Algebraic Attacks on Human Identification Protocols*

Hassan Jameel Asghar[1], Ron Steinfeld[2], Shujun Li[3], Mohamed Ali Kaafar[1],
Josef Pieprzyk[4]

[1]National ICT Australia, NICTA, Australia
{hassan.asghar,dali.kaafar}@nicta.com.au
[2]Clayton School of Information Technology,
Faculty of Information Technology,
Monash University, Clayton, Australia
ron.steinfeld@monash.edu
[3]Department of Computing,
Faculty of Engineering and Physical Sciences,
University of Surrey, Guildford, Surrey, UK
shujun.li@surrey.ac.uk
[4]School of Electrical Engineering and Computer Science,
Science and Engineering Faculty,
Queensland University of Technology, Brisbane, Australia
josef.pieprzyk@qut.edu.au

July 13, 2016

Abstract

Human identification protocols are challenge-response protocols that rely on human computational ability to reply to random challenges from the server based on a public function of a shared secret and the challenge to authenticate the human user. One security criterion for a human identification protocol is the number of challenge-response pairs the adversary needs to observe before it can deduce the secret. In order to increase this number, protocol designers have tried to construct protocols that cannot be represented as a system of linear equations or congruences. In this paper, we take a closer look at different ways from algebra, lattices and coding theory to obtain the secret from a system of linear congruences. We then show two examples of human identification protocols from literature that can be transformed into a system of linear congruences. The resulting attack limits the number of authentication sessions these protocols can be used before secret renewal. Prior to this work, these protocols had no known upper bound on the number of allowable sessions per secret.

1 Introduction

A human identification (or authentication) protocol in this paper is defined as a protocol that enables a human using a terminal to prove his/her identity to a remote server in the presence of an observer.

*This is the full version of the paper with the title *On the Linearization of Human Identification Protocols: Attacks based on Linear Algebra, Coding Theory and Lattices* which appears in IEEE Transactions on Information Forensics and Security, volume 10, issue 8, pages 1643-1655. Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

In this work, we assume that the observer is a *passive* attacker who has access to the terminal, and the communication link between the terminal and the server. Moreover, the observer can also see the interaction between the human and the terminal. In contrast, an active attacker may additionally masquerade as the server and send messages of its choice to the user. However, active attacks are much more challenging to handle in the context of human identification protocols, and therefore, following the norm, we only focus on the passive observer. A secure human identification protocol lets a user authenticate to a server with the same shared password (secret) multiple times in the presence of the observer without the fear of leakage of the password which can be used for impersonation. Since the user's terminal is also in the hands of the adversary, the computations done by the user have to be mental. This is a severe constraint on the usability of such protocols as this results in an impractically high authentication time. Since the inception of this idea by Matsumoto and Imai in [1], protocol designers have attempted to decrease authentication time while providing high level of security.[1]

A general paradigm in the design of these protocols is the so-called k-out-of-n paradigm, wherein the shared secret is a set of k objects out of n publicly known objects. During authentication, the server sends a random challenge from some challenge space whose cardinality is a function of n. For instance, a challenge can be a random sample of n objects where each object is present in the sample with probability $\frac{1}{2}$. The users response is a publicly known function f of the secret and the challenge. An example response function is the modulo 2 sum of the number of secret objects present in the challenge. The set of possible responses, i.e., the response space, is generally much smaller. Due to a small response space, the protocol is iterated a number of times such that the probability that an attacker randomly guesses the answer is below a defined threshold. Given such a protocol, the goal of the attacker is to find the secret after observing a certain number of challenge-response pairs; the lesser, the better from an adversarial perspective. Notice that the only secret information is the set of secret objects; the cardinality of this set, i.e., k, is also public. The security of the protocol thus depends on the function f. A generic brute-force attack has time-complexity $\mathcal{O}(\binom{n}{k})$. As a first line of defence, this number should be high enough. This provides the first hurdle in constructing a practical human identification protocol, as the parameters n and k need to be large enough for security and small enough for usability.

Although protocol designers have attempted in vain to provide a secure *and* practical solution, we argue that the research in the security of such protocols is worth doing, due to the following main reasons:

1. The search of such protocols might eventually lead to a practical solution. One that is at least acceptable in higher risk situations. Research since the inception of the problem has come up with a number of interesting results in terms of the underlying mathematical problems (such as the learning in the presence of noise (LPN) problem used in [2]) used to construct f as well as generic attacks (for instance, the statistical attacks from Yan et al. [3]).

2. Human identification protocols can be modified to be used in resource-constrained devices. Such devices have one aspect common with humans: low memory and computational power. The Hopper and Blum (HB) human identification protocol [2] has been studied intensively for its application to RFID authentication [4]. To date, other proposed human identification protocols, the sum of k mins protocol from [2] and the Foxtail protocol from [5] to name a few, have not made progress in this line of research. Perhaps because their security has not been as comprehensively studied.

3. Such protocols can be used in a multi-factor authentication setting, where an auxiliary secure device can be given to the user. One example is the protocol from Catuogno and Galdi [6], where the device tells the user to send wrong responses to specific parts of the challenge in order

[1] A human identification protocol is essentially a conventional cryptographic identification protocol with the difference being that in practice the protocol steps of the prover have to be mentally computed by a human. Since the protocols discussed in this paper have been proposed in the research context of human identification protocols, we shall refer to them as such. Of course, the results still apply if the human prover is replaced with a device.

to *confuse* the observer. Under multi-factor authentication, we can use such auxiliary devices as computational aid for the human, thus making these protocols more practical.

With this in mind, we focus on an important goal of protocol designers: to increase the number of possible authentication sessions with a given secret. Since the protocols from Hopper and Blum in [2], an important goal has been to propose protocols that are secure against observations for at least $\mathcal{O}(n^2)$ sessions. One way to achieve this is to ensure that the function f cannot be written as a system of linear equations or congruences in n. If this is not ensured, then one can, for instance, use Gaussian elimination to obtain the unique solution (secret) after observing about n challenge-response pairs.

