

treatment is more detailed. Section 7.2 contains a brief background on parameterized complexity theory. This is followed by a series of reduction results showing the fixed-parameter intractability of the underlying problems in the Sum of k Mins Protocol in Section 7.3, HB and the Example Protocol in Section 7.4, the Counting Edges Protocol in Section 7.5, and the Foxtail Protocol [14] in Section 7.6. The final section contains the concluding remarks.

7.1 A Motivating Example: The HB Protocol

The HB protocol [8], named after its inventors Hopper and Blum, has been the focus of researchers as a protocol for resource constrained devices [55]. However, the original protocol was intended for human identification. The protocol is as follows.

Protocol: HB.

Setup: Let μ and n be publicly known positive integers. Let $0 \leq \epsilon < 1/2$ be the noise parameter. \mathcal{H} and \mathcal{C} share a secret $\mathbf{x} \in \mathbb{Z}_2^n$.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} samples a random $\mathbf{c} \in \mathbb{Z}_2^n$ and sends to \mathcal{H} .
- 3: \mathcal{H} computes $r = \mathbf{c} \cdot \mathbf{x}^T$. With probability ϵ , \mathcal{H} sends the wrong response $1 - r$, else he sends the correct response r to \mathcal{C} .
- 4: **if** the number of correct responses are $\geq (1 - \epsilon)\mu$ **then** \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

Consider the problem presented to the adversary, \mathcal{A} . \mathcal{A} is given a few challenge-response pairs (the challenge vectors \mathbf{c}_i 's and their responses r_i 's). In order to find the secret vector \mathbf{x} , \mathcal{A} has to solve the so-called LEARNING PARITY WITH NOISE problem which is known to be NP-Hard [8, 57]. As before, we shall call this the LPN problem in short. If the above mentioned protocol does not have any other weaknesses, finding the secret might be the only feasible way to impersonate \mathcal{H} .

For the sake of usability, Hopper and Blum proposed two main changes to the protocol. One was changing the base from 2 to 10, which is arguably easier for most humans. The second was restricting the Hamming weight of \mathbf{x} to k . In such a case, k can be small, say around 15. Let us call this the k -LPN problem, where k emphasizes the fact that $\text{wt}(\mathbf{x}) = k$. The issue here is that once k is fixed, this problem is not NP-Hard. An obvious brute force algorithm to find \mathbf{x} runs in time $\mathcal{O}(\binom{n}{k})$. A better algorithm sketched by Hopper and Blum is the meet-in-the-middle attack, which works in time $\mathcal{O}(\binom{n}{k/2})$. These two terms are bounded by $\mathcal{O}(n^k)$. Since, k is fixed, these algorithms are polynomial in the size of the problem.

Notice that $\mathcal{O}(n^k)$ is an exponential time bound, if k is not fixed and is considered as part of the input to the algorithm. However, in the context of human identification protocols, k has to be kept small due to the inherent limitations of human memory. Every protocol designed for human identification would require a “cap” on the cardinality of the secret, k , or else it is not practical. In general among a pool of n objects,

\mathcal{H} would have to choose k objects as a secret. As mentioned above, there is a trivial brute force algorithm that runs in time $\mathcal{O}\left(\binom{n}{k}\right)$ to find the secret for such protocols.¹ If k is small and fixed, this means that it is easy to break the protocol in the sense of the traditional notion of polynomial time computability. However, if n is large, say 200, then a modest value of k , say 15, means that the run time of the brute force algorithm is about 2^{73} . Thus, even though the algorithm is polynomial in n , it is not feasible in practice.

Traditional complexity theory divides problems into two main classes: those that can be solved by polynomial time algorithms and those that can be solved by only exponential time algorithms. This split is made on the grounds of *asymptotic* behavior of algorithmic complexities. This asymptotic behavior is of little relevance to human computational and memorizing abilities, which are limited. Due to this reason the value of k in particular has to be kept small. Thus, an algorithm that takes exponential time only in k , can be an efficient algorithm in practice. An example, that we have already mentioned, is that of the VERTEX COVER problem that is known to be NP-Complete. However, there is an algorithm that solves this in time $\mathcal{O}(kn + (\frac{4}{3})^k k^2)$ [44]. This time is exponential if k is considered as part of the input. But if this problem is to be used as a primitive for a human identification protocol, then this algorithm can efficiently find the secret. For instance, if $n = 200$ and $k = 15$, the running time of the algorithm is only about 2^{14} in practice. Compare this to the brute force attack which takes time 2^{73} .

Researchers have in fact implicitly assumed that the best known attacks on their proposed human identification protocols run in time close to $\mathcal{O}\left(\binom{n}{k}\right)$, or $\mathcal{O}(n^{f(k)})$ for some function f . The protocols from [8] and the CHC Protocol from [21], discussed in Chapter 4 are examples. However, they have not explicitly shown whether these claims have any ground. Through the theory of fixed-parameter intractability we can show that the best possible algorithms to solve a problem run in time $n^{f(k)}$ for some function f . This framework allows us to formally show the (worst-case) hardness of the underlying problems used in human identification protocols. Although these results do not directly show the hardness of random instances of the problems, they give some evidence that the problems used to construct human identification protocols are likely to be hard.

7.2 Parameterized Complexity Theory

All the definitions used in this section are taken from [20] with similar notation. Let Σ be an alphabet. Let $Q \subseteq \Sigma^*$ be a problem in classical complexity theory. Define $\kappa : \Sigma^* \rightarrow \mathbb{N}$ to be a parameterization of Q . That is, to every finite string x the parameterization of Q associates a natural number $\kappa(x)$. The problem (Q, κ) is called a parameterized problem.

Definition 9. A parameterized problem (Q, κ) is called *fixed-parameter tractable* if there is an algorithm that decides Q in time at most $f(\kappa(x))p(|x|)$ for some computable

¹provided, of course, that at least one challenge-response pair has been observed.

function f and some polynomial p .

An algorithm that runs within the aforementioned time is called an fpt-algorithm. The class FPT contains all fixed-parameter tractable problems. This can be considered as the class of problems easy to solve much like the class PTIME in classical complexity. Let (Q, κ) and (Q', κ') be two parameterized problems over alphabets Σ and Σ' .

Definition 10. An fpt-reduction from (Q, κ) to (Q', κ') is the map $R : \Sigma \rightarrow \Sigma'$ such that

1. For all $x \in \Sigma^*$, $x \in Q$ if and only if $R(x) \in Q'$.
2. R is computable by an fpt-algorithm with respect to κ .
3. There is a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, such that $\kappa'(R(x)) \leq g(\kappa(x))$, for all $x \in \Sigma^*$.

The corresponding notion of a nondeterministic Turing machine in parameterized complexity theory is the notion of κ -restricted nondeterministic Turing machine [20]. Let $\kappa : \Sigma^* \rightarrow \mathbb{N}$ be a parameterization.

Definition 11. A nondeterministic Turing machine \mathcal{M} with input alphabet Σ is called κ -restricted, if there are computable functions f and h , and a polynomial p , such that on every input $x \in \Sigma^*$, the machine runs for at most $f(\kappa(x))p(|x|)$ steps, at most $h(\kappa(x)) \log |x|$ of which are nondeterministic.

Definition 12. $W[P]$ is the class of all parameterized problems (Q, κ) which can be decided by a κ -restricted nondeterministic Turing machine.

Both FPT and $W[P]$ are closed under fpt-reductions, and $FPT \subseteq W[P]$ [20]. By closure, we mean if a problem (Q, κ) is fpt-reducible to a problem $(Q', \kappa') \in FPT$ (resp. $W[P]$), then $(Q, \kappa) \in FPT$ (resp. $W[P]$). There is in fact a whole hierarchy of classes called the W-hierarchy. Each class in this hierarchy is closed under fpt-reductions [20], so that the following is true:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq W[P]$$

The reader interested in the difference in these classes can refer to [20]. For the sake of this text, it suffices to know that all classes $W[t]$, with $t \geq 1$ and the class $W[P]$ are fixed-parameter intractable. In other words, it is unlikely that there is an fpt-algorithm that solves these problems, in the same sense that there is unlikely that polynomial time algorithms exist for NP-Hard or NP-Complete problems. Furthermore, we shall not be interested in the membership of any problem in the W-Hierarchy. Instead we shall only be interested in the question whether a problem is as hard as any problem in the W-Hierarchy and hence is not fixed-parameter tractable.

7.2.1 Parameterized Counting Problems

$\#P$ is the class of intractable counting problems in classical complexity theory. Interestingly, the counting versions of many easy decision problems are $\#P$ -Complete. For instance, deciding if a bipartite graph contains a perfect matching is polynomial time solvable. On the other hand, counting the number of perfect matchings in a bipartite graph is $\#P$ -Complete, and is therefore a hard problem [20, 58]. Just like the W -Hierarchy for fixed-parameter intractable decision problems, there is a corresponding $\#W$ -Hierarchy for fixed-parameter intractable counting problems. Following the notation used in [20], the class of fixed-parameter tractable counting problems will also be denoted by FPT.

Here again, there are many fixed-parameter tractable decision problems whose corresponding counting versions are intractable. An example is finding a path of length k in a directed or undirected graph. Let us call this problem k -PATH.² This problem is fixed-parameter tractable as it can be solved by an algorithm running in time $2^{O(k)}n^{2.376}$ [20, 59]. The corresponding counting problem of counting the number of paths of length k in a directed or undirected graph, i.e., k - $\#PATH$, is $\#W[1]$ -Complete [60].

7.3 The Sum of k Mins Protocol

The section shows the hardness of the SUM OF k MINS problem, used as a primitive in the Sum of k Mins Protocol from [8]. Since this is the first of the hardness results in this chapter, our treatment is very detailed. We first describe the problem, and show how this is used as a primitive in the Sum of k Mins Protocol from [8]. This is followed by illustrating our method of reduction on a variant of the SUM OF k MINS problem (one without the modulus). We then move to show that the original problem is both NP-Complete and $W[1]$ -Hard.

A construct that shall prove useful in the following discussion is the map:

$$\text{em} : \mathbb{Z}^n \rightarrow \mathbb{Z}^{\binom{n}{2}}$$

which is defined as:

$$\begin{aligned} \text{em}(\mathbf{v}) = & (\min\{v_1, v_2\}, \dots, \min\{v_1, v_n\}, \\ & \min\{v_2, v_3\}, \dots, \min\{v_{n-1}, v_n\}) \end{aligned}$$

where $\mathbf{v} = (v_1, \dots, v_n)$ and $\text{em}(\mathbf{v})$ has $\binom{n}{2} = \frac{n(n-1)}{2}$ elements. Notice that $\text{em}(\mathbf{v})$ contains all possible pairwise minimums from \mathbf{v} . We say that \mathbf{v} is *expanded by taking minimums*.

²The correct notation should be (PATH, k) , where PATH is the classical problem of finding a path of maximum length in the graph. However, we shall shorten this by using the prefix k before the name of a classical problem. In some problems the parameter k shall be explicit. Such as the SUM OF k MINS problem which does not have a classical counterpart. In such a case we shall not use the prefix k .

7.3.1 The Sum of k Mins Problem

We now restate the SUM OF k MINS problem from [8] using similar notation. Let n and k be positive integers, where $k \leq n$. Also, let $d \geq 2$ be a positive integer. Let $z = \{(x_1, y_1), \dots, (x_k, y_k)\}$ be a set of ordered pairs, where $1 \leq x_i, y_i \leq n$, and for each ordered pair (x_i, y_i) , $x_i \neq y_i$. Furthermore, let \mathbf{v} be an n -tuple whose elements are integers mod d . Define $f(\mathbf{v}, z)$ as:

$$f(\mathbf{v}, z) = \sum_{i=1}^k \min\{\mathbf{v}[x_i], \mathbf{v}[y_i]\} \bmod d$$

Then the SUM OF k MINS problem is:

Given m ordered pairs $(\mathbf{v}_1, u_1), \dots, (\mathbf{v}_m, u_m)$, find a set z such that $u_i = f(\mathbf{v}_i, z)$ for all $i = 1, \dots, m$, where:

$$\mathbf{v}_i \in \mathbb{Z}_d^n, u_i \in \mathbb{Z}_d, \text{ and } \log_d \binom{n(n-1)/2}{k} < m < \binom{n}{2}.$$

As the reader might have guessed, the inequality:

$$\log_d \binom{n(n-1)/2}{k} < m = m_{\text{lb}}$$

is required to ensure a unique solution, and is the oft-mentioned information theoretic bound on m . Finding a solution to the problem will be relatively easier, for say $m = 2$, since on average there will be about $\frac{1}{d^2} \binom{n(n-1)/2}{k}$ solutions (provided the n -tuples are generated uniformly at random). Thus, this is the information theoretic lower bound to guarantee a unique solution. $m < \binom{n}{2}$ is necessary since this is the expected number of pairs required to use Gaussian elimination to solve uniquely for z . Notice that the $m = \binom{n}{2}$ pairs obtained should be linearly independent in order to apply Gaussian elimination. However, we can safely assume that below this bound the adversary cannot use Gaussian elimination since it requires at least $\binom{n}{2}$ pairs. We have used the same argument in the previous chapters and it applies whenever we mention the use of Gaussian elimination throughout this chapter. The difficulty in solving the problem lies in the conjecture that there is no known efficient algorithm that can solve the problem given m ordered pairs within these two bounds.

7.3.2 The Protocol

This problem can be used as a primitive to construct a human identification protocol described in the following. It is a reduced version of the protocol presented in [8], since we consider passive adversaries only.

Protocol: Sum of k Mins.

Setup: \mathcal{H} and \mathcal{C} agree on a secret which is a set of k ordered pairs of indices:
 $z = \{(x_1, y_1), \dots, (x_k, y_k)\}$ such that $1 \leq x_i, y_i \leq n$, and for each ordered pair (x_i, y_i) , $x_i \neq y_i$.

- 1: \mathcal{H} sends an identification request to \mathcal{C} .
- 2: **for** $i = 1$ to μ **do**
- 3: \mathcal{C} sends a random n -tuple, $\mathbf{v} \in \mathbb{Z}_d^n$ to \mathcal{H} .
- 4: \mathcal{H} computes the sum of k mins function $f(\mathbf{v}, z)$ as described above and returns the answer to \mathcal{C} .
- 5: **if** all the μ answers are correct **then** \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

For implementation, typical values of parameters can be $d = 10$, $k = 12$, $n = 100$ to 200, and $\mu = 6$ [8]. This gives a one in a million chance for success of a random guess. The passive adversary \mathcal{A} can see the random n -tuples sent by \mathcal{C} and the corresponding responses by \mathcal{H} . If the adversary wishes to find the secret shared between the two, it can do so by solving the SUM OF k MINS problem.

7.3.3 Matrix Representation

To better understand the SUM OF k MINS problem, it is helpful to represent it in terms of matrices as is done in [8] as follows:

$$\begin{bmatrix} v_{1,1,2} & v_{1,1,3} & \cdots & \cdots & v_{1,n-1,n} \\ v_{2,1,2} & \cdots & v_{2,i,j} & \cdots & v_{2,n-1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m,1,2} & \cdots & v_{m,i,j} & \cdots & v_{m,n-1,n} \end{bmatrix} \begin{bmatrix} z_{1,2} \\ \vdots \\ z_{i,j} \\ \vdots \\ z_{n-1,n} \end{bmatrix} \equiv \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} \pmod{d}$$

Here, $v_{l,i,j} = \min\{\mathbf{v}_l[i], \mathbf{v}_l[j]\}$, for $1 \leq l \leq m$, $z_{i,j} = 1$ if $(i, j) \in z$, $z_{i,j} = 0$ if $(i, j) \notin z$, and $1 \leq i < j \leq n$. Notice that row i of the left hand side matrix is $\mathbf{em}(\mathbf{v}_i)$, for $1 \leq i \leq m$. Let this system be denoted by $V\mathbf{z} \equiv \mathbf{u} \pmod{d}$. The problem then translates into finding the binary vector \mathbf{z} of Hamming weight k . Since this is a system of $\binom{n}{2}$ unknowns, we need $m \geq \binom{n}{2}$ to solve uniquely for z using Gaussian elimination. However, since the Hamming weight k is restricted, we can still find a unique solution with $m \geq m_{\text{lb}} = \log_d \binom{n(n-1)/2}{k}$. But, for $m < \binom{n}{2}$, the best known algorithms are the aforementioned meet-in-the-middle attack which can solve this particular problem with time complexity $\mathcal{O}(\binom{n(n-1)/2}{k/2})$ [8]. A slightly better version of the meet-in-the-middle attack shown in Chapter 5 works with time complexity $\mathcal{O}(\binom{n(n-1)/4}{k/2})$. These algorithms are better than brute force but are still superpolynomial in the size of the secret if k is logarithmic in n .