Our Contributions. In this paper, we look at the design goal of *non-linearity* of f in detail. We first show how a system of linear congruences modulo some integer $d \geq 2$ can be attacked using different techniques from algebra, lattices and coding theory. This is important since although Gaussian elimination can be used to obtain the secret after observing n challenge-response pairs, the cardinality of the secret set k being less than n means that other attacks may be possible for a much smaller number of observations. We show that this is indeed possible for small values of k and n. We then study two protocols from literature: the Catuogno and Galdi (CG) protocol from [6] and the modified form of the Foxtail protocol [5] from [7],[2] and show how they can be attacked by transforming them into a system of linear congruences. Both of these protocols were constructed with non-linearity of the response function in mind. Neither protocol has a known upper bound on the number of allowable sessions for the recommended parameters. The attacks shown here impose this limit. More specifically, we show that the secret in the CG protocol can be obtained by observing only 80 sessions with the recommended parameters. And in the case of the modified Foxtail protocol, we show that it can be used for fewer than 500 sessions; a number less than the 711 mark obtained by the statistical attack from Yan et al. [3] on the original Foxtail protocol [5] due to which the protocol was modified in [7].

Organization. The rest of the paper is organized as follows. Section 2 describes the related work, followed by preliminaries and background on human identification protocols in Section 3. In Section 4 we present a detailed analysis of possible attacks on a system of n linear congruences over integers modulo $d \geq 2$ in the n-element unknown binary vector $\mathbf{x}$ of Hamming weight k. The goal of this section is to find feasible ways to obtain $\mathbf{x}$ with fewer than n congruences. Equipped with this knowledge, we show an attack on the C protocol from [6] in Section 5 by describing how the protocol can be represented as a system of linear congruences. In Section 6 we show how another protocol, namely the Foxtail protocol [5, 7], can also be represented as a system of linear congruences. Here again, the analysis from Section 3 proves useful in determining possible feasible attacks once a system of linear congruences is obtained. We discuss some of the implications of our results in Section 7 and present concluding remarks in Section 8.

2 Related Work

Consistent with the focus of this paper, we mostly limit this brief literature overview to proposals for human identification protocols constructed to avoid representation as a system of linear equations or congruences. The idea of authenticating a human in the aforementioned threat model was first put forward by Matsumoto and Imai [1], where the authors proposed the first human identification protocol. The protocol was broken in [8]. The initial research by Matsumoto and Imai has followed a string of proposals for human identification protocols [9–13] which were broken by subsequent attacks [3, 14–17]. Most of these protocols were based on an ad-hoc design. By contrast, Matsumoto proposed another protocol based on linear algebra which can be used for $\mathcal{O}(n)$ sessions after which

[2] The original protocol was found vulnerable to statistical attacks by Yan et al. [3]. We therefore chose the version which is proven secure against such attacks as well as all other known attacks.

Gaussian elimination can be used to find the unique secret [18]. The advantage of this protocol over the others is that the security of the protocol can be argued against what is known about the underlying mathematical problem. Since then various attempts have been made to construct protocols that avoid representation as a linear system of equations or congruences. These include the HB protocol and the sum of k mins protocol from Hopper and Blum [2], the Foxtail protocol from Li and Shum [5], the Asghar, Pieprzyk and Wang (APW) protocol [19], and the protocols from Catuogno and Galdi [6] to name a few. Among these protocols the Foxtail protocol was shown to be susceptible to a counting based statistical attack from Yan et al. [3], after which a fixed version of the protocol was proposed in [7]. The attack from Yan et al. applies to the original Foxtail protocol due to a contrived method of generating the challenges. Once that is removed, statistical attacks are no longer applicable. Likewise, the same holds for the other protocols mentioned above. Apart from this statistical attack, there are no known passive attacks on these protocols, barring the generic brute force and the meet-in-the-middle attack of complexity $\mathcal{O}(\binom{n}{k/2})$ [2]. Recently, a timing based side-channel attack of time complexity $\sim 2^{50}$ has been shown on the HB protocol by Čagalj and Perković [20], which exploits the difference in response times which are proportional to the cognitive load on humans when computing challenges of varying complexity. This attack is complementary to the algebraic attacks discussed in this paper.

3 Preliminaries and Background

3.1 Notation

Let $\mathbf{x}$ denote a column vector. Then $\mathbf{x}^T$ denotes a row vector, where T stands for transpose. Where there is no chance of ambiguity, we shall refer to column or row vectors as simply vectors. For any two vectors $\mathbf{x}$ and $\mathbf{y}$ having the same number of elements, $\mathbf{x} \cdot \mathbf{y}^T = \mathbf{x}^T \cdot \mathbf{y}$ denotes their dot product. The length or Euclidean norm of a vector $\mathbf{x}$, denoted $\|\mathbf{x}\|$, is defined as $\sqrt{\sum_i x_i^2}$, where x_i is the ith element of $\mathbf{x}$. If $\mathbf{x}$ is binary, its Hamming weight, denoted $\mathsf{wt}(\mathbf{x})$, is defined as $\|\mathbf{x}\|^2$; in other words, the sum of the elements of $\mathbf{x}$. Let $d \geq 2$ be an integer. The set $\mathbb{Z}_d$ denotes the set of integers modulo d. $\mathbb{Z}_d^n$ is the set of all n-element vectors with entries from $\mathbb{Z}_d$. For a square matrix A, let $\mathsf{det}(A)$ denote its determinant. For any $n \geq 1$, I_n is the $n \times n$ identity matrix. I_n^M is the mirror image of the identity matrix I_n, i.e., the $n \times n$ matrix obtained by vertically flipping the diagonal of I_n. For $m, n \geq 1$, $\mathbf{0}_{m,n}$ and $\mathbf{1}_{m,n}$ represent the $m \times n$ matrices of all zeros and all ones, respectively. When $m = n$, we shall simply drop one of the two subscripts.

3.2 Background on Human Identification Protocols

We model human identification protocols as challenge-response protocols between a human user (the prover) and a server (the verifier). An authentication session, or simply a session, is a sequence of challenge-response pairs followed by a decision from the server. Given a challenge, the user response is a function of the challenge and a secret shared with the server (the verifier). Following convention, we shall call the shared secret, the password. Generally, the set of all possible challenges and responses, respectively termed the challenge space and the response space, is public information. Let p be the probability that a response randomly selected from the response space is the correct response to a given challenge, where the probability is taken over all possible responses, challenges and passwords. The server might need to send multiple challenges so that the probability of an impersonator is below a certain confidence level. The number of rounds r in a protocol is defined as the number of challenge-response pairs required to obtain a certain confidence level. Let γ be the confidence parameter, which is defined as follows: for $i \geq 1$, $\gamma = i$ if the success probability of the random response attack described above is $\frac{1}{2^i}$ in a *session*. Thus, the number of rounds r in a session should satisfy

$$\frac{1}{2^\gamma} \geq p^r \Rightarrow r \geq \frac{\gamma}{\log_2 \frac{1}{p}}. \tag{1}$$

To better understand the constructions of human identification protocols and to facilitate the ensuing analysis, we begin with the example of a protocol which is a modified version of the protocol from Hopper and Blum [2] and also resembles the protocol from Matsumoto proposed in [18]. The user and the server choose a binary vector $\mathbf{x}$ of n elements as the shared password. During authentication, the server generates a random binary vector $\mathbf{c}$ of n elements. The user computes the dot product $\mathbf{c}^T \cdot \mathbf{x}$ modulo 2, and sends the response bit to the server. Since the probability that a random response is the correct response to a given challenge is $\frac{1}{2}$, we see from Eq. (1) that for a desired confidence parameter γ, the number of rounds per session is precisely γ. To help the user memorize the password, the weight of the vector $\mathbf{x}$ is fixed at k, where k is much smaller than n. The protocol can be generalized to any modulus $d \geq 2$, in which case the secret remains a binary vector but the challenge is now a random vector from $\mathbb{Z}_d^n$ and the response is the dot product modulo d. The number of rounds in this case is given by $\left\lceil \frac{\gamma}{\log_2 d} \right\rceil$.

A brute force attack of time complexity $\mathcal{O}(\binom{n}{k})$ can find the password $\mathbf{x}$ given enough challenge-response pairs. To estimate the number of challenge response pairs we see that given one challenge-response pair, a $\frac{1}{d}$ fraction of the number of possible passwords $\binom{n}{k}$ give the same response as the password $\mathbf{x}$. Given m challenge-response pairs the fraction of total passwords agreeing with the m responses of $\mathbf{x}$ is $\frac{1}{d^m}$. Here we have assumed a uniform distribution over the challenge, response and password spaces. The expected number of challenge-response pairs m required to obtain a unique $\mathbf{x}$ is then

$$m > \log_d \binom{n}{k} = \frac{\log_2 \binom{n}{k}}{\log_2 d}. \tag{2}$$

Since this number is much smaller than n, it is generally ensured that the parameters k and n are large enough so that the brute force attack is infeasible. Given the infeasibility of the brute force attack, the adversary needs to look at other ways to attack this system. Notice that the bound above is the minimum necessary to obtain a unique password. Below this we will have multiple candidates for the password.

4 Attacking a System of Linear Congruences

Under the aforementioned threat model, the observer can record any number of challenge-response pairs $(\mathbf{c}_i, r_i)$ gathered over multiple sessions. After gathering m challenge-response pairs of the protocol described in the previous section, the following system can be constructed:

$$C\mathbf{x} \equiv \mathbf{r} \bmod d,$$

where C is the challenge matrix, built from m row vectors $\mathbf{c}_i^T$, $\mathbf{r}$ is the response vector whose entries are the m responses, and $\mathbf{x}$ is the unknown binary vector with Hamming weight k. We now look at different ways, other than brute force, to attack this system of linear congruences to obtain the binary vector $\mathbf{x}$ with as small a value of m as possible. We begin by discussing the inherent hardness of the problem from a complexity theoretic viewpoint.

4.1 Hardness of the Problem

Finding $\mathbf{x}$ from this system of congruences was shown to be NP-Complete in [21] in the binary case (when $d = 2$) and an extension of the result over all finite fields was shown by Barg [22]. The same reduction used in [21] can be used to show that the problem is NP-Complete for all d. In [23], Downey et al. showed that the problem is fixed parameter intractable in the binary case. This notion means that there is no algorithm which can solve this problem in time $f(k)n^c$, where c is a constant and $f(k)$ is an arbitrary function of k [23]. The problem was shown to be NP-Complete and fixed parameter intractable for all d in [24]. These results shed light on the inherent hardness of this problem, at least in the worst case. However, as we shall see, in practice, in the average case, the problem can be easier for small values of k and n.

When $d = 2$, this problem is also known as learning k parities without noise in the *attribute efficient* setting [25]. The reader might also find this problem somewhat similar to the Short Integer Solution (SIS) problem [26], which is stated as follows: given a uniformly random $m \times n$ matrix C with entries from $\mathbb{Z}_d$, where d and n are polynomially bounded in m, find an m-element non-zero integer vector $\mathbf{x}$ such that $C\mathbf{x} \equiv \mathbf{0} \bmod d$ and $\|x\| \leq k$. Here k should be large enough so that a solution exists. In contrast the problem considered in this paper uses much smaller values of d, asks for a binary solution and has a non-zero response vector.

4.2 Gaussian Elimination

Since there are n unknowns in $\mathbf{x}$, the observer can collect n challenge-response pairs to obtain a unique solution to the above system of linear congruences using the well-known efficient method of Gaussian elimination. However, uniqueness through Gaussian elimination is only guaranteed if the n rows (or columns) in C are linearly independent. Since each $\mathbf{c}_i$ in C is generated randomly, the n vectors may not always be linearly independent. But as we discuss in Appendix A, with high probability the n rows are linearly independent. Even if that is not the case, the attacker can observe a little more than n challenge-response pairs and choose the n that are linear independent. We also find bounds on the expected number of challenge-response pairs required to obtain a set of linearly independent vectors in Appendix A. Thus, we shall assume that the attacker can solve the above system of linear congruences to obtain the password $\mathbf{x}$ with high probability after observing n challenge-response pairs. Thus the first constraint on the usage of this protocol is that it cannot be used for more than n challenge-response pairs or $\frac{n \log_2 d}{\gamma}$ sessions. But comparing this with Inequality (2), we see that Gaussian elimination is not optimal in terms of the number of challenge-response pairs m required to obtain the password (since $\log_d \binom{n}{k}$ is less than n). We therefore look at other feasible ways to find the password in the hope to find a more optimal solution.