Hopper and Blum mentioned that representing the problem in this way leads to a sparse subset sum problem [8, §3.2]. However, this way of describing the problem bears more resemblance to some sort of maximum likelihood decoding problem [44]. We call this problem the MODULO d MAXIMUM LIKELIHOOD DECODING problem defined as follows:

Given an $m \times n$ matrix A with entries from \mathbb{Z}_d for some $d \geq 2$ and a vector $b \in \mathbb{Z}_d^m$, is there a binary vector \mathbf{x} of Hamming weight k such that $A\mathbf{x} \equiv \mathbf{b} \pmod{d}$?

With $d = 2$, this is the MAXIMUM LIKELIHOOD DECODING problem described in [44], which is not only NP-Complete but also W[1]-Hard [44, 61]. It should be noted that the version described in [44] and [61] asks for an \mathbf{x} with Hamming weight *at most* k . However, it is easy to see that the hardness results also apply to the version with an exact Hamming weight k . As mentioned before, W[1]-Hardness implies that the problem is fixed-parameter intractable in the sense that it cannot be solved in time $f(k)p(n)$ for some computable function f and some polynomial p [44]. In other words the best algorithms for this problem are not expected to run in time less than $n^{f(k)}$ for some function f .

Nevertheless, this transformation does not guarantee that the original SUM OF k MINS problem has no other weaknesses which could be exploited to solve it efficiently without having to solve the MODULO d MAXIMUM LIKELIHOOD DECODING problem. From here onwards, we shall call the original SUM OF k MINS problem, the MODULAR SUM OF k MINS problem to distinguish it from a generalized form of the problem described next, which will simply be referred to as the SUM OF k MINS problem. This generalized problem is used to illustrate the reduction used in the main result, which is the NP-Completeness and W[1]-Hardness of MODULAR SUM OF k MINS.

7.3.4 Generalized Sum of k Mins

We define the (generalized) SUM OF k MINS problem as follows:

Given positive integers n and m , a collection of n -tuples $\mathbf{v}_1, \dots, \mathbf{v}_m$ with non-negative integers as elements, and a collection of non-negative integers u_1, \dots, u_m , is there a set of pairs of indices $\{(x_1, y_1), \dots, (x_k, y_k)\}$ with $k \leq n(n-1)/2$ and $1 \leq x_i < y_i \leq n$, such that:

$$u_j = \sum_{i=1}^k \min \{ \mathbf{v}_j[x_i], \mathbf{v}_j[y_i] \}$$

for all $j = 1, \dots, m$?

We will illustrate our method of reduction by reducing a form of 0-1 INTEGER PROGRAMMING problem to it. We call this problem the 0-1 INTEGER PROGRAMMING FEASIBILITY problem or 0-1 IPF in short, following a terminology similar to [62]. 0-1 IPF can be described as follows:

Given positive integers n and m , an $m \times n$ matrix A with non-negative integer entries, and an m -tuple \mathbf{b} whose elements are non-negative integers, is there a binary vector \mathbf{x} of Hamming weight $k \leq n$ such that:

$$A\mathbf{x} = \mathbf{b}?$$

The reason for calling this 0-1 IPF is that this is the feasibility version of one class of 0-1 INTEGER PROGRAMMING problems (namely the class of problems in which the coefficients are non-negative integers).

Illustration through an Example

We illustrate our method of reduction with the help of a small example. Suppose, $A = \mathbf{a} = (8, 3, 9, 5)$ and $b = 17$ is an instance of 0-1 IPF. That is, with $n = 4$ and $m = 1$. In terms of a matrix equation, this becomes:

$$\begin{bmatrix} 8 & 3 & 9 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = 17$$

We need to find values of the binary variables x_i , for $i = 1, 2, 3, 4$. If we treat this as an instance of SUM OF k MINS with $m = 1$, we get:

$$\begin{bmatrix} 3 & 8 & 5 & 3 & 3 & 5 \end{bmatrix} \begin{bmatrix} z_{1,2} \\ z_{1,3} \\ z_{1,4} \\ z_{2,3} \\ z_{2,4} \\ z_{3,4} \end{bmatrix} = 17$$

where the vector on the left hand side is $\mathbf{em}(\mathbf{a})$; the expansion of \mathbf{a} by taking minimums. As can be seen, this trivial transformation does not give the required result, as expansion by taking minimums changes the structure of the original 4-tuple. That is, after expanding the original n -tuple by taking minimums, the largest integer is left out (if it occurs only once). Additionally, it disturbs the original order of integers in the n -tuple. However, notice that if we append the largest element in front of the n -tuple, then the first n integers, after expanding by taking minimums, will be the original integers in the same order. Thus, if we create a 5-tuple $\mathbf{a}' = (9, 8, 3, 9, 5)$, then the equation becomes:

$$\begin{bmatrix} 8 & 3 & 9 & 5 & 3 & 8 & 5 & 3 & 3 & 5 \end{bmatrix} \begin{bmatrix} z_{1,2} \\ \vdots \\ z_{4,5} \end{bmatrix} = 17$$

The first four integers of $\mathbf{em}(\mathbf{a}')$ are the same as in \mathbf{a} . All we need to do now is to somehow remove the “influence” of the last 6 digits, in the above mentioned row vector, from the overall equation. To do this we can introduce 6 more 5-tuples, such that the last 6 variables in the equation above equate to 0. Thus after taking minimums, each of these 5-tuples should have only a single 1 at the right place (corresponding to the binary variable) with all other entries being 0. We can see that these 5-tuples should have only two 1's such that after taking mins there is only one 1 left. All other elements should be 0. Let us introduce the 6 new 5-tuples, $\mathbf{v}_1, \dots, \mathbf{v}_6$:

$$\begin{aligned} &(0, 1, 1, 0, 0) \\ &(0, 1, 0, 1, 0) \\ &(0, 1, 0, 0, 1) \\ &(0, 0, 1, 1, 0) \\ &(0, 0, 1, 0, 1) \\ &(0, 0, 0, 1, 1) \end{aligned}$$

Then the m -pairs $(\mathbf{a}', b), (\mathbf{v}_1, 0), \dots, (\mathbf{v}_6, 0)$ can be represented as a SUM OF k MINS problem as:

$$\begin{bmatrix} 8 & 3 & 9 & 5 & 3 & 8 & 5 & 3 & 3 & 5 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots & & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_{1,2} \\ \vdots \\ z_{4,5} \end{bmatrix} = \begin{bmatrix} 17 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Thus any solution to the problem will have the last six binary variables fixed at 0. Thus if the original instance of 0-1 IPF has a solution, so does this system of equations. This illustrates the transformation we are going to use in our main result. Note that this illustration can easily be extended for any $m \geq 1$, as is done in the formal reduction described next.

Formal Reduction

We now formalize the results illustrated in the previous example. Our first lemma shows that $(n+1)$ -tuples of the form of \mathbf{a}' in the example above, have the same property in general.

Lemma 5. *Let $\mathbf{a} = (a_1, \dots, a_n)$ be an n -tuple, whose elements are non-negative integers. Let $a' = \max\{\mathbf{a}\}$ be the largest element of \mathbf{a} . Let $n' = n + 1$ and let $\mathbf{a}' = (a'_1, a'_2, \dots, a'_{n'})$ be an n' -tuple such that $a'_1 = a'$ and $a'_i = a_{i-1}$ for $2 \leq i \leq n'$. For $1 \leq i < j \leq n'$, define:*

$$b_{i,j} = \min\{a'_i, a'_j\}$$

then $b_{1,j} = a_{j-1}$ for $2 \leq j \leq n'$. In other words, the first n elements of $\mathbf{em}(\mathbf{a}')$ will be the elements of \mathbf{a} in the same order.

Proof. Since $a' = \max \mathbf{a}$, the minimum of a' with any element a_i will be the element a_i . Since it is the first element of \mathbf{a}' and the remaining n elements are exactly the elements of \mathbf{a} , the result follows. \square

The following lemma shows that the binary n -tuples of the form described in the example above have the same property in general as is exploited in the example.

Lemma 6. *Let n be a positive integer. Let i be an integer between 2 and $n-1$ inclusive and s be an integer between $i+1$ to $n-i$ inclusive. Let \mathbf{v} be an n -tuple with binary elements such that: $v[i] = 1$, $v[s+i] = 1$ and $v[j] = 0$, for all $j \neq i, s+i$. Let \mathbf{v}' be the $n(n-1)/2$ -tuple: $\mathbf{em}(\mathbf{v})$. That is, with elements:*

$$\begin{aligned} v'_1 &= \min\{v_1, v_2\} \\ \vdots & \quad \quad \quad \vdots \\ v'_{n-1} &= \min\{v_1, v_n\} \\ v'_n &= \min\{v_2, v_3\} \\ \vdots & \quad \quad \quad \vdots \\ v'_{n(n-1)/2} &= \min\{v_{n-1}, v_n\} \end{aligned}$$

Finally, let $s' = (n-1) + (n-2) + \dots + (n-i+1) + s$. Then $\mathbf{v}'[s'] = 1$ and $\mathbf{v}'[j] = 0$ for all $j \neq s'$.

Proof. Notice that the lemma states that \mathbf{v}' will have only one non-zero entry. Since there are only two 1's in \mathbf{v} , the result of the minimum of any two elements in \mathbf{v} will yield 0 except when we are taking the minimum of the two 1's. The exact location of the 1 in \mathbf{v}' can be calculated as follows: Since the first element in \mathbf{v} is a 0, the first $n-1$ elements in \mathbf{v}' will be 0. If the second element is also 0, the next $n-2$ elements in \mathbf{v}' will be zero. This continues until we reach the index i . The minimum of $\mathbf{v}[i]$ with the element $\mathbf{v}[s+i]$ will be the only result equal to 1. The minimum of $\mathbf{v}[i]$ with elements at index less than $s+i$ will again be 0. There will be exactly $s-i-1$ such 0's. The result then follows. \square

We now describe the reduction formally. Suppose the instance of 0-1 IPF is: $A\mathbf{x} = \mathbf{b}$, where A is the $m \times n$ matrix:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

and $\mathbf{b} = (b_1, b_2, \dots, b_m)$. Our transformation is as follows: let $a'_i = \max\{\mathbf{a}_i\}$ be the largest element in \mathbf{a}_i , the i th row of A . Let $n' = n + 1$. Construct m n' -tuples $\mathbf{a}'_i = (a'_i, a_{i,1}, \dots, a_{i,n})$. That is, one for each row of A . Construct $t = n(n-1)/2$ n' -tuples with binary elements, such that each one is a unique n' -tuple with exactly two 1's and the first element fixed at 0. We have:

$$\begin{aligned} &(0, 1, 1, 0, 0, 0, \dots, 0, 0) \\ &(0, 1, 0, 1, 0, 0, \dots, 0, 0) \\ &\vdots \\ &(0, 0, 0, 0, 0, 0, \dots, 1, 1) \end{aligned}$$

This enumeration exhausts all such n' -tuples. Now, given an instance $A\mathbf{x} = \mathbf{b}$ of 0-1 IPF we can transform it into an instance of SUM OF k MINS, by using the n' -tuples constructed above and letting $u_1 = b_1, u_2 = b_2, \dots, u_m = b_m, u_{m+1} = 0, \dots, u_{m+t} = 0$. To see that this transformation gives the desired result, we use Lemmas 5 and 6 to expand the sum of k mins function on these inputs in the form of matrices as in Section 7.3.3. In the following, let $a_{i,j|k,l}$ denote the minimum of $a_{i,j}$ and $a_{k,l}$. Also, let \mathbf{a}'_i denote the i th n' -tuple constructed above corresponding to the i th row of A and $\mathbf{0}$ denote the all-zero n' -tuple. We get:

$$\begin{bmatrix} \mathbf{a}'_1 & a_{1,1|1,2} & a_{1,1|1,3} & \cdots & a_{1,n-1|1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}'_m & a_{m,1|m,2} & a_{m,1|m,3} & \cdots & a_{m,n-1|m,n} \\ \mathbf{0} & 1 & 0 & \cdots & 0 \\ \mathbf{0} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{n+t} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

This shows that a solution to the above matrix equation should have $z_{n+1} = \dots = z_{n+t} = 0$. The assignment to the variables z_1, \dots, z_n then decides whether the 0-1 IPF instance has a solution or not. Therefore, 0-1 IPF has a solution iff SUM OF k MINS has a solution in our transformation.

This transformation is polynomial-time. The first part of the transformation finds the maximum elements in \mathbf{a}_i 's which can be done in time polynomial in the size of the elements of A . The second part introduces $n(n-1)/2$, $(n+1)$ -tuples with binary elements which can straightforwardly be constructed in polynomial time. Thus, 0-1 IPF can be reduced to SUM OF k MINS in polynomial time.

The original problem in [8] is stated with bounded integers (modulo 10 or in general, modulo d). We next show that this reduction can be used to relate the hardness of the original problem to the MODULO d MAXIMUM LIKELIHOOD DECODING problem mentioned in Section 7.3.3. As mentioned before, with $d = 2$ this problem is NP-Complete and W[1]-Hard [44, 61]. We shall show that the problem is also NP-Complete and W[1]-Hard with a general d .³ This implies that the MODULAR SUM OF k MINS problem is also NP-Complete and W[1]-Hard, which shows that this problem is fixed parameter intractable, where the fixed parameter is k . Thus it is unlikely to substantially improve on the algorithms mentioned in Section 7.3.3 for the MODULAR SUM OF k MINS problem.

7.3.5 Modular Sum of k Mins

We can employ the reduction used in the previous section to illustrate hardness of the MODULAR SUM OF k MINS problem. Consider the system of linear equations modulo d in n unknowns:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

represented as $A\mathbf{x} \equiv \mathbf{b} \pmod{d}$, where A is an $m \times n$ matrix with elements from $\{0, \dots, d-1\}$, \mathbf{x} is the unknown binary vector of Hamming weight k and \mathbf{b} is a vector with elements from $\{0, \dots, d-1\}$. If $m > n$, we can use Gaussian elimination to get a unique solution for \mathbf{x} (provided, of course, the m rows of A are linearly independent). However, if $\log_d \binom{n}{k} < m < n$, it gives rise to what we call the MODULO d MAXIMUM LIKELIHOOD DECODING problem⁴, where once again the lower bound on m is the information theoretic lower bound to guarantee a unique solution.

When $d = 2$, the variant of the problem which asks for a binary vector \mathbf{x} with Hamming weight *at most* k was shown to be NP-Complete in [61] and W[1]-Hard in [44]. However, it can easily be seen that this result also applies to the version of the problem with Hamming weight *exactly* k . Thus, we need not consider this distinction

³Although we mentioned in Section 6.6.3 of Chapter 6 that it is shown in [50, Appendix A] that the problem is NP-Complete with a general d , their result does not imply W[1]-Hardness of this problem.

⁴This problem is simply referred to as a sparse subset sum problem in [8].

here. To prove the result for an arbitrary d , we use some of the results from [44]. The following notations and definitions are also taken from [44]. Let $G = (V, E)$ be a graph. A set of vertices $V' \subseteq V$ is said to be a *perfect code* in G , if every vertex of V is either in V' or has a unique neighbour in V' , but not both. The PERFECT CODE problem is described as follows:

Given a graph $G(V, E)$ and a positive integer k , is there a k -vertex perfect code in G ?

This problem is NP-Complete and W[1]-Hard [44]. A *red-blue graph* $G = (R, B, E)$ is a bipartite graph with the partition of vertices into the red set R and the blue set B , where E is the edge set. Let $R' \subseteq R$ be a non-empty subset of red vertices. Then:

- R' is a *dominating set* if every vertex in B (the set of blue vertices) has *at least* one neighbour in R' .
- R' is a *perfect code* if every vertex in B has a *unique* neighbour in R' .

As in [44], the definition above of a perfect code for a red/blue bipartite graph is different from the one for general graphs. However, the distinction will be clear from the context. Central to the reduction is Theorem 1 from [44] reproduced here:

Theorem 3. *Let G be a graph on n vertices, and let k be a positive integer. In time polynomial in n and k we can produce a red/blue bipartite graph G' and a positive integer k' , such that:*

- P1.** *Every dominating set in G' has size at least k' .*
- P2.** *Every dominating set in G' of size k' is a perfect code in G' .*
- P3.** *There is a perfect code of size k in G if and only if there is a perfect code of size k' in G' .*

See [44] for the proof. We can use this theorem to show that MODULO d MAXIMUM LIKELIHOOD DECODING problem is NP-Complete and W[1]-Hard. The reduction is from PERFECT CODE.

Theorem 4. MODULO d MAXIMUM LIKELIHOOD DECODING for $d \geq 2$ is NP-Complete and W[1]-Hard.

Proof. First, notice that it is easy to see that MODULO d MAXIMUM LIKELIHOOD DECODING is in NP. Now, given an instance $G(V, E)$ and k of PERFECT CODE, we construct the red/blue bipartite graph $G' = (R, B, E')$ of Theorem 3. Let $|R| = n'$ and $|B| = m'$. Let A' be the red-blue adjacency matrix of G' . That is, set $R = \{1, \dots, n'\}$ and $B = \{1, \dots, m'\}$, and let $a'_{i,j} = 1$ if and only if $i \in B$ is adjacent to $j \in R$ in G' . Let $\mathbf{1}$ denote the all 1 binary vector of m' elements.

Now, from **P2** of Theorem 3, every solution \mathbf{x}' of Hamming weight k' satisfies $A'\mathbf{x}' = \mathbf{1}$ (or $A'\mathbf{x}' \equiv \mathbf{1} \pmod{d}$). This is true since this property implies that every column of A' corresponding to a dominating set $R' \subseteq R$ in G' of size k' has exactly one

entry set to 1 (because such a dominating set is a perfect code in G'). Thus a solution of Hamming weight k' implies a perfect code in G' of size k' , and vice versa. From **P3** of Theorem 3, this implies that there is a solution \mathbf{x}' of Hamming weight k' if and only if there is a perfect code of size k in G . This completes the transformation. \square

We now state our main result.

Theorem 5. MODULAR SUM OF k MINS is NP-Complete and $W[1]$ -Hard.

Proof. An instance $A\mathbf{x} \equiv \mathbf{b} \pmod d$ of MODULO d MAXIMUM LIKELIHOOD DECODING can be transformed into an instance of the MODULAR SUM OF k MINS problem using the same general reduction described in Section 7.3.4 as follows. We first construct a new matrix V from A , by appending the integer $d - 1$ in front of each row in A and introducing $n(n - 1)/2$ binary $(n + 1)$ -tuples of the form described in the reduction, as $n(n - 1)/2$ new rows in A . The resulting matrix V is as follows:

$$\begin{bmatrix} d-1 & a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ d-1 & a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n-1} & a_{m,n} \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}$$

Next we introduce a binary vector \mathbf{z} of $n + n(n - 1)/2$ binary variables $z_{i,j}$. Finally we create the vector \mathbf{u} of $m + n(n - 1)/2$ elements, whose first m elements are the same as \mathbf{b} and the last $n(n - 1)/2$ elements are all 0's. This constitutes an instance of MODULAR SUM OF k MINS. Now, by expanding the matrix V by taking minimums and calling the resulting matrix V' , we get the system of equations $V'\mathbf{z} \equiv \mathbf{u} \pmod d$, with $n + n(n - 1)/2 = n(n + 1)/2$ variables and $m' = m + n(n - 1)/2$ pairs. Since $m' > n(n + 1)/2$ is the requirement to use Gaussian elimination for this set of equations, we have the restriction on m' :

$$m' < \frac{n(n + 1)}{2}$$

Now suppose, there is an algorithm that can solve the MODULAR SUM OF k MINS problem with $m' < n(n + 1)/2$. From our transformation, we have seen that this also solves the MODULO d MAXIMUM LIKELIHOOD DECODING problem detailed above with $m' = m + n(n - 1)/2 < n(n + 1)/2$ or $m < n(n + 1)/2 - n(n - 1)/2 = n$. Thus, an algorithm for MODULAR SUM OF k MINS can be used to solve the MODULO d MAXIMUM LIKELIHOOD DECODING problem. Finally, note that it is easy to see that this problem is in NP as any candidate solution can be verified in polynomial time by simply carrying out the sum over mins function m times. This can be carried out in time polynomial in the size of the input. Thus, this shows that this problem is both NP-Complete and $W[1]$ -Hard. \square

This result suggests that it is unlikely to substantially improve on the algorithms mentioned in Section 7.3.3, for the MODULAR SUM OF k MINS problem. We like to

reiterate an important point here. While the protocol based on the MODULAR SUM OF k MINS problem uses random instances, these reductions are done on worst-case instances. Thus, these results should not be taken as asserting the security of the protocol.

7.3.6 A Short Digression: The Case when $d = 2$

We digress here briefly to discuss another application of showing the fixed-parameter intractability of problems used as primitives in human identification protocols. It has been suggested that human identification protocols can be modified for use as identification protocols in resource limited devices, such as Radio-Frequency Identification (RFID) tags [55], since these devices share certain attributes with humans, e.g., limited memory and computational power. Thus, the case when $d = 2$ deserves further discussion as this base naturally leads to low cost arithmetic. In [8], Hopper and Blum proposed the base 10 as a natural base for humans. For computational devices, however, base 2 is preferable. Thus, we now assume that computations are done in the field \mathbb{Z}_2 . The group table for the “min” function is shown in Table 7.1. The table shows that the min function over elements in \mathbb{Z}_2 is none other than bit multiplication. Thus, it makes more sense to call the modular sum of mins function as the sum of products function in this case.

In light of this observation, we redefine the MODULAR SUM OF k MINS problem, when $d = 2$. Let n and k be positive integers, where $k = \mathcal{O}(\log_2 n)$. Let $z = \{(x_1, y_1), \dots, (x_k, y_k)\}$ be a set of ordered pairs, where $1 \leq x_i, y_i \leq n$, and for each ordered pair (x_i, y_i) , $x_i \neq y_i$. Let $\mathbf{v} \in \mathbb{Z}_2^n$. Define $f(\mathbf{v}, z)$ as:

$$f(\mathbf{v}, z) = \sum_{i=1}^k \mathbf{v}[x_i] \mathbf{v}[y_i] \bmod 2$$

The SUM OF k PRODUCTS problem is:

Given m ordered pairs $(\mathbf{v}_1, u_1), \dots, (\mathbf{v}_m, u_m)$, find a set z such that $u_i = f(\mathbf{v}_i, z)$ for all $i = 1, \dots, m$, where:

$$\mathbf{v}_i \in \mathbb{Z}_2^n, u_i \in \mathbb{Z}_2, \text{ and } \log_2 \binom{n(n-1)/2}{k} < m < \binom{n}{2}.$$

Then, a protocol similar to the one presented in Section 7.3.2 can be constructed. The memory or space requirement for this protocol is $\mathcal{O}((\log_2 n)^2)$, which is polylogarithmic. The time complexity of computing the above function is linear in n . As mentioned earlier, if $m < \binom{n}{2}$, the best known attacks for these parameters work in $\mathcal{O}(\binom{n(n-1)/4}{k/2})$ time. A further issue that needs to be addressed here is the restriction on the parameter k , the length of the secret. Humans would memorize a sum of k mins secret as ordered pairs of objects. An object could be made up of textual characters or graphical icons, depending on the type of implementation. On the other hand, in computing devices, each secret index will be stored as $\log_2 n$ bits in memory. Thus, it is tempting to remove the logarithmic restriction on k and enable it to be any value between 1 and

TABLE 7.1: The min function in \mathbb{Z}_2 is simply bit multiplication.

min	0	1
0	0	0
1	0	1

$n(n-1)/2$. However, this makes the space requirement quadratic in n , and hence the protocol would not be useful for resource constrained devices.

Typical values of parameters can be $n = 1000$, $k = 20$ and the number of loops required to avoid a random guess attack can be 20. This gives a one in a million chance for a random guess to be successful. The best known attack, mentioned above, requires $\approx 2^{157}$ time and the protocol can be used for more than 24,000 sessions, before Gaussian elimination can be used to obtain the secret. It should be noted that this protocol is only secure against passive adversaries and needs to be modified to prevent active attacks.

A final matter of concern regarding this protocol is the disparity evident from Table 7.1. The bit 0 has the probability $3/4$ to be an output of the product of two bits as opposed to the bit 1 which has probability $1/4$. But, since the results of the products are being summed multiple times, this difference in probability is balanced out. More specifically, let $p(k, b)$ denote the probability that the bit b is obtained as the sum of k products. Then, it is clear from the table that $p(1, 0) = 3/4$ and $p(1, 1) = 1/4$. It can be shown by a simple induction that:

$$p(k, b) = p(k-1, b)p(1, 0) + p(k-1, 1-b)p(1, 1)$$

A dynamic programming algorithm can then be used to compute these probabilities, similar to the one outlined in [8], and described in detail in Section 5.2.5. Define the statistical distance between the uniform distribution and this distribution as:

$$\Delta = \frac{1}{2} \left| \left(\frac{1}{2} - p(k, 0) \right) + \left(\frac{1}{2} - p(k, 1) \right) \right|$$

Then the expected number of samples required to differentiate the two distributions is $1/\Delta$. For $k = 20$, $1/\Delta = 2,097,152$. Thus, it is safe to assume that with the values of parameters mentioned above, an adversary cannot use the difference in the two distributions to find the secret with fewer than $2,097,152/20 \approx 104,857$ observed sessions. This is already higher than the sessions required for Gaussian elimination, which as stated above is 24,000. Thus, care must be taken in choosing values of parameters such that this attack always requires more challenge-response pairs than Gaussian elimination. As a final observation, as illustrated in Section 7.3.5, the MAXIMUM LIKELIHOOD DECODING problem⁵ directly reduces to the SUM OF k PRODUCTS problem, which shows that this problem is both NP-Complete and W[1]-Hard. This implies that it is unlikely to substantially improve on the running time $\mathcal{O}\left(\binom{n(n-1)/4}{k/2}\right)$ to solve the SUM OF k PRODUCTS problem.

⁵i.e., the version which asks for an \mathbf{x} of Hamming weight exactly k , with $d = 2$.

7.4 HB and the Example Protocol

We now return to the problem k -LPN, the underlying problem of the HB Protocol described in Section 7.1. This is a straightforward reduction from the MAXIMUM LIKELIHOOD DECODING problem mentioned in the previous section. Given an instance of MAXIMUM LIKELIHOOD DECODING we can reduce it to k -LPN by simply setting $\epsilon = 0$. Thus, k -LPN is both NP-Hard and W[1]-Hard.

It is important to note that MAXIMUM LIKELIHOOD DECODING is polynomial time solvable if $m \geq n$ and if A contains n linearly independent rows. In such a case, Gaussian elimination can be used to efficiently find an \mathbf{x} of Hamming weight k . It is for this reason, that a protocol based on this problem can only be used before the adversary has observed $\mathcal{O}(n)$ challenge-response pairs. In fact, the Example Protocol from Chapter 2 is based on this exact problem. On the other hand, the HB protocol tries to stretch the required number of observed sessions by the adversary to at least quadratic in n by introducing noise into the system of linear equations. However, the reduction above does not shed light on this “extension”.

The reduction used for the MODULAR SUM OF k MINS problem showed that if an algorithm exists to solve this problem with $m < \mathcal{O}(n^2)$, then the same algorithm can solve the MODULO d MAXIMUM LIKELIHOOD DECODING problem with $m < \mathcal{O}(n)$. It would be nice to have a similar reduction for k -LPN, highlighting the number of observed sessions required to find the secret in the HB Protocol. This is an interesting open problem.

7.5 The Counting Edges Protocol

We restate the COMMON VERTEX SUBSET problem, used in the construction of the basic Counting Edges Protocol.

Given m pairs $(G_1, r_1), \dots, (G_m, r_m)$, where G_i is a graph on a vertex set V of n vertices, find a k -element subset K of V such that the induced subgraphs $G_i[K]$ have r_i edges, for $1 \leq i \leq m$.

For any $m > 1$, we call this problem, COMMON VERTEX SUBSET. With $m = 1$ this is related to the HEAVIEST UNWEIGHTED SUBGRAPH⁶ problem which is NP-Hard [64]. This can be shown with a reduction from the CLIQUE problem. The parameterized version of the CLIQUE problem, k -CLIQUE is W[1]-Complete [65]. k -CLIQUE is the problem of finding a clique of size k in a graph. A clique of size k is a complete subgraph of k vertices. We show here that COMMON VERTEX SUBSET is W[1]-Hard, and hence not fixed-parameter tractable, by a reduction from k -CLIQUE.

Theorem 6. COMMON VERTEX SUBSET is W[1]-Hard.

Proof. Given an instance of k -CLIQUE, (G, k) , where G is a graph on a vertex set V of n vertices, we transform it to an instance of COMMON VERTEX SUBSET as follows.

⁶also known as the DENSEST k -SUBGRAPH problem [63].

We set $G_1 = G$ and $r_1 = \binom{k}{2}$. For the remaining pairs, $(G_2, r_2), \dots, (G_m, r_m)$, we set each G_i to be an edgeless graph on V , and set $r_i = 0$. \square

Note that this reduction can easily be used to show that even with $m = 1$ the problem is $W[1]$ -Hard (which corresponds to the parameterized version of HEAVIEST UNWEIGHTED SUBGRAPH). The problem used in the main Counting Edges Protocol is different from COMMON VERTEX SUBSET, as in the main protocol the user \mathcal{H} returns an edge count modulo an integer d . However, it can be easily shown that this variant is also $W[1]$ -Hard, simply by choosing $d = \binom{k}{2} + 1$.

7.6 The Foxtail Protocol

The Foxtail Protocol from Li and Shum [14] is a human identification protocol that also tries to increase the number of authentication sessions a secret can be used to at least quadratic in n . We show here, that the underlying problem of this protocol is also fixed-parameter intractable. Central to the protocol is the *foxtail* function, which is computed as follows:

$$r = \text{ft}(\mathbf{c}, \mathbf{x}) = \left\lfloor \frac{\mathbf{c} \cdot \mathbf{x}^T \bmod 4}{2} \right\rfloor$$

where \mathbf{c} and \mathbf{x} are binary vectors of n elements, and \mathbf{x} has Hamming weight k . Intuitively, $\text{ft}(\cdot, \cdot)$ is a function that first maps the result of the dot product of two binary vectors to the range $\{0, 1, 2, 3\}$, and then maps 0 and 1 to 0, and 2 and 3 to 1, which means that $r \in \{0, 1\}$. Thus, the protocol is designed such that the adversary does not know the result of the dot product of the challenge and the secret vector, and therefore cannot represent the gathered challenge-response pairs as a system of linear equations with a non-negligible probability. The protocol is described here for reference [14].

Protocol: Foxtail.

Setup: Let μ , n and $k \leq n$ be publicly known positive integers. \mathcal{H} and \mathcal{C} share a random n -element binary vector \mathbf{x} of Hamming weight k as a secret. Also let $l \geq 3$ be a publicly known positive integer.

- 1: **for** $i = 1$ to μ **do**
- 2: \mathcal{C} generates two n -element vectors $\mathbf{c}_{i,1}$ and $\mathbf{c}_{i,2}$ with $\text{wt}(\mathbf{c}_{i,1}) = \text{wt}(\mathbf{c}_{i,2}) = l$. $\mathbf{c}_{i,1}$ is generated randomly from the space of all n -bit binary vectors of Hamming weight l . $\mathbf{c}_{i,2}$ is chosen such that $\mathbf{c}_{i,2} \cdot \mathbf{x}^T = i$ with probability $1/4$, for $i \in \{0, 1, 2, 3\}$. \mathcal{C} sends these vectors to \mathcal{H} .
- 3: \mathcal{H} sends the following response to \mathcal{C} ,

$$r = \text{ft}(\mathbf{c}_{i,1} + \mathbf{c}_{i,2}, \mathbf{x}) = \left\lfloor \frac{(\mathbf{c}_{i,1} \cdot \mathbf{x}^T + \mathbf{c}_{i,2} \cdot \mathbf{x}^T) \bmod 4}{2} \right\rfloor$$

- 4: **if** all μ responses are correct, \mathcal{C} accepts \mathcal{H} **else** \mathcal{C} rejects \mathcal{H} .

Intuitively, the parameter l is used to ensure sparsity. That is, if the Foxtail Protocol is implemented in a way similar to the Example Protocol (cf. Chapter 2), the parameter l ensures that no more than $2l$ icons are present in the grid. The challenge vectors are divided into two sub-challenges $\mathbf{c}_{i,1}$ and $\mathbf{c}_{i,2}$, each generated from a different probability distribution. This is to ensure that the secret and the non-secret icons appear in the challenge with equal probability. See [14] for details.

We first describe a simplified problem, called FOXTAIL DECODING, as follows:

Given m vectors $\mathbf{c}_i \in \mathbb{Z}_2^n$ and an m -element binary vector \mathbf{r} , is there an \mathbf{x} with $\text{wt}(\mathbf{x}) = k$, such that $\text{ft}(\mathbf{c}_i, \mathbf{x}) = r_i$?

We show that this problem is NP-Complete and W[1]-Hard by reducing the following problem to it:

Given an $m \times n$ matrix C of binary elements, and an m -element binary vector \mathbf{r} , is there an \mathbf{x} with $\text{wt}(\mathbf{x}) = k$, such that $C\mathbf{x} \equiv \mathbf{r} \pmod{4}$?