4.3 Lattice-based Attack

We can represent the system of linear congruences as part of a lattice and then apply lattice basis reduction algorithms to find the password. More specifically, suppose the observer has m challenge-response pairs. Let $c_{i,j}$ denote the jth element of the ith challenge, where $1 \leq i \leq m$ and $1 \leq j \leq n$. Let $\mathcal{L}$ be the lattice with the following basis vectors:

$$\begin{array}{lcl}
\mathbf{b}_1 & = & (1 \quad 0 \quad 0 \quad \ldots \quad 0 \quad \mu c_{1,1} \quad \mu c_{2,1} \quad \ldots \quad \mu c_{m,1}) \\
\mathbf{b}_2 & = & (0 \quad 1 \quad 0 \quad \ldots \quad 0 \quad \mu c_{1,2} \quad \mu c_{2,2} \quad \ldots \quad \mu c_{m,2}) \\
 & \vdots & \\
\mathbf{b}_n & = & (0 \quad 0 \quad 0 \quad \ldots \quad 1 \quad \mu c_{1,n} \quad \mu c_{2,n} \quad \ldots \quad \mu c_{m,n}) \\
\mathbf{b}_{n+1} & = & (0 \quad 0 \quad 0 \quad \ldots \quad 0 \quad \mu d \quad 0 \quad \ldots \quad 0) \\
\mathbf{b}_{n+2} & = & (0 \quad 0 \quad 0 \quad \ldots \quad 0 \quad 0 \quad \mu d \quad \ldots \quad 0) \\
 & \vdots & \\
\mathbf{b}_{n+m} & = & (0 \quad 0 \quad 0 \quad \ldots \quad 0 \quad 0 \quad 0 \quad \ldots \quad \mu d) \\
\mathbf{b}_{n+m+1} & = & (0 \quad 0 \quad 0 \quad \ldots \quad 0 \quad \mu r_1 \quad \mu r_2 \quad \ldots \quad \mu r_m),
\end{array}$$

where μ is a positive number which shall be explained later. The lattice $\mathcal{L}$ is then the set of *integer* linear combinations of the above basis vectors. This lattice is a modified form of the lattice from [27] which is derived from the lattice from [28]. Consider the $(n + m)$-element vector

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n & 0 & \cdots & 0 \end{pmatrix},$$

which is the same as $\mathbf{x}$ except that it is padded with m zeros at the end. Abusing notation we shall also call it $\mathbf{x}$. This vector belongs to $\mathcal{L}$. To see this, first define the quotients

$$\sum_{j=1}^{n} c_{i,j} x_i = q_i d + r_i$$

for $1 \leq i \leq m$. Such q_i's should exist because $\mathbf{c}_i^T \cdot \mathbf{x} \equiv r_i \bmod d$. Then we see that

$$\sum_{i=1}^{n} x_i \mathbf{b}_i - \sum_{i=1}^{m} q_i \mathbf{b}_{n+i} - \mathbf{b}_{n+m+1} = \mathbf{x},$$

is indeed part of $\mathcal{L}$. Notice that $\mathbf{x}$ is a short vector in $\mathcal{L}$ with length (Euclidean norm) $\sqrt{k}$. Given a lattice reduction algorithm, such as the Lenstra-Lenstra-Lovász (LLL) algorithm [29], we can find a short vector whose length (Euclidean norm) is within a range which is exponential in the length of the shortest non-zero vector in $\mathcal{L}$. Note that LLL algorithm only runs in polynomial time given such an exponential guarantee [30]. Thus, there is a trade-off between running time and the range of length of a short vector obtained. On the other hand, we can use the technique from [28] to show that if m satisfies Inequality (2) then with high probability the vectors $\mathbf{x}$ and $-\mathbf{x}$ are the *only* short vectors in $\mathcal{L}$ within the exponential bound mentioned above [27]. The LLL algorithm is then guaranteed to find the vector $\mathbf{x}$ in polynomial time. The parameter μ above plays a central role in determining the bound on this probability by ensuring that the short vectors in $\mathcal{L}$ are within this length [27, 28]. The probabilistic result from [27] about finding the vector $\mathbf{x}$ holds for a rather large d, and on top of that it is an upper bound on the probability. Thus, to see if the LLL algorithm can indeed find the vector $\mathbf{x}$ in practice (in our case) is through actual experiments. Note that our persistence in using the LLL algorithm is just representational and the algorithm can be substituted by any other lattice reduction algorithm that is efficient in practice. An example being the block Korkin-Zolotarev (BKZ) algorithm [31]. BKZ is likely to give slightly better results at the expense of running-time.

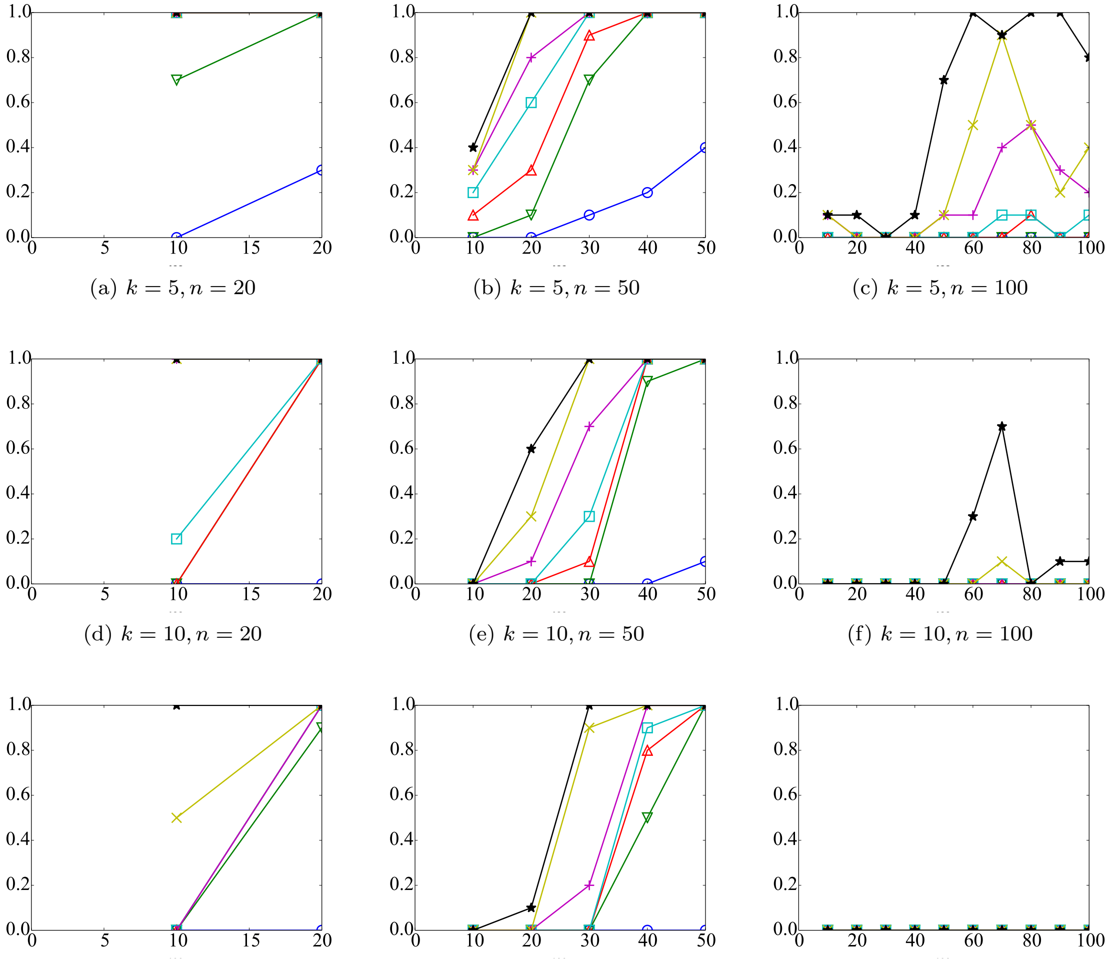
We implemented the above lattice in `Sage`,[3] an open-source mathematics software, and used the implementation of LLL available therein to find $\mathbf{x}$ with the default settings. The results are as shown in Figure 1. For each value of the tuple (d, k, m, n, μ) the algorithm was run 10 times, each time with a new random system of congruences. Although the number of runs is not enough for high precision, it is sufficient for the main purpose of these simulations which is to show the difference in performance of the lattice-based attack with changes in the values of the variables involved. The fraction of times $\mathbf{x}$ was found was noted as the success percentage. For each tuple (d, k, m, n), μ was assigned the values from 10 to 100 in steps of 10 with only the value of μ corresponding to the highest success rate retained. The value of m ranged from 10 to n, again in steps of 10. As expected, as m approaches n, the success rate increases. Furthermore, for m much smaller than n, the success rate is still high. The success rate is also high for smaller values of k. On the other hand, smaller values of d give the worst success rate. In particular, the success rate is lowest for $d = 2$. Finally, notice that as k and n increase the success rate decreases, with $k = 15$ and $n = 100$ giving a success rate of 0.0 for all values of d. This is expected as lattice reduction algorithms do not perform well for higher dimensions. In our case the dimension of the lattice is $n + m + 1$. Regardless, we can see that small values of k and n are susceptible to lattice based attacks. We note that there is an improved lattice from [30] which was modified in [27] to obtain a better probability result in terms of the required number of challenge-response pairs, m. The first $n + m$ basis vectors of this lattice are the same as $\mathcal{L}$. The only difference is the last basis vector, which is given by