We call this problem SPARSE DECODING. The problem used in the protocol above is slightly different from FOXTAIL DECODING. Specifically, instead of having a binary matrix C ,⁷ we have a matrix C whose elements are from $\{0, 1, 2\}$. To see this, observe that the two vectors $\mathbf{c}_{i,1}$ and $\mathbf{c}_{i,2}$ in each challenge i can have overlapping entries set to 1. It can easily be seen that this problem is at least as hard as the FOXTAIL DECODING problem, since an algorithm to solve this problem can also solve the case when C is binary.

Theorem 7. SPARSE DECODING is NP-Complete and W[1]-Hard.

Proof. The proof is the same as the proof of Theorem 4, except for a minor difference. Since, the result $A'\mathbf{x}' = \mathbf{1}$ holds with or without a modulus d in Theorem 4, we see that this also holds for $A'\mathbf{x}' \equiv \mathbf{1} \pmod{4}$. \square

The main result for FOXTAIL DECODING is as follows.

Theorem 8. FOXTAIL DECODING is NP-Complete and W[1]-Hard.

Proof. Given an instance of SPARSE DECODING, $C\mathbf{x} \equiv \mathbf{r} \pmod{4}$, we convert it into an instance of FOXTAIL DECODING as follows. From the matrix C with elements $c_{i,j}$, we create a new matrix C' , and from the vector \mathbf{r} we create a new vector \mathbf{r}' . For $1 \leq i \leq m$, the construction is as follows. If $r_i = 0$, we create the following new rows in C' and \mathbf{r}' :

$$\begin{bmatrix} c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 0 & 0 \\ c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 1 & 0 \end{bmatrix} \in C', \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbf{r}'$$

if $r_i = 1$:

$$\begin{bmatrix} c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 0 & 0 \\ c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 1 & 0 \end{bmatrix} \in C', \begin{bmatrix} 0 \\ 1 \end{bmatrix} \in \mathbf{r}'$$

⁷Composed of the m vectors $\mathbf{c}_i \in \mathbb{Z}_2^n$.

if $r_i = 2$:

$$\begin{bmatrix} c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 0 & 0 \\ c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 1 & 0 \end{bmatrix} \in C', \begin{bmatrix} 1 \\ 1 \end{bmatrix} \in \mathbf{r}'$$

and if $r_i = 3$:

$$\begin{bmatrix} c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 0 & 0 \\ c_{i,1} & c_{i,2} & \cdots & c_{i,n} & 1 & 0 \end{bmatrix} \in C', \begin{bmatrix} 1 \\ 0 \end{bmatrix} \in \mathbf{r}'$$

Finally we complete the construction of C' and \mathbf{r}' with the following two rows,

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} \in C', \begin{bmatrix} 0 \\ 1 \end{bmatrix} \in \mathbf{r}'$$

Then, we have a matrix C' with $2m + 2$ rows and $n + 2$ columns, and a vector \mathbf{r}' of $m + 2$ entries. Let \mathbf{x}' be an unknown $(n + 2)$ -bit vector with Hamming weight $k + 2$. Then, we can see that a solution to $\text{ft}(C', \mathbf{x}') = \mathbf{r}'$ is a solution to the original system of linear equations $C\mathbf{x} \equiv \mathbf{r} \pmod{4}$ and vice versa. This completes the proof. \square

Just as in the case of k -LPN, the above reduction does not reflect on the number of observed sessions the Foxtail Protocol can be used. It would be nice to have a reduction from a problem that shows fixed-parameter intractability of k -FOXTAIL DECODING even if the matrix C has $m > n$ rows out of which n are linearly independent. This remains an open problem. However, like the HB Protocol, we can show that the Foxtail Protocol is based on a hard to learn problem, i.e., learning with errors.

Learning with Errors Formulation

Suppose C is an $m \times n$ matrix with entries from $\{0, 1, 2\}$. Then, we can formulate the foxtail function as:

$$C\mathbf{x} \equiv 2\mathbf{r} + \mathbf{e} \pmod{4}$$

where, \mathbf{r} is the m -element binary response vector from the Foxtail Protocol and \mathbf{e} is a random m -element binary vector. Rearranging, we get:

$$\begin{aligned} C\mathbf{x} - \mathbf{e} &\equiv 2\mathbf{r} \pmod{4} \\ \Rightarrow C\mathbf{x} + 3\mathbf{e} &\equiv 2\mathbf{r} \pmod{4} \\ \Rightarrow C\mathbf{x} + \mathbf{e}' &\equiv \mathbf{r}' \pmod{4} \end{aligned} \tag{7.1}$$

where above, $\mathbf{e}' = 3\mathbf{e}$ and $\mathbf{r}' = 2\mathbf{r}$. Presented in this way, the problem can be considered as a LEARNING WITH ERRORS problem [66] (or LWE in short) over \mathbb{Z}_4^n . Here \mathbf{e}' is an error vector generated with the probability distribution χ :

$$\chi(0) = \frac{1}{2}, \chi(1) = 0, \chi(2) = 0, \chi(3) = \frac{1}{2}$$

It should be noted that this version of LWE is over the ring \mathbb{Z}_4 . Nevertheless, there is evidence that LWE over rings is also hard [67]. In any case, learning with errors requires a polynomial in n number of samples. Thus, we can conjecture that a solution of Equation 7.1 requires $m > \text{poly}(n)$. In fact we can assume that the best known algorithm requires $2^{\mathcal{O}(n)}$ equations [66, p. 2]. It remains an interesting open problem to evaluate the hardness of the problem when $\text{wt}(\mathbf{x}) = k$ and C is sparse.

7.7 Conclusion

We have shown that several human identification protocols in literature, as well as the Counting Edges Protocol from Chapter 6, are based on fixed-parameter intractable problems. This gives us evidence that the best possible algorithms to solve the underlying problems run in time $n^{f(k)}$. Thus, it is very likely that the best possible attacks on these protocols are the time-memory tradeoff algorithms from [8] and Chapter 5. Central to many reductions in this chapter has been the MODULO d MAXIMUM LIKELIHOOD DECODING problem. As we have seen, a protocol based on this problem can only be used for $\mathcal{O}(n)$ observed challenge-response pairs before secret renewal. An example is the Example Protocol from Chapter 2. Yet, the problem is fixed-parameter intractable. Thus, fixed-parameter intractability only tells us that there are instances of a problem that are hard. In this case, when $m < n$. If there are n linearly independent rows in an instance of the MODULO d MAXIMUM LIKELIHOOD DECODING problem, then we have an efficient algorithm, namely Gaussian elimination, that can be used to solve the problem.

However, in showing the fixed-parameter intractability of the MODULAR SUM OF k MINS problem, we found a nice result that an algorithm to solve this problem with $m < n(n-1)/2$ can solve the MODULO d MAXIMUM LIKELIHOOD DECODING problem with $m < n$. Thus, this gives some evidence that there is no efficient (in the notion of fixed-parameter tractability) attack on the Sum of k Mins Protocol when only $m < \mathcal{O}(n^2)$ challenge-response pairs are observed. Unfortunately, the other reductions did not shed light on this important parameter in human identification protocols. However, in such cases, evidence of the hardness of the problem is gathered from elsewhere. For instance, we showed that the underlying problem used in the Foxtail Protocol from [14], can be viewed as some sort of LEARNING WITH ERRORS problem. In case of the Counting Edges Protocol, the underlying problem is shown to be fixed-parameter intractable with a reduction from a fixed-parameter intractable problem from graph theory, namely k -CLIQUE. This gives us evidence that the best way to attack the protocol is through the adjacency matrix formulation (cf. Section 6.4), but this expands the problem into quadratic in n , meaning the protocol can be used for $\mathcal{O}(n^2)$ sessions before secret renewal. It would have been nice to find a reduction for the underlying problem used in the Kangaroo Hopping Protocol from Chapter 5. However, it is not clear how this can be done. Nevertheless, we can see that the protocol attempts to avoid the underlying problem from being represented as a system of linear equations with a non-negligible probability. Thus, in some sense, it can be thought of as being based on some sort of LEARNING WITH ERRORS problem, just as the Foxtail Protocol from [14].

8

Conclusion and Future Research Directions

This thesis has primarily focused on the analytical aspects of human identification protocols. We have shown that without a comprehensive security analysis, in particular without properly analyzing the underlying problem used in the construction of human identification protocols, it is possible to break the protocols through simple but innovative attacks, and obtain the secret after observing only a handful of authentication sessions. This is true even if the protocol is meant to be used for a small number of sessions. We have demonstrated this by analyzing and breaking two protocols from literature. Namely, the Predicate-based Authentication Service (PAS) [17] and the Convex Hull Click (CHC) [21] protocols. The security analysis of these protocols shows that it is important to first represent the problem of an adversary in terms of solving a mathematical problem, and then analyze how the information about the secret can be leaked after the observation of one or more challenge-response pairs. As such, both information theoretic and computational security aspects have to be taken into account.

A parallel theme in this thesis has been the search for protocols which can be used for at least $\mathcal{O}(n^2)$ challenge-response pairs before secret renewal; an increase from $\mathcal{O}(n)$ challenge-response pairs offered by, for instance, Matsumoto's protocols from [19]. Protocols from Hopper and Blum [8], and Li and Shum [14] are previous examples of protocols that achieve this quadratic increase (or more). The Kangaroo Hopping Protocol in Chapter 5 was our first attempt towards this end. The idea behind the design was to avoid the underlying problem from being represented as a system of linear equations in n unknowns with a non-negligible probability. This prevents the adversary from finding the secret after observing n challenge-response pairs through Gaussian elimination. We then proceeded to the construction of the Counting Edges Protocol in Chapter 6. The intent was to construct a protocol based on a problem that can be considered hard in the relatively new complexity theoretic sense of fixed-parameter intractability. Since in general, the shared secret in human identification

protocols is a set of k objects from a large pool of n objects, these protocols have an inherent additional parameter k (other than n). Therefore, fixed-parameter intractable problems are natural candidates as primitives for such protocols. We have argued that if the primitive used to construct a protocol is fixed-parameter intractable, it provides strong evidence that breaking the protocol will be hard in practice. We showed that many existing human identification protocols are in fact based on fixed-parameter intractable problems, without the inventors of these protocols realizing the connection. In short, the main contributions of this thesis can be summarized as follows.

- We showed a probabilistic attack on Bai et al.’s Predicate-based Authentication Service (PAS) from [17] which demonstrates the insecurity of the protocol. In particular, the adversary has a 50% chance of finding the secret after observing a mere 7 observed sessions. Even with only 2 observed sessions, the adversary has a non-negligible, 3.5%, chance of finding the secret. These numbers are considerably lower than the figure of 10 authentication sessions suggested by Bai et al. for the use of their protocol, before secret renewal.
- We showed a probabilistic attack (different from above) on the Convex Hull Click (CHC) Protocol from Sobrado and Birget [9, 21]. With the help of this attack, the adversary has a 15% chance of impersonating the user, after observing only 10 authentication sessions. These figures are for values of protocol parameters suggested by Weidenbeck et al. for high security in [9].
- We proposed a protocol, named the Kangaroo Hopping Protocol, which is designed to be used for at least $\mathcal{O}(n^2)$ authentication sessions before secret renewal. The protocol is designed so that the responses are non-linearly dependent on the challenges. As a result, the underlying problem cannot be represented as a system of linear equations in n unknown. This prevents the use of Gaussian elimination from being used to find the secret after the observation of $\mathcal{O}(n)$ sessions. With the recommended values of protocol parameters, we conjecture that the protocol can be used for about 2,000 sessions before secret renewal.
- We gave the construction of a protocol, named the Counting Edges Protocol, which is based on a fixed-parameter intractable problem. We also conjecture that this protocol can be used for $\mathcal{O}(n^2)$ authentication sessions before secret renewal through a comprehensive security analysis. Fixed-parameter intractability of the underlying problem gives us some evidence that finding the secret from the observation of a small number of challenge-response pairs is not straightforward. We suggest the use of the protocol for more than 1,500 authentication sessions before secret renewal.
- We argued that since human identification protocols involve a secret of k objects out of a total of n , fixed-parameter intractable problems are natural candidates for use as primitives for these protocols, since these problems have a natural parameter k . Thus, proving the fixed-parameter intractability of the underlying problem used in a human identification protocol gives strong evidence for security

of the protocol. We showed that many existing protocols in literature are based on fixed-parameter intractable problems.

However, we do not claim that the treatment of the subject of human identification protocols in this thesis is comprehensive. We mentioned in the introduction that many different aspects need to be considered when designing such protocols. Our treatment of the subject has focused on the analytical aspects, i.e., security analysis of the protocols by clearly representing the problem of the adversary in mathematical terms. In practice, however, the security of such protocols should also consider human behavior while executing these protocols. Furthermore, comprehensive usability studies have to be carried out to determine the feasibility of these protocols. Such studies can lead to improved and innovative ways of implementing these protocols. In the following, we highlight this as well as some other future research directions in the area of human identification protocols.

Side Channel Attacks

Side channel attacks try to extract information about secrets used in cryptosystems by analyzing the side channels that are available when such systems are implemented. For instance, the attacker can analyze the power consumption during a block cipher's computation on a message [68], and if there is a difference in power consumption during the processing of 0 and 1 bits, the attacker can know which of the two bits a particular point in the *trace* of the power consumption corresponds to [68]. Similarly, by examining the time difference in computing cryptographic operations, the adversary can again gain knowledge of the secret through what is known as the timing attack [69].

In a similar sense, side channel attacks can also be mounted on human identification protocols. In Section 2.5, we mentioned the Undercover system from [18], which is different from the human identification protocol considered in this thesis as it uses a separate “secure” channel. The channel is realized through a haptic device, on which the user rests his palm. The user then gets part of the challenge through touch sensations on his palm. This part of the challenge is assumed to be hidden from the observation of the adversary. However, it was shown in [28] that the system can be broken with high probability after the observation of only about 10 authentication sessions, through observing the user behavior.

Similarly, for the human identification protocols purported to be secure under Matsumoto and Imai's threat model, information can be leaked by observing the user's behavior during the execution of these protocols. For instance, if the user points at one of the icons displayed on the screen, the adversary can guess that with high probability it is one of the user's secret icons. Furthermore, if a reduced number of icons are present in a challenge, then the adversary can gain information by examining the time required to respond to each challenge; less time implies less number of secret icons present in a challenge. The two proposed protocols in this thesis do not seem to be vulnerable to this type of attack. Still however, user training is required to inform the users about how information about the secret can be leaked due to an insecure

interaction with the terminal. In short, the security of human identification protocols should also be analyzed against these side channel attacks.

Variants of Matsumoto and Imai's Threat Model

Matsumoto [19] suggested that different variants of the Matsumoto and Imai's threat model can be considered, for example by assuming that only one of the two channels, one through which a challenge is sent from the server and the other through which the user sends his response, is insecure. Such variants are not completely unrealistic, since the Undercover system described above utilizes a similar assumption and realizes it with a haptic device. Thus, depending on the environment, one can assume different variants of the aforementioned threat model to be applicable. Several other authentication systems have used different variants of Matsumoto and Imai's threat model. For instance, one variant is the model that assumes the user terminal to be secure in the sense that its internal computations cannot be viewed by the adversary, but he can still observe the display. This assumption has been used in many shoulder-surfing resistant authentication schemes. However, it is not clear how this assumption can be used to help a user in his computations, as his interaction with the terminal is largely through the display, and this information can also be viewed by the adversary. Due to this reason, from a theoretical viewpoint, we didn't consider this a different threat model in this thesis. Nevertheless, different variants of Matsumoto and Imai's threat model can be considered for different scenarios, and the protocols can potentially take advantage of the additional secure links at disposal.

Another example of a different threat model is a model that assumes that the adversary cannot observe more than a fixed number m_{seq} of consecutive challenge-response pairs. A protocol can then be constructed that takes advantage of the fact that the adversary has not observed at least one among $m_{\text{seq}} + 1$ consecutive challenge-response pairs. For instance, some additional information can be shown to a user at the end of each challenge-response pair (or authentication session) without which the adversary cannot learn the secret. One possibility is to show a new picture (or icon) at the end of a session, which the user replaces with one of his secret pictures. Of course, user studies have to be carried out to establish the feasibility of this method, in particular to find if it is at all easy for humans to recall constantly renewed information. Nonetheless, this is only a simple example of how a protocol can be constructed under this threat model. We would like to emphasize that this model is not unrealistic. If we take the example of ATM authentication, it is reasonable to assume that an attacker cannot mount hidden cameras in all the ATMs in a considerably large geographical area. Thus, a mobile user is bound to use different ATMs, and with high probability one of the used ATMs can be hidden camera free. Note that this threat model is not unprecedented, and has been employed in authentication protocols for RFID tags [70].