$$\mathbf{b}_{n+m+1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \mu r_1 & \mu r_2 & \cdots & \mu r_m \end{pmatrix}.$$

We can then see that the vector

$$\begin{pmatrix} x_1 - \frac{1}{2} & x_2 - \frac{1}{2} & \cdots & x_n - \frac{1}{2} & 0 & \cdots & 0 \end{pmatrix},$$

[3] http://www.sagemath.org/



(a) $k = 5, n = 20$ (b) $k = 5, n = 50$ (c) $k = 5, n = 100$

(d) $k = 10, n = 20$ (e) $k = 10, n = 50$ (f) $k = 10, n = 100$

(g) $k = 15, n = 20$ (h) $k = 15, n = 50$ (i) $k = 15, n = 100$

Figure 1: The success of the LLL algorithm against different values of d, k, m and n. The x-axis in the graphs represents m. The y-axis is the success percentage. Legend: ○ $d = 2$, ▽ $d = 3$, △ $d = 4$, □ $d = 5$, + $d = 7$, × $d = 10$, ★ $d = 15$.

is part of the resulting lattice from which $\mathbf{x}$ can be easily extracted. The length (Euclidean norm) of this vector is far less than its counterpart in the lattice $\mathcal{L}$. So, one can see why such a lattice should give the probabilistic assurance for a smaller value of m. However, to convert this lattice into an integer lattice, we need to multiply the basis by 2. The resulting lattice then has larger short vectors than its non-integer counterpart. As a result, we were not able to find better results from this lattice using `Sage`. Regardless, we observe that with value of n much larger than 100 the system of linear congruences under discussion is unlikely to be susceptible to lattice based attacks.

4.4 Coding Theory based Attack

If we let d be a prime, we can view the $m \times n$ matrix C as a random parity check matrix,[4] and $C\mathbf{x} \equiv \mathbf{r} \bmod d$ can be thought of as the syndrome of some codeword, where $\mathbf{x}$ is viewed as an error

[4] Provided, of course, that the matrix has rank m, which as we have seen is true with high probability.

vector of Hamming weight k. Finding $\mathbf{x}$ is thus the classic problem of *syndrome decoding* of random linear codes. Since the parity check matrix C is generated at random, there is no underlying code structure. In such a case, the best algorithms to find a solution to this problem belong to the class of algorithms called information set decoding [32]. Since the introduction of the technique by Prange [33], many improved variants of information set decoding have been proposed. Asymptotically, these algorithms have exponential time complexities, but for smaller values of n and k, information set decoding can be feasible. Here we take the variant described in [34] as an example.

We first randomly permute the matrix C and transform it into the following form

$$C = \begin{pmatrix} I_{m-l} & C_1 \\ \mathbf{0}_{l,m-l} & C_2 \end{pmatrix},$$

where I_{m-l} is the $(m-l) \times (m-l)$ identity matrix, $\mathbf{0}_{l,m-l}$ is the $l \times (m-l)$ zero matrix, C_1 is a $(m-l) \times (n-m+l)$ matrix, and C_2 is a $l \times (n-m+l)$ matrix. Notice that the configuration above can be obtained with high probability [35, §4, p. 46]. Furthermore, the system of congruences has the property that permutations and elementary row operations can be done without affecting the unknown $\mathbf{x}$ [35, Lemma 3.1.3., §3, p. 27]. The only change can be a permuted $\mathbf{x}$ but by applying inverse permutations we can achieve the original form after a solution has been found. After C has been transformed in the above form, it is hoped that p of the k "errors" from $\mathbf{x}$ correspond to the last $n-m+l$ columns (corresponding to columns of C_1 and C_2), and the remaining $k-p$ errors belong to the first $m-l$ columns. If we write the syndrome $\mathbf{r}$ as $\begin{pmatrix}\mathbf{r}_1 & \mathbf{r}_2\end{pmatrix}^T$ then we find an $(n-m+l)$ element vector $\mathbf{x}_2$ (corresponding to the columns of C_1 and C_2) of Hamming weight p such that $C_2\mathbf{x}_2$ equals $\mathbf{r}_2$ modulo d. If we find such a vector, we subtract $\begin{pmatrix}C_1\mathbf{x}_2 & C_2\mathbf{x}_2\end{pmatrix}^T$ from $\mathbf{r}$. This gives us a vector that is *zero* in its last $m-l$ elements. If the resulting vector has weight $k-p$ we have found the solution.