Help from Artificial Intelligence

A few protocols in literature are based on problems that can be considered hard from an artificial intelligence perspective. The protocol from [15] is an example. It uses a

secret question which has a binary answer when applied to an image. For instance, the secret question can be: “Does the picture contain any animals?” The protocol is an attempt to extend the idea of CAPTCHAs [26] for use in human identification protocols. The security of the protocol is based on the conjecture that it is hard for an automated adversary to obtain the secret since it cannot obtain a set of common features from a given set of images without a non-negligible error. Moreover, it is conjectured that the problem is also hard for a human without the knowledge of the secret question. This second claim does not seem easy to justify. We can perhaps, extend the idea of CAPTCHAs in a different way so that both human and automated adversaries are dependent on each other to obtain the secret.

A naive way to construct a CAPTCHA-based human identification protocol is as follows. The human user and the computer server share a secret which is a set of k objects out of n . A challenge is composed of n cells, where each cell contains a unique object from the pool of n objects, and a text-based CAPTCHA (e.g., reCAPTCHA [71]). The user decodes each CAPTCHA corresponding to his k secret objects, and sends the resulting text response to the server. Now an automated adversary who observes a challenge-response pair has a negligible probability of decoding each CAPTCHA. The adversary inevitably requires human assistance. Since, there are n cells, this would require $\mathcal{O}(n)$ human (adversarial) computational time. We can safely assume that a human takes at least 1 second to decode each CAPTCHA. Then if $n = 200$, this means at least 200 seconds of human computational time. Although this cannot be considered computationally infeasible, the time required is many orders of magnitude higher than what an automated adversary would have consumed if CAPTCHAs had not been used. This approach has been used before in some protocols. For example, the response table in PAS [17] contains CAPTCHAs in each cell (cf. Chapter 3). However, this is done only to avoid automated processing of responses, and does not provide sufficient security against human adversaries. Thus, one would like to increase human adversaries’ computational time.

One way to increase (adversarial) human computational time, is to construct some form of contrived responses from the decoded CAPTCHAs as is done in the authentication scheme from [72]. In a nutshell, the user only sends part of the decoded text from the CAPTCHA as the response. However, this only has the ability to increase the time linearly in the number of the challenges. Here, we outline a different approach. We define a multivariate CAPTCHA as a function which takes k objects as argument and maps them to a response string $\{0,1\}^*$, such that it is hard for the automated adversary to decode the CAPTCHA given the k objects, but easy for humans. An important requirement of such a function is that it should not be possible to obtain its output by concatenating the output of functions with fewer than k arguments. In other words, the multivariate CAPTCHA should not be decomposable. Otherwise, a naive way to construct a multivariate CAPTCHA is to concatenate the answers of two *single input* CAPTCHAs. However, this violates the criterion just mentioned.

If multivariate CAPTCHAs can be constructed, it will increase the time of the adversary even with a very small size of the user’s secret. For instance, if a 4-variable CAPTCHA function is conceivable, then it would require $\binom{n}{4}$ (adversarial) human computations. This corresponds to about 2 years of human computational time, if

$n = 200$. On the other hand, the legitimate human user has to merely remember 4 secret locations and compute a 4-input CAPTCHA function. If the response is from a large response space, then the total time required by a legitimate user is low. Of course, stating a mere definition does not imply the existence of multivariate CAPTCHAs. A possible instantiation of a multivariate CAPTCHA can be realized as follows. In the setup phase, the user is shown a video clip. During authentication, the challenge grid sent to the user contains randomly scattered snapshots from this video taken at different intervals. The user is asked to output the original sequence in which the snapshots appear in the video corresponding to his secret locations. Of course, this assumes that the video shown to the user remain hidden from the adversary. Perhaps, this is the price that has to be paid for such constructions. In any event, artificial intelligence problems offer some potential for use in human identification protocols.

Pervasive Devices

In [55], Juels and Weis suggested that human identification protocols can be modified for use in low cost pervasive devices, since these devices have certain characteristics in common with humans; restricted memory and computational abilities. They modified the HB Protocol from [8], for use in the authentication of RFID tags [55]. The modified protocol also considers security against active adversaries. Since then, many variants of the HB Protocol have been proposed as candidates for lightweight authentication protocols for resource constrained devices [73]. In Chapter 7 we demonstrated how a variant of the Sum of k Mins Protocol can be used as a protocol for pervasive devices. We also showed that this protocol, the Sum of k Products Protocol, is based on a fixed-parameter intractable problem. However, there were two main short-comings. First, the protocol only considers passive adversaries, and secondly, the reduction to the fixed-parameter intractable problem only demonstrates the hardness of worst case instances. Thus it is an interesting area of research to improve on the Sum of k Products Protocol to incorporate security against active adversaries, as well as analyze the average case hardness of the underlying problem. Compounding on this, more fixed-parameter problems can also be found for use as primitives in human identification protocols, and in authentication protocols for resource constrained devices.

Improving Implementation

An interesting area of research that also touches the area of Human-Computer Interaction (HCI), is to improve the implementation of human identification protocols. We have argued that it is hard to display more than (say) 200 icons in a challenge due to the restricted size of the screen at the user's terminal. However, various techniques can be used to increase this number. For instance, each cell in a challenge can be *programmed* to display a different icon after a fixed duration of time, say 3 seconds. This is done for each challenge, until the user has responded to the challenge. This increases the size of the challenge to 600, without requiring a larger display. Of course, there can be some usability issues with such a method, but it shows that further improvement in

the way the user interacts with the terminal can increase the usability of such protocols. Recent advances in authentication methods have used increasingly sophisticated and innovative ways for the user to interact with the terminal. Some examples are described in [74]. These designs can also realize secure channels. For instance, the Pressure-Grid scheme from [74] realizes a secure channel by virtue of the user applying minute pressure on a touch screen, undetectable by the adversary. Recently, Drucker showed an interesting way to use images to mentally compute multiplication of multi-digit numbers [75]. The images are used to aid the human in computations much like the implementation of the protocols described in this thesis. The techniques illustrated by Drucker are not limited to addition and multiplication, and can be extended to help a human compute other logical operations [75]. These techniques can then be used to improve implementation of human identification protocols which involve more sophisticated functions.

Small Number of Authentications

A negative aspect of human identification protocols secure under Matsumoto and Imai's threat model is that despite incremental improvement in their design, the time consumed per authentication session is still unacceptable for widespread use. An average time of about 3 minutes seems to be the best achievable for such human identification protocols. On the positive side, these protocols can be used for a larger number of authentication sessions before secret renewal. Perhaps, it might be possible to improve on this running time by constructing protocols that are secure for a smaller number of authentication sessions, say around 20. Then an authentication time of around 20 to 30 seconds might be achievable; still high, but a time that users might be ready to compromise in return for added security. We have seen that some protocols in literature were designed for a small number of authentication sessions. However, the number of authentication sessions before secret renewal in these protocols is either too low, 3 to 4 [32] (in terms of security against automated shoulder-surfers), or the protocols' steps are too computationally intensive for a protocol that can only be used for a small number of authentication sessions [17]. Thus, while this appears to be an easier target, the jury is still out to decide on an acceptable solution.



Appendix

A.1 Turk's Method of Generating a Random Point Inside a Triangle

Let A , B and C be the vertices of a triangle. Let s and t be uniform random real numbers in the interval $[0, 1]$. Turk's method generates a random point P contained in the triangle as follows [37].

Algorithm: Turk's Method.

Input: Vertices A , B and C and the random numbers s and t .

Output: Random point P contained in the triangle ΔABC .

- 1: **if** $s + t > 1$ **then**
- 2: $s \leftarrow 1 - s$.
- 3: $t \leftarrow 1 - t$.
- 4: $a \leftarrow 1 - s - t$.
- 5: $b \leftarrow s$.
- 6: $c \leftarrow t$.
- 7: Output $P \leftarrow aA + bB + cC$.

A.2 Optimum Value of m

We have:

$$\frac{n^{m+1}}{d^m} \binom{n/2 + k/2 - 1}{k/2} + \frac{\binom{n+k-1}{k}}{d^m}$$

Let $C_1 = \binom{n+k-1}{k}$ and $C_2 = \binom{n/2+k/2-1}{k/2}$. Taking the first derivative test to find the minimum:

$$\begin{aligned}
n C_2 \frac{d}{dm} \left(\frac{n}{d} \right)^m + C_1 \frac{d}{dm} \left(\frac{1}{d^m} \right) &= 0 \\
\Rightarrow n C_2 \left(\frac{n}{d} \right)^m \ln \left(\frac{n}{d} \right) + C_1 \left(\frac{1}{d^m} \right) \ln \left(\frac{1}{d} \right) &= 0 \\
\Rightarrow n C_2 \left(\frac{n}{d} \right)^m \ln \left(\frac{n}{d} \right) &= \frac{C_1}{d^m} \ln d \\
\Rightarrow n^{m+1} &= \frac{C_1 \ln d}{C_2 \ln(n/d)} \\
\Rightarrow m &= \frac{\ln \left(\frac{C_1 \ln d}{C_2 \ln(n/d)} \right)}{\ln n} - 1 \\
\Rightarrow m &= \frac{\ln \left(\frac{\binom{n+k-1}{k} \ln d}{\binom{n/2+k/2-1}{k/2} \ln(n/d)} \right)}{\ln n} - 1
\end{aligned} \tag{A.1}$$

Let,

$$A = \frac{\binom{n+k-1}{k} \ln d}{\binom{n/2+k/2-1}{k/2} \ln(n/d)} \tag{A.2}$$

Then putting the value of A from Equation (A.2) into Equation (A.1), we get:

$$m = \frac{\ln A}{\ln n} - 1 \tag{A.3}$$

Also re-arranging Equation (A.3) gives us:

$$n^{m+1} = A \tag{A.4}$$

Now let,

$$\begin{aligned}
W &= \frac{n^{m+1}}{d^m} C_2 + \frac{C_1}{d^m} \\
\Rightarrow d^m W &= n^{m+1} C_2 + C_1
\end{aligned} \tag{A.5}$$

Putting the value of n^{m+1} from Equation (A.4) into Equation (A.5):

$$d^m W = A C_2 + C_1$$

Now, putting the value of A from Equation (A.2), we have:

$$\begin{aligned}
d^m W &= \frac{C_1 \ln d}{C_2 \ln(n/d)} C_2 + C_1 \\
\Rightarrow d^m W &= C_1 \left(\frac{\ln d}{\ln(n/d)} + 1 \right) \\
\Rightarrow d^m W &= C_1 \left(\frac{\ln d + \ln(n/d)}{\ln(n/d)} \right) \\
\Rightarrow d^m W &= C_1 \left(\frac{\ln n}{\ln(n/d)} \right)
\end{aligned} \tag{A.6}$$

Let,

$$D = \frac{\ln n}{\ln(n/d)} \quad (\text{A.7})$$

Then Equation (A.6) becomes:

$$\begin{aligned} d^m W &= C_1 D \\ \Rightarrow m \ln d + \ln W &= \ln C_1 + \ln D \\ \Rightarrow \ln W &= \ln C_1 + \ln D - m \ln d \end{aligned} \quad (\text{A.8})$$

Putting in the value of m from Equation (A.3) into Equation (A.8):

$$\begin{aligned} \ln W &= \ln C_1 + \ln D - \left(\frac{\ln A}{\ln n} - 1 \right) \ln d \\ \Rightarrow \ln W &= \ln C_1 + \ln D - \frac{\ln A}{\ln n} \ln d + \ln d \\ \Rightarrow W &= C_1 D (e^{-\ln A})^{\ln d / \ln n} d \\ \Rightarrow W &= C_1 D d A^{-\ln d / \ln n} \end{aligned} \quad (\text{A.9})$$

Now, putting the values of D and A from Equations (A.7) and (A.2) respectively into Equation (A.9), we get:

$$\begin{aligned} W &= C_1 \cdot \frac{\ln n}{\ln(n/d)} \cdot d \cdot \left(\frac{C_1 \ln d}{C_2 \ln(n/d)} \right)^{-\ln d / \ln n} \\ \Rightarrow W &= \frac{\ln n}{\ln(n/d)} \left(\frac{\ln d}{\ln(n/d)} \right)^{-\ln d / \ln n} d C_1^{1-\ln d / \ln n} C_2^{\ln d / \ln n} \end{aligned}$$

And by neglecting the logarithmic terms in the product:

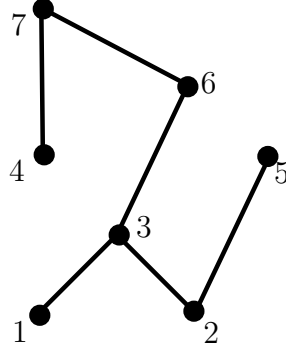
$$W = \mathcal{O} \left(d C_1^{1-\ln d / \ln n} C_2^{\ln d / \ln n} \right)$$

Assuming d to be small, and putting in the values of C_1 and C_2 , we see that the total computational time required is:

$$\mathcal{O} \left(\binom{n+k-1}{k}^{1-\ln d / \ln n} \binom{n/2+k/2-1}{k}^{\ln d / \ln n} \right) \quad (\text{A.10})$$

A.3 Graphs

The elementary concepts of graphs described here are taken from [45]. A graph G is a pair (V, E) of a vertex set V and an edge set E , where E is a set of two-element subsets of V . Elements of V and E are called vertices and edges, respectively. A directed graph is a graph whose edges are *ordered pairs* of vertices as opposed to *unordered pairs* in

FIGURE A.1: A simple undirected graph G .

the case of undirected graphs. A graph is simple if each edge is a pair of *distinct* vertices. This text is only concerned with simple undirected graphs, which will simply be referred to as graphs. Figure A.1 shows a simple undirected graph G , with vertex set $V = \{1, \dots, 7\}$ and edge set $E = \{\{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 6\}, \{4, 7\}, \{6, 7\}\}$.

A graph with a vertex set V is called a graph *on* V . The order of a graph G is the number of vertices in its vertex set. It is denoted by $|G|$. If G is a graph on V , then $|V|$ will also represent the size of the vertex set i.e., the number of elements of V . Let $v_1, v_2 \in V$, then the edge $\{v_1, v_2\}$ will simply be written as v_1v_2 . Two vertices of G are adjacent if they share an edge. Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. If $V' \subseteq V$ and $E' \subseteq E$ then G' is called a subgraph of G . Furthermore, if G' contains all the edges vw from G such that $v, w \in V'$, then G' is called an *induced* subgraph of G , and is denoted as $G' = G[V']$. The adjacency matrix A of a graph G on a vertex set V with $|V| = n$, is a $n \times n$ (square) matrix whose ij th entry, a_{ij} , is 1 if vertex v_i is adjacent to vertex v_j , and 0 otherwise. By definition, in a simple graph, a vertex v_i is not considered adjacent to itself and hence $a_{ii} = 0$ for $1 \leq i \leq n$. The adjacency matrix of the graph in Figure A.1 is:

$$\begin{array}{c}
 \begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\
 \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}
 \end{array}$$

A.4 Coding Theory

This short introduction to coding theory uses some notation different from the standard notation used in coding theory texts. This is done so that the treatment is consistent with the use of notation elsewhere in this thesis. These definitions are taken from [76].

For a more detailed account see any introductory text in coding theory. Let \mathbb{F}_d be a finite field, where d is a prime.¹ Let \mathbb{F}_d^n be an n -dimensional vector space over \mathbb{F}_d . A code is a subset C of \mathbb{F}_d^n . A linear code is a linear subspace of \mathbb{F}_d^n . An element $\mathbf{c} \in C$ is called a codeword. Let G be a $r \times n$ matrix whose rows form the basis of C . We say that C is a $[n, r]$ code, with generator matrix G . The dual code of C , denote C^\perp , is the orthogonal space of C . That is, it consists of vectors $\mathbf{y} \in \mathbb{F}_d^n$ such that $\mathbf{y} \cdot \mathbf{x}^T \equiv 0 \pmod{d}$ for every $\mathbf{x} \in C$. If C is an $[n, r]$ code, then C^\perp is an $[n, n - r]$ code. Let $m = n - r$. Any generator matrix of C^\perp is called a parity check matrix of C . If A is a parity check matrix, then C consists of all vectors \mathbf{x} such that:

$$A\mathbf{x} \equiv \mathbf{0} \pmod{d}$$

Clearly A is an $m \times n$ matrix. Let \mathbf{m} be an r element (message) vector. Then $\mathbf{x} = G^T \mathbf{m}$ is the encoding of \mathbf{m} . Suppose \mathbf{x} is sent over a noisy channel that introduces an error vector \mathbf{e} with weight $\text{wt}(\mathbf{e}) = k$. Then $\mathbf{y} \equiv \mathbf{x} + \mathbf{e} \pmod{d}$ is the received codeword. Let A be a parity check matrix. Let $A\mathbf{y} \equiv \mathbf{z} \pmod{d}$. $A\mathbf{y}$ is called the syndrome of the codeword \mathbf{y} . We know that $A\mathbf{x} \equiv \mathbf{0} \pmod{d}$. Then, we have:

$$\begin{aligned} A\mathbf{y} - A\mathbf{x} &\equiv \mathbf{z} - \mathbf{0} \pmod{d} \\ \Rightarrow A(\mathbf{y} - \mathbf{x}) &\equiv \mathbf{z} \pmod{d} \\ \Rightarrow A\mathbf{e} &\equiv \mathbf{z} \pmod{d} \end{aligned}$$

Thus, recovering the codeword \mathbf{x} from \mathbf{y} is the same as finding an \mathbf{e} of Hamming weight k that satisfies the above system of linear equations.

A.5 Coskun and Herley's Attack

The attack from Coskun and Herley [27] uses the observation that if a small number of bits of the secret is used to respond to challenges, then among two “candidates” for the secret, the one that is similar to the secret (in terms of the number of bits) is more likely to generate the same response. The adversary can know which of the candidates for the secret is more likely to be “close” to the secret, by checking their responses on a given set of challenges. From this, the adversary can “move” further towards the secret by checking if the “neighbours” of a candidate produce responses similar to the ones generated by the secret. If they do, they are more likely to be closer to the secret. Thus, the adversary can iteratively move towards the secret.

As mentioned before, Coskun and Herley's attack was shown on a generic human identification protocol which has an N -bit secret out of which U bits are used to construct a response to a challenge, and secrets can differ from each other by 1 or more bits. Generalizing this attack on the Counting Edges protocol requires some changes to the results described in [27]. We begin by first describing the notation that should be used exclusively in this section, which corresponds to the notation used by Coskun and Herley in [27].

¹Even though there exists a finite field for every prime power, we restrict our treatment to finite fields when d is a prime. As \mathbb{Z}_d is a finite field for every prime d , this allows us to use notation from modular arithmetic.

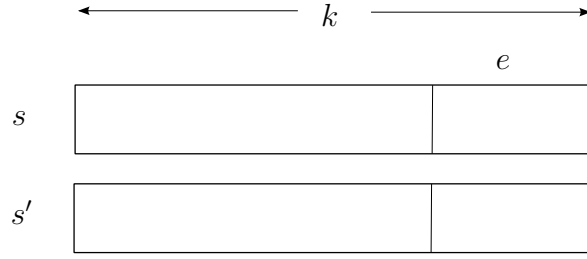


FIGURE A.2: $k - e$ icons between s and s' are the same, and e are different.

Notation

Let S denote the secret space, where $|S| = n$. A secret is a k -element subset of S . We shall call the elements of S , icons (or interchangeably, objects). Thus $s \in S$ is a set of k icons. The symbol s shall be reserved for the secret. A candidate for s (the secret) is any k -element subset of S . Symbols s', s'', \dots , shall represent candidates for s . A k -element subset of S , s' , is not a candidate for the secret, if given a set of challenges, at least one of the responses from s' is different from s .

A response is an element from the set $\{0, 1, \dots, d-1\}$, for some $d \geq 2$. A response shall be denoted by r . A response stream is a sequence of responses to m challenges, denoted by $r_1 r_2 \dots r_m$. We shall denote the response stream from the secret s by \mathfrak{R} . Similarly, for candidates s', s'', \dots , the response streams shall be denoted by $\mathfrak{R}', \mathfrak{R}'', \dots$. Let $s, s' \in S$, we say $\text{diff}(s, s') = e$, if $s - s' = e$. That is, the k -element subsets s and s' differ from each other in e elements (or icons). We say that s and s' are a distance e from each other, or are distance- e neighbours. See Figure A.2. For any s, s' in S , if $\text{diff}(s, s') = 1$, s and s' may simply be referred to as neighbours (instead of distance-1 neighbours). For any two response streams \mathfrak{R} and \mathfrak{R}' , define $\text{sim}(\mathfrak{R}, \mathfrak{R}')$ to be the number of challenges to which s and s' give the same response. In other words, the number of places in the m -element strings: $r_1 \dots r_m$ and $r'_1 \dots r'_m$, such that $r_i = r'_i$. Clearly, $0 \leq \text{sim}(\mathfrak{R}, \mathfrak{R}') \leq m$.

A.5.1 The Attack on Counting Edges Protocol

Our description of the attack on the Counting Edges Protocol follows the same structure as in [27]. In particular, the section headings are the same. We also implemented various results from [27] using MATLAB, and the results shall be discussed after the description of the attack. Recall that in a nutshell, the Counting Edges Protocol requires the user to count the number of edges in the induced subgraph of his secret vertices, and return the result modulo a positive integer d . Note that, the values of n , k , k' and n' should be chosen according to the rule: $\frac{n}{k} = \frac{n'}{k'}$. The suggested values of these parameters are: $n = 200$, $k = 20$, $n' = 50$, $k' = 5$ and $d = 3$.

“When Secrets are Close Responses are Closer”

Suppose, the adversary is given one challenge-response pair corresponding to the secret s . Let r denote the response. Let r' be the corresponding response constructed from a candidate s' to the secret. Assume $\text{diff}(s, s') = e$. Now the probability that none of the e different icons between s and s' were used in constructing the responses r and r' is given by:

$$\prod_{i=0}^{k'-1} \frac{(k-i) - e}{(k-i)}$$

Let p_e denote the probability that $r = r'$, when $\text{diff}(s, s') = e$. Then:

$$p_e = \prod_{i=0}^{k'-1} \frac{(k-i) - e}{(k-i)} + \frac{1}{d} \left(1 - \prod_{i=0}^{k'-1} \frac{(k-i) - e}{(k-i)} \right)$$

Intuitively, when e is low, p_e is close to 1, and if e is closer to $k = 20$, then p_e is approximately equal to the success probability of random guess, i.e., $\frac{1}{d} = \frac{1}{3}$. Let $0 \leq i \leq m$. The probability that $\text{sim}(\mathfrak{R}, \mathfrak{R}') = i$, such that $\text{diff}(s, s') = e$, is given by:

$$\Pr[\text{sim}(\mathfrak{R}, \mathfrak{R}') = i | \text{diff}(s, s') = e] = \binom{m}{i} (p_e)^i (1 - p_e)^{m-i} = B_{\text{pdf}}(i, m, p_e)$$

That is, the above probability is binomially distributed. We shall refer to the above probability as the p_e *binomial*, following the terminology used in [27]. Let m_{p_e} denote the mean of the p_e binomial. As e decreases, the mean of the binomial moves towards the right; in other words, the similarity increases.

“It’s Easy to Find a Secret That’s Close”

Let τ denote an integer between 0 and m , which is referred to as the threshold. Coskun and Herley’s algorithm chooses (retains) those candidates s' for which:

$$\text{sim}(\mathfrak{R}, \mathfrak{R}') \geq \tau \tag{A.11}$$

holds. That is, the algorithm only retains those candidates for further processing, whose response streams are identical to the response stream of the secret in at least a threshold τ number of places. The threshold can be equal to the mean of the p_e binomial, i.e., $\tau = m_{p_e}$, for some e . The reason for this is that, if we choose say $e = 12$, and choose the mean of the p_{12} binomial as the threshold, then we would expect a large number of candidates to be retained which are at a distance of $e = 12$ from the secret, as opposed to those which are at a distance of $e = 40$. Thus, one can choose the mean of a given p_e binomial as the threshold, such that there is a 50% chance that at least one of the candidates retained is at a distance e from the secret. For instance, if the threshold is chosen to be $\tau = 6$, i.e., the mean of the p_6 binomial, an approximately 0.52 fraction of candidates which are a distance 6 from the secret are retained. On the other hand, only about 0.0079 fraction of candidates which are a distance 12 from the secret are retained.

Now, given a secret s , the total number of candidates that are a distance i from the secret is given by:

$$T(n, k, i) = \binom{k}{k-i} \binom{n-k}{i}$$

where i ranges from 0 to k . Recall that there are $\binom{n}{k}$ total candidates for the secret (including the secret itself). Then, the total number of candidates that pass the test of Equation (A.11) is given by:

$$\begin{aligned} & \sum_{i=0}^k \binom{k}{k-i} \binom{n-k}{i} \left(1 - \sum_{j=\tau}^m \binom{m}{j} (p_i)^j (1-p_i)^{m-j} \right) \\ &= \sum_{i=0}^k T(n, k, i) (1 - B_{\text{cdf}}(\tau, m, p_i)) \end{aligned}$$

Since the total number of candidates that are a distance e from the secret are exactly $T(n, k, e)$, if the adversary chooses a random sample of size:

$$\frac{\binom{n}{k}}{T(n, k, e)} \tag{A.12}$$

then he would expect at least one candidate to be a distance e from the secret s . If e is very low, the size of the sample is almost the same as the size of the whole secret space. As e increases, the size of the random sample decreases. For instance, if $e = 1$, the size of the random sample is approximately 2^{79} , with $e = 6$ it is about 2^{40} , and when $e = 10$ the size is 2^{20} .

Now, in this random sample, we expect:

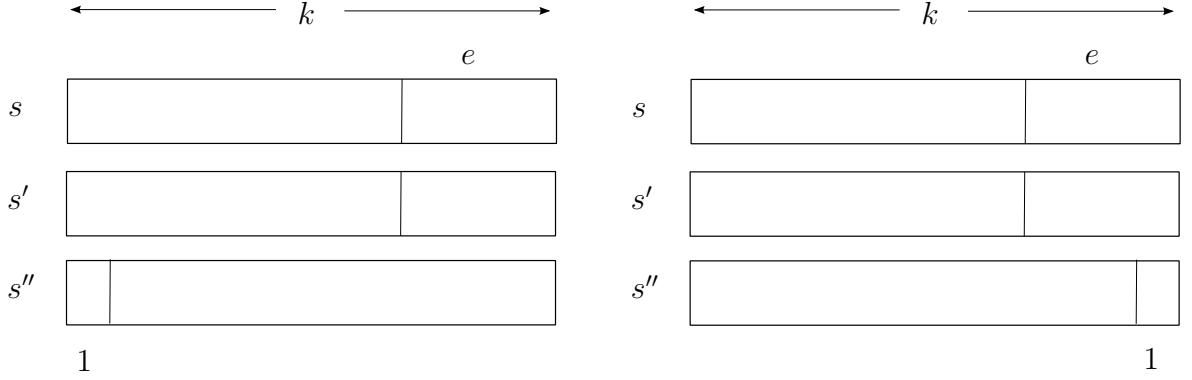
$$\frac{\sum_{i=0}^k T(n, k, i) (1 - B_{\text{cdf}}(\tau, m, p_i))}{\binom{n}{k}}$$

candidates to pass the test of Equation (A.11). Thus, the total candidates retained will be:

$$\frac{\sum_{i=0}^k T(n, k, i) (1 - B_{\text{cdf}}(\tau, m, p_i))}{T(n, k, e)}$$

As an example, if $m = 200$ and $\tau = 6$, the total number of candidates retained is 2^{33} . This is obtained after checking 2^{40} candidates from a random sample (cf. Equation (A.12)) to see if they pass the test of Equation (A.11). If we increase m to 1000, the number of candidates retained becomes 2^{13} . However, evaluating 2^{40} candidates on 1000 challenges takes considerably more time than evaluating them on say, 200 challenges. Thus, ideally m should be low enough, or else the attack is infeasible.

Recall that, retaining candidates in the way mentioned above means that if we choose $\tau = m_{p_e}$ then the attacker has a 50% chance that he will have one candidate which is a distance e from the secret. By doubling the size of the random selection (Equation (A.12)), the attacker can increase the probability of success exponentially. If the attacker increases the random sample q -fold, then the probability that the sample contains at least one candidate which is a distance e from the secret, is given by $1 - (1 - 0.5)^q$ [27].

FIGURE A.3: Graphical illustration of neighbours of s' .**“Once Close, It’s Easy to Get Closer”**

Now, suppose that the attacker has a candidate s' such that $\text{diff}(s, s') = e$. Recall that if $\text{diff}(s', s'') = 1$, s'' is a distance-1 neighbour or simply a neighbour of s' . That is, s'' is different from s' in only one icon. This implies that there are a total of $k(n - k)$ neighbours of s' .

Now, for every neighbour s'' of s' , there are three cases when considering $\text{diff}(s, s'')$. For some of these neighbours, we have:

$$\text{diff}(s, s'') = e - 1,$$

for some others we have:

$$\text{diff}(s, s'') = e,$$

and for the rest of the neighbours, we have:

$$\text{diff}(s, s'') = e + 1$$

This result is different from Coskun and Herley’s result from [27]. This is because in [27] they deal with differences of single bits, but in our case the difference is in terms of icons rather than bits. Therefore, in [27], there are only two possibilities; either the difference between s and s'' is $e + 1$ or $e - 1$. The following theorem quantifies the number of neighbours belonging to the above mentioned three categories.

Theorem 9. *Suppose $\text{diff}(s, s') = e$, and let s'' be a neighbour of s' . Then,*

$$\begin{aligned} \text{diff}(s, s'') = e - 1 & \quad \text{for } e^2 \text{ neighbours of } s' \\ \text{diff}(s, s'') = e & \quad \text{for } e(k - e) + (n - k - e)e \text{ neighbours of } s' \\ \text{diff}(s, s'') = e + 1 & \quad \text{for } (n - k - e)(k - e) \text{ neighbours of } s' \end{aligned}$$

Proof. Consider Figure A.3. Call the icon in s'' which is not present in s' , the singleton. First suppose that the singleton corresponds to one of the $k - e$ icons s' has in common with s (See left hand figure in Figure A.3). This implies that s'' is already different

TABLE A.1: Number of neighbours with differences $e - 1$, e and $e + 1$.

n	k	e	diff = $e - 1$	diff = e	diff = $e + 1$	Total = $n(n - k)$
200	20	6	36	1128	2436	3600

from s in e icons. Thus regardless of the choice of the singleton, $\text{diff}(s, s'')$ cannot be $e - 1$ in this case. Next suppose that the singleton corresponds to the e icons in which s' and s are different from each other (See right hand figure in Figure A.3). Then, if the singleton is one of the e icons of s , which are different from s' , we have $\text{diff}(s, s'') = e - 1$. There are a total of e possibilities for such a singleton. Therefore, we have $e \cdot e = e^2$ neighbours of s' for which $\text{diff}(s, s'') = e - 1$.

Again, suppose that the singleton corresponds to one of the $k - e$ icons s' has in common with s . Then, if the singleton is one of the e icons of s that are different from s' , we get $\text{diff}(s, s'') = e$. That is, there is no change in difference. There are a total of $k - e$ possibilities for such singletons in this case, which implies $e(k - e)$ possible neighbours s'' which are a distance e from s . Next suppose that the singleton corresponds to the e icons in which s' and s are different from each other. If the singleton is from the $n - k - e$ icons that are different from the k icons of s , and the e icons of s' in which it differs from s , then $\text{diff}(s, s'') = e$. There are e possibilities for such singletons, giving the number $(n - k - e)e$. Combining the two, we get the result that for $e(k - e) + (n - k - e)e$ neighbours of s' , we have $\text{diff}(s, s'') = e$.

Finally, suppose again that the singleton corresponds to one of the $k - e$ icons s' has in common with s . If the singleton belongs to the $n - k - e$ icons, that are different from the k icons of s , and the e icons of s' in which it differs from s , then $\text{diff}(s, s'') = e + 1$, as s'' already differs from s by e icons. There are a total of $k - e$ possibilities for such singletons, giving us the quantity: $(n - k - e)(k - e)$. If the singleton corresponds to the e icons in which s' and s are different from each other, then regardless of the choice of the singleton, the difference cannot be $e + 1$. Thus, for $(n - k - e)(k - e)$ neighbours of s' , we get $\text{diff}(s, s'') = e + 1$. \square

The sum of these neighbours is of course $k(n - k)$. Table A.1 shows the number of neighbours with the three difference levels. As can be seen, there is a much smaller number of neighbours which have difference $e - 1$ from the secret. The adversary does not have the knowledge of which neighbours are closer to s and which are farther. But he can guess that those neighbours which produce a response stream closer to the response stream of the secret, are more likely to be closer to the secret. The two probability distributions corresponding to the differences e and $e - 1$ are given by:

$$\Pr[\text{sim}(\mathfrak{R}, \mathfrak{R}'') = i | \text{diff}(s, s'') = e] = B_{\text{pdf}}(i, m, p_e)$$

and:

$$\Pr[\text{sim}(\mathfrak{R}, \mathfrak{R}'') = i | \text{diff}(s, s'') = e - 1] = B_{\text{pdf}}(i, m, p_{e-1})$$

Notice that we are checking the difference between e and $e - 1$ instead of $e + 1$ and $e - 1$ as in [27]. For a fixed e , the distance between the two binomials increases with increasing m . For smaller e 's a smaller value of m suffices. However, as e increases, m needs to be increased to increase the distance. Thus, the target is to find a balance between a high e and a low m .