One way to find the vector $\mathbf{x}_2$ of weight p is to divide the search space in half by considering vectors of weight $\frac{p}{2}$ and finding collisions [34, 36]. The total work required by the algorithm can be optimised by choosing p which minimizes the work factor (WF). Of particular interest is the case when $p = 0$ (and hence $l = 0$). This corresponds to the original (plain) information set decoding introduced by Prange [33]. This variant essentially randomly permutes C and then transforms it so that the first m columns of C become the identity matrix. If all k errors of $\mathbf{x}$ correspond to the first m columns, we have the solution. An estimate of the work required in this case is given by

$$\frac{\binom{n}{k}}{\binom{m}{k}},$$

which is the reciprocal of the probability that the k errors are confined to the m columns. Notice that since we want to find a unique $\mathbf{x}$ we choose an $m > \log_d \binom{n}{k}$.

Intuitively, if the Hamming weight of $\mathbf{x}$ is small, this basic approach should be good enough. We see that this is indeed true for the parameter values under consideration. We take $d = 2$ as an example and choose $n = 100$ and $k = 15$. For the work factor derived in [34] we see that for all m in the range $(\log_2 \binom{100}{15} \approx 57.8, 100)$ the best result is obtained when $p = 0$ which corresponds to plain information set decoding. Figure 2 shows the work factor (WF) in $\log_2$-scale. We see that the value of WF is always less than 2^{14}. Thus, for small values of n and k, one can use information set decoding to find an $\mathbf{x}$ with an m much smaller than n. We note that, following the discussion in Appendix A, we can use the above technique to find an $\mathbf{x}$ for any d not necessarily prime, also with a very high probability.

5 Linearization of the CG Protocol

Since many human identification protocols have been constructed to avoid linearity [2, 5], it seems rather pedantic to analyze the problem of finding a solution to the system of linear congruences in such detail. Still, as we shall show, it is possible to transform some of these protocols as a system

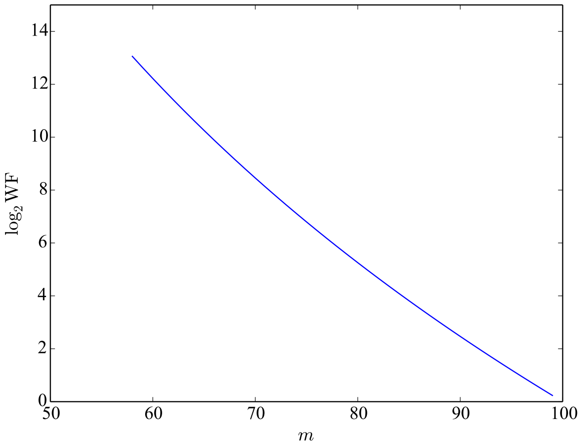


Figure 2: The logarithmic plot of the work factor (WF) of the information set decoding algorithm described in [34] against m.

of linear congruences. Such transformation may not be clear at first glance. We highlight this using a protocol from Catuogno and Galdi [6]. In [6] Catuogno and Galdi described several protocols. Of most interest is the "selective wrong-correct answer" protocol, which we refer to as the CG protocol. This protocol seems to avoid all attack strategies considered in [6] (more on this in the following). We briefly describe the protocol here.

The user and the server share a set of k objects out of n as the secret. We call these k objects the pass-objects. A challenge from the server consists of two rows. Each row contains $\frac{n}{2}$ objects. The user notes the row numbers in which the k pass-objects appear in the challenge. For exactly $\frac{k}{2}$ pass-objects the user sends the correct row numbers as the response. For the remaining $\frac{k}{2}$ pass-objects the user sends the *wrong* row number as the response. The server accepts the user by verifying the received response through its own copy of the shared secret. Note that all k responses are sent in the clear. The set of $\frac{k}{2}$ pass-objects for which the wrong responses are to be sent changes for every challenge. Catuogno and Galdi argue that this can be achieved through an auxiliary device possessed by the user which specifies which pass-objects belong to the wrong response set [6]. Note that this device does not need to know the pass-objects. It only knows the parameter value k, which is public. This means that the wrong responses are deterministic and hence the user cannot arbitrarily choose $\frac{k}{2}$ wrong responses.

Catuogno and Galdi recommended the parameters $n = 80$ and $k = 15$. Exactly $\lceil \frac{k}{2} \rceil = 8$ of the responses are wrong and $\lfloor \frac{k}{2} \rfloor = 7$ responses are correct. A random response has probability 2^{-15} of being successful which is higher than the security of the commonplace PIN based authentication. Notice that since the attacker does not know which of the k answers are wrong, it is not possible to employ a brute force strategy in which the attacker attempts to find which of the possible $\binom{n}{k}$ secrets yield a consistent response across different challenges (See [6] for details). Here we show how to obtain a system of linear congruences modulo 2 from the challenge-response pairs obtained from the protocol. We represent a challenge as an n-element vector $\mathbf{c}$ whose ith entry is 0 if the ith object is in the first row, and 1 otherwise. we represent the unknown set of k pass-objects as an n-element binary vector $\mathbf{x}$ of Hamming weight k. For the given challenge $\mathbf{c}$, we represent the vector corresponding to the correct responses by $\mathbf{x}_1$ and the one corresponding to the wrong responses by $\mathbf{x}_2$. Note that both these vectors are n-element vectors of Hamming weight $\frac{k}{2}$. Furthermore, these vectors are different over different challenges pertaining to the condition

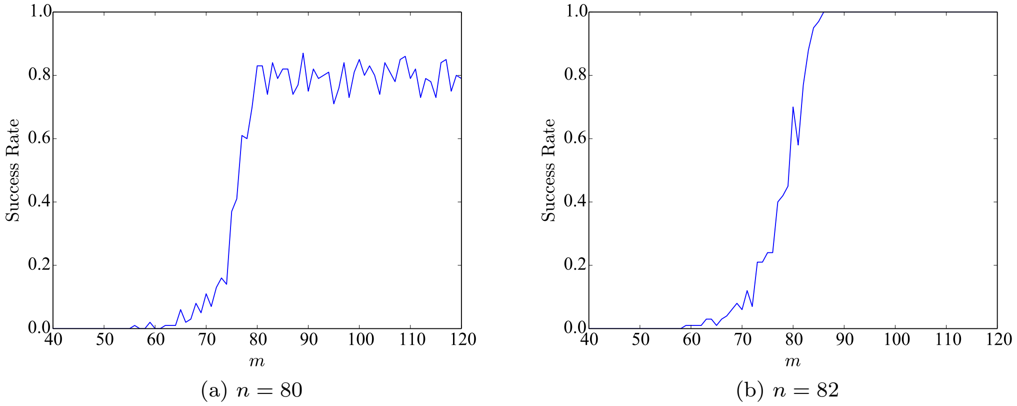
$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2.$$