“Putting the Pieces Together”

Finally, Coskun and Herley's algorithm on k -out-of- n protocols is as follows.

Attack: Coskun and Herley's Divide-and-Conquer Attack.

Input: A set of m challenge-response pairs.

Output: A candidate for the secret.

```

1: for all  $q \times \frac{\binom{n}{k}}{T(n,k,e)}$  random candidates  $s'$  do
2:   if  $\text{sim}(\mathfrak{R}, \mathfrak{R}') < \tau$  then
3:     skip to the next candidate.
4:   else
5:     initialize an empty list. Put  $s'$  in the list.
6:     for  $i = 0$  to  $e$  do
7:       for all  $k(n - k)$  distance-1 neighbour  $s''$  of each element in list
8:         do
9:           calculate  $\text{sim}(\mathfrak{R}, \mathfrak{R}'')$ . Put  $s''$  in list.
10:          retain the 10 candidates that maximize  $\text{sim}(\mathfrak{R}, \mathfrak{R}'')$ , and discard
11:            all others.
12:          if  $\text{sim}(\mathfrak{R}, \mathfrak{R}'') = m$  then
13:            break.

```

Suppose $q = 1$, then the time-complexity of the above attack is:

$$\frac{\binom{n}{k} + 10 \times k(n - k) \times e \times \sum_{i=0}^k T(n, k, i)(1 - B_{\text{cdf}}(\tau, m, p_i))}{T(n, k, e)} \times m$$

where the fundamental unit of complexity is the computation of one response. To decrease the time-complexity, the adversary needs to choose a higher e . However, if e is too high, the response streams of the distance-1 neighbours are not distinguishable. Central to the feasibility of the above mentioned attack is the subroutine constituting Steps 5 to 11. We call this the Hopping-to-the-Secret subroutine, to acknowledge that this part of the algorithm attempts to move closer to the secret by eliminating the difference from the secret by at least one icon per iteration. As e increases, we need a larger value of m to find the secret through the Hopping-to-the-Secret subroutine. For instance, through our implementation, we found that for $e = 5$, $m = 300$ suffices. However, for $e = 6$, $m = 300$ is not enough to find the secret.

Performance of the Attack on the Counting Edges Protocol

As already mentioned, one needs to increase e to make Coskun and Herley's attack feasible. The downside is that increasing e means an increase in the number of challenge-response pairs m on which candidates for the secret needs to be evaluated. Our implementation results showed that when a candidate is a distance $e = 6$ from the secret, the Hopping-to-the-Secret subroutine did not find the secret when $m = 300$. Thus, we can consider $m = 300$ to be a lower bound to obtain the secret when we have at least one candidate in the random selection from Equation (A.12) which is a distance 6 from the secret.

If we take the underestimate that the evaluation of $m = 300$ challenges takes 1 second, this means that in a random sample of 2^{40} candidates, this would take about 2^{40} seconds, or more than 34,000 years for the attack to output the secret. With a lower value of e , say 5, the situation is no better, as the size of the random sample (cf. Equation (A.12)) increases as well (2^{46} , when $e = 5$). Furthermore, with a decreased e , a much larger number of candidates pass the test of Equation (A.12), which in turn means that the Hopping-to-the-Secret subroutine needs to be run for each of these candidates. Given one candidate for the secret such that $e = 6$, and with $m = 1000$, the Hopping-to-the-Secret subroutine took about 3 days to output the secret on an Intel Core 2 Duo processor at 3.00GHz. Thus, on a larger number of candidates this subroutine is bound to take considerable time. Thus, we can assume that the attack is infeasible in practice on the Counting Edges Protocol.

The infeasibility of the attack is perhaps due to the fact that the number of secret bits to compute the secret in the Counting Edges Protocol is higher than that evaluated by Coskun and Herley in their generic protocol. Let U denote the number of bits of the secret used to compute a response. Then, for the Counting Edges protocol we have:

$$U = \log_2 \binom{n}{k'} = \log_2 \binom{200}{5} \approx 31$$

In contrast, Coskun and Herley gave theoretical results for a maximum value of $U = 10$. They claim that beyond this, the protocols are bound to take an impractical amount of time (more than 10 minutes per authentication session). However, we argue that this is not always true, as it depends on how the protocol is implemented, and therefore human computational time cannot be directly calculated as a function of the number of secret bits involved in computing challenges.

References

- [1] S. Li and H.-Y. Shum. *Secure Human-Computer Identification against Peeping Attacks (SecHCI): A Survey*. Technical Report. (2003). URL <http://www.hooklee.com/Papers/SecHCI-Survey.pdf>. 1, 20, 21, 23
- [2] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography* (CRC Press, Inc., Boca Raton, FL, USA, 1996), 1st ed. 1, 2, 80
- [3] L. Lamport. *Password Authentication with Insecure Communication*. Communications of the ACM **24**, 770 (1981). 1
- [4] A. Fiat and A. Shamir. *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*. In *Proceedings of CRYPTO '86*, pp. 186–194 (Springer-Verlag, Berlin, 1987). 1
- [5] T. Matsumoto and H. Imai. *Human Identification through Insecure Channel*. In *Proceedings of Eurocrypt '91*, vol. 547, pp. 409–421 (Springer-Verlag, Berlin, 1991). 2, 4, 6, 22, 23
- [6] D. Bradbury. *A Hole in the Security Wall: ATM Hacking*. Network Security **2010**(6), 12 (2010). URL [http://dx.doi.org/10.1016/S1353-4858\(10\)70082-9](http://dx.doi.org/10.1016/S1353-4858(10)70082-9). 3
- [7] P. Tuyls, A. H. Akkermans, T. A. Kevenaar, G.-J. Schrijen, A. M. Bazen, and R. N. Veldhuis. *Practical Biometric Authentication with Template Protection*. In *Proceedings of AVBPA '05*, pp. 436–446 (Springer-Verlag, Berlin, 2005). 4
- [8] N. J. Hopper and M. Blum. *Secure Human Identification Protocols*. In *Proceedings of Asiacrypt '01*, vol. 2248, pp. 52–66 (Springer-Verlag, Berlin, 2001). 4, 5, 6, 10, 11, 12, 19, 21, 24, 25, 73, 80, 81, 82, 83, 84, 86, 88, 92, 95, 102, 105, 106, 107, 108, 109, 111, 112, 113, 118, 121, 122, 127, 129, 134
- [9] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. *Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme*. In *Proceedings of AVI '06*, pp. 177–184 (ACM, 2006). 4, 8, 13, 16, 20, 26, 27, 47, 49, 52, 53, 56, 58, 71, 130
- [10] D. Weinshall. *Cognitive Authentication Schemes Safe Against Spyware (Short Paper)*. In *Proceedings of SP '06*, pp. 295–300 (IEEE Computer Society, Washington, 2006). 5, 26, 27, 38, 88

- [11] M. Lei, Y. Xiao, S. V. Vrbsky, and C.-C. Li. *Virtual Password using Random Linear Functions for On-line Services, ATM Machines, and Pervasive Computing*. Computer Communications **31**(18), 4367 (2008). 5, 27
- [12] C.-H. Wang, T. Hwang, and J.-J. Tsai. *On the Matsumoto and Imai's Human Identification Scheme*. In *Proceedings of EUROCRYPT '95*, vol. 921, pp. 382–392 (Springer-Verlag, Berlin, 1995). 5, 23
- [13] X.-Y. Li and S.-H. Teng. *Practical Human-Machine Identification over Insecure Channels*. Journal of Combinatorial Optimization **3**(4), 347 (1999). 23
- [14] S. Li and H.-Y. Shum. *Secure Human-Computer Identification (Interface) Systems against Peeping Attacks: SecHCI*. Technical Report. (2003). URL <http://www.hooklee.com/Papers/SecHCI.pdf>. 5, 25, 93, 106, 108, 124, 125, 127, 129
- [15] H. Jameel, R. Shaikh, H. Lee, and S. Lee. *Human Identification Through Image Evaluation Using Secret Predicates*. In *Proceedings of CT-RSA 2007*, pp. 67–84 (Springer-Verlag, Berlin, 2007). 5, 25, 132
- [16] H. Jameel, R. Shaikh, L. Hung, Y. Wei, S. Raazi, N. Canh, S. Lee, H. Lee, Y. Son, and M. Fernandes. *Image-Feature Based Human Identification Protocols on Limited Display Devices*. In *Proceedings of WISA '08*, pp. 211–224 (Springer-Verlag, Berlin, 2008). 25
- [17] X. Bai, W. Gu, S. Chellappan, X. Wang, D. Xuan, and B. Ma. *PAS: Predicate-based Authentication Services Against Powerful Passive Adversaries*. In *Proceedings of ACSAC '08*, pp. 433–442 (IEEE Computer Society, Washington, 2008). 8, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 47, 88, 91, 129, 130, 133, 135
- [18] H. Sasamoto, N. Christin, and E. Hayashi. *Undercover: Authentication Usable in Front of Prying Eyes*. In *Proceeding of CHI '08*, pp. 183–192 (ACM, New York, 2008). 5, 27, 131
- [19] T. Matsumoto. *Human-Computer Cryptography: An Attempt*. In *Proceedings of CCS '96*, pp. 68–75 (ACM, New York, 1996). 6, 15, 17, 19, 23, 24, 25, 73, 80, 129, 132
- [20] J. Flum and M. Grohe. *Parameterized Complexity Theory* (Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006). 7, 8, 25, 109, 110, 111
- [21] L. Sobrado and J.-C. Birget. *Graphical Passwords*. The Rutgers Scholar 4 (2002). URL <http://rutgersscholar.rutgers.edu/volume04/sobrbirg/sobrbirg.htm>. 8, 13, 16, 20, 21, 27, 47, 49, 71, 109, 129, 130
- [22] L. S. Charlap, H. D. Rees, and D. P. Robbins. *The Asymptotic Probability that a Random Biased Matrix is Invertible*. Discrete Math. **82**, 153 (1990). 19, 103

- [23] I. Jermyn, A. Mayer, F. Monroe, M. K. Reiter, and A. D. Rubin. *The Design and Analysis of Graphical Passwords*. In *Proceedings of USENIX '99*, pp. 1–14 (USENIX Association, Berkeley, 1999). [20](#)
- [24] J. Thorpe and P. C. van Oorschot. *Graphical Dictionaries and the Memorable Space of Graphical Passwords*. In *Proceedings of USENIX '04*, pp. 135–150 (USENIX Association, Berkeley, 2004). [20](#)
- [25] X. Suo, Y. Zhu, and G. S. Owen. *Graphical Passwords: A Survey*. In *Proceedings of ACSAC '05*, pp. 463–472 (IEEE Computer Society, Washington, 2005). [20](#)
- [26] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. *CAPTCHA: Using Hard AI Problems for Security*. In *Proceedings of EUROCRYPT '03*, pp. 294–311 (Springer-Verlag, Berlin, 2003). [25](#), [133](#)
- [27] B. Coskun and C. Herley. *Can "Something You Know" Be Saved?* In *Proceedings of ISC '08*, pp. 421–440 (Springer-Verlag, Berlin, 2008). [26](#), [104](#), [141](#), [142](#), [143](#), [144](#), [145](#), [147](#)
- [28] T. Perkovic, A. Mumtaz, Y. Javed, S. Li, S. A. Khayam, and M. Cagalj. *Breaking Undercover: Exploiting Design Flaws and Nonuniform Human Behavior*. In *Proceedings of SOUPS '11* (ACM, New York, 2011). [27](#), [131](#)
- [29] P. Golle and D. Wagner. *Cryptanalysis of a Cognitive Authentication Scheme (Extended Abstract)*. In *Proceedings of SP '07*, pp. 66–70 (IEEE Computer Society, Washington, 2007). [27](#), [38](#), [88](#)
- [30] S. Li, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz. *Breaking Randomized Linear Generation Functions based Virtual Password System*. In *Proceedings of ICC '10*, pp. 1–6 (IEEE Communications Society, 2010). [27](#)
- [31] R. Dhamija and A. Perrig. *Déjà Vu: A user study using images for authentication*. In *Proceedings of USENIX '00*, pp. 45–58 (USENIX Association, Berkeley, 2000). [27](#)
- [32] V. Roth, K. Richter, and R. Freidinger. *A PIN-Entry Method Resilient Against Shoulder Surfing*. In *Proceedings of CCS '04*, pp. 236–245 (ACM, New York, 2004). [27](#), [135](#)
- [33] H. Zhao and X. Li. *S3PAS: A Scalable Shoulder-Surfing Resistant Textual-Graphical Password Authentication Scheme*. In *Proceedings of AINAW '07*, pp. 467–472 (IEEE Computer Society, Washington, 2007). [27](#)
- [34] J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. *Algorithms for the Satisfiability (SAT) Problem: A Survey*. In *Satisfiability Problem: Theory and Applications*, vol. 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, chap. 2, pp. 19–152 (American Mathematical Society, 1996). [38](#)

- [35] D. Florêncio and C. Herley. *A Large-Scale Study of Web Password Habits*. In *Proceedings of WWW '07*, pp. 657–665 (ACM, New York, 2007). 39
- [36] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms* (McGraw-Hill Higher Education, 2001), 2nd ed. 50
- [37] G. Turk. *Graphics gems*, pp. 24–28 (Academic Press Professional, Inc., San Diego, CA, USA, 1990). 57, 62, 137
- [38] S. Ross. *A First Course in Probability* (Prentice Hall, 2006), 7th ed. 69
- [39] S. Man, D. Hong, J.-C. Birget, and M. Mathews. *A Shoulder-Surfing Resistant Graphical Password Scheme*. Technical Report. (2005). URL <http://clam.rutgers.edu/~birget/grPssw/dwManWIW.pdf>. 71
- [40] J. M. Pollard. *Monte Carlo Methods for Index Computation (mod p)*. *Mathematics of Computation* **32**, 918 (1978). 73
- [41] D. R. Stinson. *Some Baby-Step Giant-Step Algorithms for the Low Hamming Weight Discrete Logarithm Problem*. *Mathematics of Computation* **71**, 379 (2002). 73, 80, 81
- [42] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone. *An Implementation for a Fast Public-Key Cryptosystem*. *Journal of Cryptology* **3**(2), 63 (1991). 81
- [43] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. *Monographs in computer science* (Springer, 1999). 91, 107
- [44] R. G. Downey, M. R. Fellows, A. Vardy, and G. Whittle. *The Parametrized Complexity of Some Fundamental Problems in Coding Theory*. *SIAM J. Comput.* **29**, 545 (1999). 92, 109, 113, 114, 118, 119
- [45] R. Diestel. *Graph Theory* (Springer-Verlag, Berlin, Germany, 2005). 94, 139
- [46] K. P. Bogart. *Introductory Combinatorics* (Pitman Publishing, Inc., Marshfield, MA, USA, 1983), 4th ed. 96
- [47] B. Lin and Y. Chen. *The Parameterized Complexity of k -Edge Induced Subgraphs*. *CoRR* **abs/1105.0477** (2011). 96
- [48] D. L. Donoho and J. Tanner. *Sparse Nonnegative Solution of Underdetermined Linear Equations by Linear Programming*. *Proceedings of the National Academy of Sciences of the United States of America* **102**(27), 9446 (2005). 98
- [49] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. *Counting Paths and Packings in Halves*. In *Proceedings of ESA '09*, pp. 578–586 (Springer-Verlag, Berlin, 2009). 102