In other words, even though the vectors are different over different challenges, their sum is constant, namely, the secret vector $\mathbf{x}$. Now observe that sending a wrong response from the challenge $\mathbf{c}$ is the same as sending the correct response from the vector $\mathbf{1} - \mathbf{c}$, where $\mathbf{1}$ is the n-element vector all entries of which are 1. Let r represent the *sum* of all the responses. Note that $\mathbf{1} \cdot \mathbf{x}_2^T = \frac{k}{2}$. Then

$$\begin{aligned}
\mathbf{c} \cdot \mathbf{x}_1^T + (\mathbf{1} - \mathbf{c}) \cdot \mathbf{x}_2^T &= r \\
\Rightarrow \mathbf{c} \cdot \mathbf{x}_1^T - \mathbf{c} \cdot \mathbf{x}_2^T + \mathbf{1} \cdot \mathbf{x}_2^T &= r \\
\Rightarrow \mathbf{c} \cdot \mathbf{x}_1^T - \mathbf{c} \cdot \mathbf{x}_2^T &= r - \frac{k}{2} \\
\Rightarrow \mathbf{c} \cdot \mathbf{x}_1^T - \mathbf{c} \cdot \mathbf{x}_2^T &\equiv r - \frac{k}{2} \bmod 2 \\
\Rightarrow \mathbf{c} \cdot \mathbf{x}_1^T + \mathbf{c} \cdot \mathbf{x}_2^T &\equiv r + \frac{k}{2} \bmod 2 \\
\Rightarrow \mathbf{c} \cdot \mathbf{x}^T &\equiv r + \frac{k}{2} \bmod 2 \qquad (3)
\end{aligned}$$

Since r and $\frac{k}{2}$ are known, we can reduce them modulo 2 for each challenge, resulting in a system of linear congruences. Given $m \approx n = 80$ challenge-response pairs, we can then apply Gaussian elimination to obtain the secret $\mathbf{x}$. We implemented the CG protocol. From the challenge-response pairs thus obtained we constructed this system in `Sage` and used the `solve_right()` method to obtain the solution. For each value of m, we ran 100 simulations each time with a new secret $\mathbf{x}$ and random challenges. The result is shown in Figure 3a. After $m = 77$ challenge-response pairs we were able to obtain the secret more than 50 percent of the time. The peak is achieved at $m = 80$ where we find the secret more than 80 percent of the time. However, the success rate is never 100 percent. This is due to the fact that the $m \times n$ matrix C obtained from m challenges described above has the peculiar characteristic that the highest possible rank is always $n - 1$ when $\frac{n}{2}$ is even (recall that this matrix has rows which have exactly $\frac{n}{2}$ ones and $\frac{n}{2}$ zeros). Therefore, any number of $m >= n$ rows are linearly dependent. Gaussian elimination requires n linearly independent rows (or columns) to obtain a unique solution. Notice that Gaussian elimination still works when the rank is less than n. The only difference is that we get multiple solutions. We, therefore, do not get a unique solution from this linear system of congruences when $\frac{n}{2}$ is even. On the other hand, if $\frac{n}{2}$ is odd, the highest achievable rank is n, which means a unique solution is possible. This is illustrated in Figure 3b, where we use $n = 82$ and get the secret all the time after $m = 85$ challenge-response pairs. The proof of why the rank of the matrix C is related to $\frac{n}{2}$ being even or odd is given in Appendix B.

There are a number of interesting observations. First, the CG protocol uses one round per session. Thus, for a given confidence level $\gamma \geq 1$, instead of $\frac{n}{\gamma}$ sessions achieved by the protocol described in Section 3, we have a protocol that achieves approximately n sessions. If Gaussian elimination is optimal we can use the protocol for about 70 sessions, which for the same confidence level $\gamma = 15$ would give only 6 sessions of the protocol described in Section 3. Below $m = 70$, one may apply the information set decoding attack mentioned in the last section. The attack should work with m greater than $\log_2 \binom{80}{15} \approx 52.56$. Since n is small we see that the complexity of the attack is very low. We, however, did not implement the attack, as the number of allowable sessions with the Gaussian elimination based attack is already quite low. The second observation is that since this is a two-factor authentication system, learning the secret does not allow the adversary to readily impersonate the user. This is true since the attacker does not know which set of answers should be inverted. The last observation is that if instead of exactly $\frac{k}{2}$ wrong responses, the number of wrong responses varies over challenges, we cannot write it as a system of linear congruences described above. This is a possible fix of the protocol against the attack described here.



(a) $n = 80$

(b) $n = 82$

Figure 3: Success rate of finding $\mathbf{x}$ from the system of linear congruences built from Eq. (3).

Lastly, in the paper from Catuogno and Galdi [6] and the follow-ups [37, 38], much attention has been given to show the feasibility of finding the secret $\mathbf{x}$ in the above protocol using the so-called SAT solvers. This line of attack works by representing the system in the form of boolean satisfiability clauses and then running a SAT solver to find the secret. The analysis from Catuogno and Galdi [6, 37, 38] shows that SAT solvers were only able to find the secret in the above protocol for much smaller values of k and n and the complexity grows exponentially. This conclusion is perhaps not surprising since boolean satisfiability in general is an NP-Complete problem. However, despite this, SAT solvers are known to work efficiently in practice. In the following we attempt to explain why the SAT based attack did not work against this protocol.

5.1 SAT Solvers and Analysis of Weinshall's Protocol

In the context of human identification protocols, SAT solvers have been used as a sort of optimised brute force strategy to find the secret. In particular, if a weakness is not known in a protocol, the SAT solver is not likely to find it, as the satisfiability clauses are constructed from what is known apriori about the information leakage of the protocol. Thus, for instance, if the only brute-force way to find the secret is to check all possible $\binom{n}{k}$ combinations, then SAT solvers are not likely to find the secret in feasible time.

The interest in SAT solvers in the context of human identification protocols stems from the cryptanalysis of the Cognitive Authentication Scheme (CAS), proposed by Weinshall in [9], by Golle and Wagner [14]. By representing CAS as a series of boolean satisfiability clauses Golle and Wagner were able to find the password after a small number of observed sessions. Here we focus on the so-called *high complexity* variant of CAS. The system uses n pictures out of which k are the user's secret. The proposed parameter values are $n = 80$ and $k = 30$. A naive brute force strategy has complexity $\binom{80}{30} \approx 2^{73}$, which is arguably infeasible. The protocol is briefly described here.