- [50] R. Niebuhr, P.-L. Cayrel, S. Bulygin, and J. Buchmann. *On Lower Bounds for Information Set Decoding over \mathbb{F}_q* . In *Proceedings of SCC '10*, pp. 143–157 (2010). [103](#), [118](#)
- [51] F. Chabaud. *On the Security of Some Cryptosystems Based on Error-correcting Codes*. In *Proceedings of EUROCRYPT '94*, pp. 131–139 (Springer-Verlag, Berlin, 1994). [103](#)
- [52] R. G. Gallager. *Low-Density Parity-Check Codes* (1963). MIT Press, Cambridge. [103](#)
- [53] M. Baldi, M. Bodrato, and F. Chiaraluce. *A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes*. In *Proceedings of SCN '08*, pp. 246–262 (Springer-Verlag, Berlin, 2008). [103](#)
- [54] U. Feige and M. Seltser. *On the Densest k -Subgraph Problem*. Technical Report. (1997). [105](#)
- [55] A. Juels and S. A. Weis. *Authenticating Pervasive Devices with Human Protocols*. In *Proceedings of CRYPTO '05*, pp. 293–308 (Springer-Verlag, Berlin, 2005). [107](#), [108](#), [121](#), [134](#)
- [56] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman & Co., New York, NY, USA, 1990). [107](#)
- [57] J. Hastad. *Some Optimal Inapproximability Results*. Journal of the ACM **48**, 798 (2001). [108](#)
- [58] L. G. Valiant. *The Complexity of Computing the Permanent*. Theoretical Computer Science **8**, 189 (1979). [111](#)
- [59] N. Alon, R. Yuster, and U. Zwick. *Color-coding*. Journal of the ACM **42**, 844 (1995). [111](#)
- [60] J. Flum and M. Grohe. *The Parameterized Complexity of Counting Problems*. In *Proceedings of FOCS '02*, pp. 538–547 (IEEE Computer Society, Washington, 2002). [111](#)
- [61] E. Berlekamp, R. McEliece, and H. Van Tilborg. *On the Inherent Intractability of Certain Coding Problems*. IEEE Transactions on Information Theory **24**(3), 384 (1978). [114](#), [118](#)
- [62] Y.-J. Chang and B. W. Wah. *Lagrangian Techniques for Solving a Class of Zero-One Integer Linear Programs*. In *Proceedings of COMPSAC '95*, pp. 156–161 (IEEE Computer Society, Washington, 1995). [114](#)
- [63] M. Liazi, I. Milis, and V. Zissimopoulos. *A Constant Approximation Algorithm for the Densest k -Subgraph Problem on Chordal Graphs*. Information Processing Letters **108**, 29 (2008). [123](#)

- [64] G. Kortsarz and D. Peleg. *On Choosing a Dense Subgraph*. In *Proceedings of SFCS '93*, pp. 692–701 (IEEE Computer Society, Washington, 1993). 123
- [65] R. G. Downey and M. R. Fellows. *Fixed-Parameter Tractability and Completeness II: On Completeness for $W[1]$* . *Theoretical Computer Science* **141**, 109 (1995). 123
- [66] O. Regev. *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography*. In *Proceedings of STOC '05*, pp. 84–93 (ACM, New York, 2005). 126
- [67] V. Lyubashevsky, C. Peikert, and O. Regev. *On Ideal Lattices and Learning with Errors over Rings*. In *Proceedings of EUROCRYPT '10*, pp. 1–23 (Springer-Verlag, Berlin, 2010). 126
- [68] P. C. Kocher, J. Jaffe, and B. Jun. *Differential Power Analysis*. In *Proceedings of CRYPTO '99*, pp. 388–397 (Springer-Verlag, Berlin, 1999). 131
- [69] P. C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. In *Proceedings of CRYPTO '96*, pp. 104–113 (Springer-Verlag, Berlin, 1996). 131
- [70] A. Juels. *Minimalist Cryptography for Low-Cost RFID Tags*. In *Proceedings of SCN '04*, pp. 149–164 (Springer-Verlag, Berlin, 2004). 132
- [71] Google Inc. *reCAPTCHA* (2010). URL <http://www.google.com/recaptcha>. 133
- [72] H. Gao and X. Liu. *A New Graphical Password Scheme Against Spyware by Using CAPTCHA*. In *Proceedings of SOUPS '09*, pp. 21:1–21:1 (ACM, New York, 2009). 133
- [73] J. Munilla and A. Peinado. *HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols*. *Computer Networks* **51**, 2262 (2007). 134
- [74] D. Kim, P. Dunphy, P. Briggs, J. Hook, J. Nicholson, J. Nicholson, and P. Olivier. *Multi-Touch Authentication on Tabletops*. In *Proceedings of CHI '10*, pp. 1093–1102 (ACM, New York, 2010). 135
- [75] A. Drucker. *Multiplying 10-Digit Numbers using Flickr: The Power of Recognition Memory*. Online Article. (2011). URL http://people.csail.mit.edu/andyd/rec_method. 135
- [76] V. D. Tonchev. *An Introduction to Coding Theory*. Lecture Notes. (2009). URL <http://www.math.mtu.edu/~tonchev/Coding-Theory-Tohoku-June-09.pdf>. 140

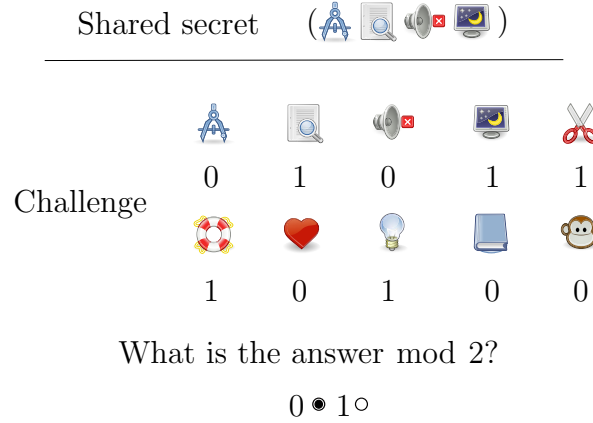


FIGURE 2.1: A challenge and response from the Example Protocol.

However, n cannot be arbitrarily large, since a reasonably sized display unit cannot accommodate a huge n . So, typically n can range between 100 to 300. If a subset of n icons is displayed, then it is possible to increase n . However, this introduces some other constraints, as we shall show during the security analysis of the convex hull click based protocol from [9, 21] in Chapter 4. Briefly, when only a subset of total icons are shown in each challenge, the probability that a secret icon appears in the subset should be the same as the probability of appearance of a non-secret icon. Otherwise, the adversary can do a “frequency analysis” to find the secret. Thus, even in this case, n cannot be made arbitrarily large, as that would mean an increase in the number of secret icons, and hence additional memory load on humans.

Text or Pictures?

As mentioned before, the protocol can also be implemented using a text-based implementation as opposed to graphical. For instance, in each cell an alphanumeric character can be shown instead of an icon. The user can then remember an alphanumeric password, and perform the steps by following the cells according to the alphanumerics in his password. However, a graphical implementation is preferred due to two main reasons. First, humans can recall a graphical object more quickly than text [1, 23]. In fact, it is much easier for humans to recall a picture due to the characteristics of human memory, as opposed to words including complete nouns, as some psychological studies have shown [24]. Secondly, in contrast to a textual implementation, a graphical implementation is less prone to dictionary attacks [24, 25]. Although there is some indication that a dictionary can be created from finding patterns in the selection of secret graphical objects by humans [24], the resulting dictionary attacks are less likely to be as severe as in the case of (textual) passwords. Thus, from both security and usability perspectives, a graphical implementation is preferable.

<div style="background: linear-gradient(to top right, #f08080, #d8bfd8); padding: 5px; display: inline-block;"> A B E Z V U X D R S W G H </div>	<div style="background: linear-gradient(to top right, #f08080, #d8bfd8); padding: 5px; display: inline-block;"> C V Y Z T U Q R D S I O N </div>	y	n
1	2	0	1
		1	0

FIGURE 3.1: A challenge and response table from PAS.

one authentication session only, and the password has to be renewed after all the len possible values are exhausted. The predicate indices of the len authentication sessions may simply be chosen as $1, \dots, len$ or a permutation of the len values. In this chapter, we assume the PAS scheme runs in a “random permutation mode”, in which a random permutation of $1, \dots, len$ determines the predicate index used for each authentication session. Note that this is the most complicated (and thus the most “secure”) way one can adopt to assign the len values of the predicate index to all the authentication sessions.

Extended PAS


























Bai et al. also extended the above basic PAS scheme to allow $k > 1$ cell indices in each secret S_i . In this case, the i th secret in the password is redefined as $S_i = (c_{i,1}, \dots, c_{i,k}, W_i)$. Accordingly, k predicate indices I_1, \dots, I_k will be sent from \mathcal{C} to \mathcal{H} for each authentication session. \mathcal{H} calculates the i th predicate pred_i as a set of k sub-predicates $\{\text{pred}_{i,j}\}_{j=1}^k$, where:

$$\begin{aligned}
\text{pred}_{i,j} &= (c_{i,\hat{I}_{j,k}}, h_{i,j}), \\
h_{i,j} &= w_i[I_{j,len}], \\
\hat{I}_{j,k} &= (I_j \bmod k) + 1, \\
\text{and } \hat{I}_{j,len} &= (I_j \bmod len) + 1.
\end{aligned}$$

With this extended predicate containing k sub-predicates, the hidden response B_i of the i th predicate is obtained as follows: \mathcal{H} first calculates k hidden sub-responses $B_{i,1}, \dots, B_{i,k}$ for the k sub-predicates in the same way as in the basic PAS scheme, and then determines B_i as the bitwise OR of the k hidden sub-responses: $B_i = B_{i,1} \vee \dots \vee B_{i,k}$. To ensure uniform distribution of B_i over $\{0, \dots, 2^l - 1\}$, the number of distinct characters in each cell of each challenge table and the corresponding probability β should be determined by Eqs. (6) and (8) in [17], respectively.

List of Parameters

A list of the parameters (with the default values) and notations involved in the description of the PAS scheme is given in Table 3.1. The column labeled “Notation (extended)” shows the notations from the extended PAS scheme.





( = 1,  = 2,  = 3,  = 4)

FIGURE 6.2: An example implementation of the basic Counting Edges Protocol.

secret was also very low for $m = 300$. Thus, it is evident that for some m in the range $[300, 400]$ with high probability the linear program above gives a solution that is also the solution of the original problem with $\text{wt}(\mathbf{x}) = \kappa$. Although, this program takes an m greater than m_{lb} , the value of m that can be safely used is not sufficient for practical purposes. This will only allow m/μ sessions before the secret needs to be renewed. We can choose μ to be 6 so that the chance of success of the random guess attack is around one in a million. With $m = 200$, this means only about 33 sessions after which the secret needs to be renewed.

Another drawback relates to the implementation of the protocol. We can implement the protocol using software icons similar to the one for the Example Protocol described in Chapter 2. A challenge can then be displayed as a grid of software icons connected through random edges. However, this way could be cumbersome for the user as n grows. A better way is to display the adjacency matrix of the graph. The borders of the matrix can be identified by software icons. In this case, the user will have to find entries in the matrix corresponding to his secret set of icons, and sum the entries (which will be binary). See Figure 6.2. The figure shows the adjacency matrix of the graph G_1 from Figure 6.1. The user only remembers the secret icons, and counts the positions corresponding to pairs of secret icons. The transparent icons in the background of each “cell” make it easier for the user to locate the correct entry. Still however, the drawback of this protocol is displaying an adjacency matrix of size up to 200×200 , which will definitely depend on the size of the display. We consider these two points and construct the revised Counting Edges Protocol next.

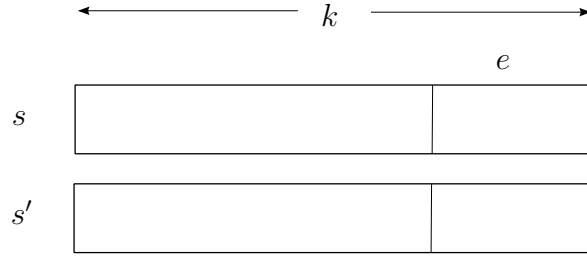


FIGURE A.2: $k - e$ icons between s and s' are the same, and e are different.

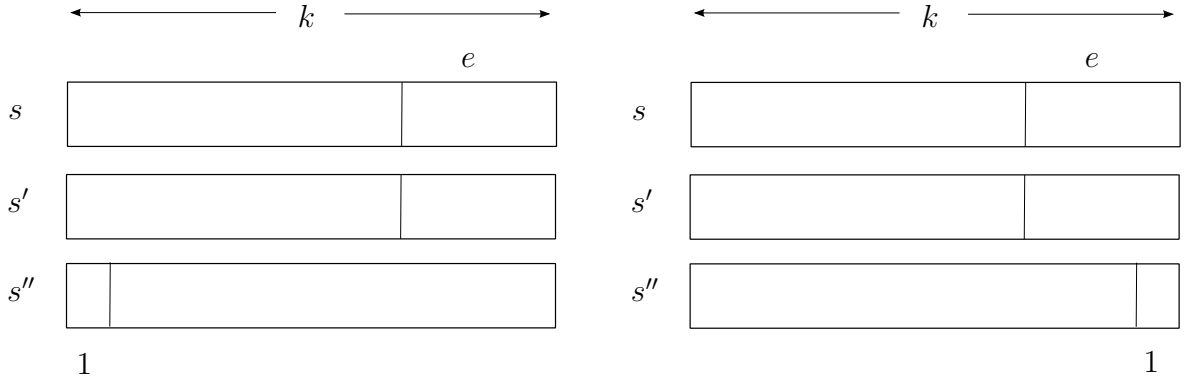
Notation

Let S denote the secret space, where $|S| = n$. A secret is a k -element subset of S . We shall call the elements of S , icons (or interchangeably, objects). Thus $s \in S$ is a set of k icons. The symbol s shall be reserved for the secret. A candidate for s (the secret) is any k -element subset of S . Symbols s', s'', \dots , shall represent candidates for s . A k -element subset of S , s' , is not a candidate for the secret, if given a set of challenges, at least one of the responses from s' is different from s .

A response is an element from the set $\{0, 1, \dots, d-1\}$, for some $d \geq 2$. A response shall be denoted by r . A response stream is a sequence of responses to m challenges, denoted by $r_1 r_2 \dots r_m$. We shall denote the response stream from the secret s by \mathfrak{R} . Similarly, for candidates s', s'', \dots , the response streams shall be denoted by $\mathfrak{R}', \mathfrak{R}'', \dots$. Let $s, s' \in S$, we say $\text{diff}(s, s') = e$, if $s - s' = e$. That is, the k -element subsets s and s' differ from each other in e elements (or icons). We say that s and s' are a distance e from each other, or are distance- e neighbours. See Figure A.2. For any s, s' in S , if $\text{diff}(s, s') = 1$, s and s' may simply be referred to as neighbours (instead of distance-1 neighbours). For any two response streams \mathfrak{R} and \mathfrak{R}' , define $\text{sim}(\mathfrak{R}, \mathfrak{R}')$ to be the number of challenges to which s and s' give the same response. In other words, the number of places in the m -element strings: $r_1 \dots r_m$ and $r'_1 \dots r'_m$, such that $r_i = r'_i$. Clearly, $0 \leq \text{sim}(\mathfrak{R}, \mathfrak{R}') \leq m$.

A.5.1 The Attack on Counting Edges Protocol

Our description of the attack on the Counting Edges Protocol follows the same structure as in [27]. In particular, the section headings are the same. We also implemented various results from [27] using MATLAB, and the results shall be discussed after the description of the attack. Recall that in a nutshell, the Counting Edges Protocol requires the user to count the number of edges in the induced subgraph of his secret vertices, and return the result modulo a positive integer d . Note that, the values of n , k , k' and n' should be chosen according to the rule: $\frac{n}{k} = \frac{n'}{k'}$. The suggested values of these parameters are: $n = 200$, $k = 20$, $n' = 50$, $k' = 5$ and $d = 3$.

FIGURE A.3: Graphical illustration of neighbours of s' .**“Once Close, It’s Easy to Get Closer”**

Now, suppose that the attacker has a candidate s' such that $\text{diff}(s, s') = e$. Recall that if $\text{diff}(s', s'') = 1$, s'' is a distance-1 neighbour or simply a neighbour of s' . That is, s'' is different from s' in only one icon. This implies that there are a total of $k(n - k)$ neighbours of s' .

Now, for every neighbour s'' of s' , there are three cases when considering $\text{diff}(s, s'')$. For some of these neighbours, we have:

$$\text{diff}(s, s'') = e - 1,$$

for some others we have:

$$\text{diff}(s, s'') = e,$$

and for the rest of the neighbours, we have:

$$\text{diff}(s, s'') = e + 1$$

This result is different from Coskun and Herley’s result from [27]. This is because in [27] they deal with differences of single bits, but in our case the difference is in terms of icons rather than bits. Therefore, in [27], there are only two possibilities; either the difference between s and s'' is $e + 1$ or $e - 1$. The following theorem quantifies the number of neighbours belonging to the above mentioned three categories.

Theorem 9. *Suppose $\text{diff}(s, s') = e$, and let s'' be a neighbour of s' . Then,*

$$\begin{aligned} \text{diff}(s, s'') &= e - 1 && \text{for } e^2 \text{ neighbours of } s' \\ \text{diff}(s, s'') &= e && \text{for } e(k - e) + (n - k - e)e \text{ neighbours of } s' \\ \text{diff}(s, s'') &= e + 1 && \text{for } (n - k - e)(k - e) \text{ neighbours of } s' \end{aligned}$$

Proof. Consider Figure A.3. Call the icon in s'' which is not present in s' , the singleton. First suppose that the singleton corresponds to one of the $k - e$ icons s' has in common with s (See left hand figure in Figure A.3). This implies that s'' is already different