The challenge consists of a grid of $r = 8$ rows and $c = 10$ columns giving us a set of 80 images which are randomly permuted. See Figure 4 for a sketch of the grid. The user starts from the top-left corner of the grid and does the following. If the picture in the current cell is one of the k secret images, the user moves down, otherwise the user moves right. This procedure is continued until the edge of the grid is reached. This could be either the bottom or right of the grid. Each cell at the edge is associated with a random binary number, with the right bottom cell containing two numbers, one at the bottom and one at the right.[5] The user sends the number thus reached as the response.

[5] The number in general can be from a set $\{0, 1, \ldots, b\}$ with $b \geq 2$. In our example we consider the case when $b = 2$ without loss of generality.

As we can see there is some additional information leakage evident from the way the protocol is constructed. For instance, the starting cell is the top-left cell in the grid. Golle and Wagner [14] exploited this and other information that could be obtained from the protocol to build satisfiability clauses, and ran a SAT solver over these clauses to find a satisfying solution. The complexity of the attack, in terms of possible values of clauses, is proportional to the number of possible paths from the top-left corner to one of the *exit points*. Here we analyze the number of possible paths. A path is a route traversed from the top-left corner of the grid to one of the exit points. Figure 4 represents the paths and the grid in the form of a graph, which we denote by $\mathcal{G}$. We shall use this equivalent representation to enumerate the number of paths. Let r and c denote the number of rows and columns in the grid. Then $P(r, c)$ denotes the total number of possible paths from the node labelled 0 in the figure to one of the $r + c$ exit points. Note that an exit point is part of the path and can be viewed as a leaf node. We see that $P(r, c)$ is equal to the sum of paths in the bottom child and the right child graphs of $\mathcal{G}$. By looking at these child graphs again, we see that we come across the same problem of finding paths in a reduced version of the graph $\mathcal{G}$. Thus, we have the recurrence relation

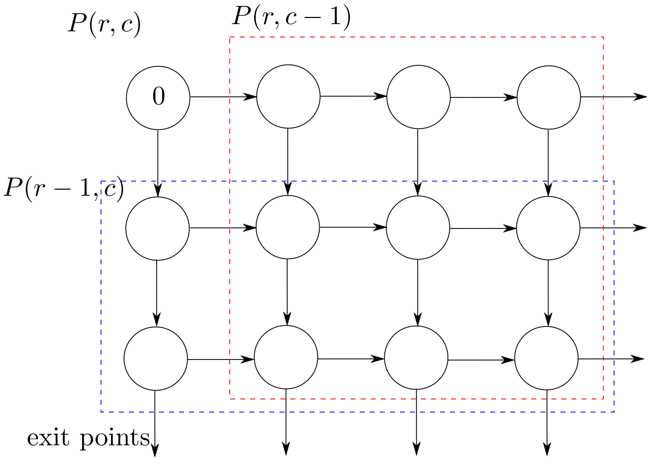


Figure 4: The number of paths $P(r, c)$ in the graph $\mathcal{G}$.

$$P(r, c) = P(r, c-1) + P(r-1, c), \tag{4}$$

with the boundary conditions

$$\begin{aligned} P(0, j) &= 1, \text{ for } 0 \leq j \leq c, \\ P(i, 0) &= 1, \text{ for } 0 \leq i \leq r. \end{aligned} \tag{5}$$

These boundary conditions essentially mean that we have reached the exit points. In Appendix C we show that this recurrence relation has the solution

$$P(r, c) = \binom{r+c}{r} = \binom{r+c}{c}.$$

This result can also be derived from a simple combinatorial argument as follows. First consider the exit points to the bottom. Each such exit point contains exactly r *down steps*. For each value of $0 \leq i \leq c-1$ there are exactly $\binom{r-1+i}{i}$ possible ways i steps to the right exist in the path exiting to the bottom (the last step is always a downward step, hence the use of $r - 1$). Therefore, the total number of paths that exit to the bottom is given by the sum over i which equals [39, §5, p. 137]

$$\sum_{i=0}^{c-1} \binom{r-1+i}{i} = \binom{r-1+c}{c-1},$$

Similarly, the total number of paths that exit to the right are

$$\binom{c-1+r}{r-1} = \binom{r-1+c}{c}.$$

Through Pascal's formula [39, §5, p. 136], the sum of these two gives us

$$P(r,c) = \binom{r+c}{c} = \binom{r+c}{r}.$$

Thus, with $r = 8$ and $c = 10$ the above equation gives 43758 possible paths, consistent with the number obtained (seemingly) empirically by Yan et al. [3]. We can use Stirling's approximation [39, §4, p. 83],

$$x! \approx \sqrt{2\pi x}\left(\frac{x}{e}\right)^x,$$

to obtain

$$P(r,c) \approx \sqrt{\frac{r+c}{2\pi rc}}\frac{(r+c)^{r+c}}{r^r c^c}.$$

Now, to ensure that the probability of reaching the exit points to the right is approximately the same as reaching the exit points at the bottom, r and c should be approximately the same. Thus, we can use the fact that $n = rc$ to approximate r and c by $\sqrt{n}$. Using this in the above and simplifying, gives us

$$P(r,c) \approx \frac{2^{2\sqrt{n}}}{\sqrt{\pi}n^{\frac{1}{4}}}.$$

For $n = 80$, the bound above is $\approx 2^{15.5}$. Thus we can see that the search space for the SAT solver is much less than $\binom{n}{k} = \binom{80}{30} \approx 2^{73}$. To obtain an equivalent number we should have $n \approx 1470$, which is surely impractical.

6 Linearization of the Foxtail Protocol

We now turn our attention to another human identification protocol named Foxtail, first proposed by Li and Shum in [5].[6] The protocol has been analyzed by Yan et al. in [3] where it was shown to be vulnerable to statistical counting based attacks. A fix was proposed in [7] wherein the modified protocol was shown to resist the statistical attacks from Yan et al. [3]. We focus here on this fixed protocol. As in the case of the CG protocol, our attack essentially limits the number of allowable authentication sessions the protocol can be used. Here again, prior to our attack, an upper bound on the allowable sessions was not known. In essence, we show that the challenge-response pairs from the protocol can be represented by a set of quadratic equations. With linearization the secret can then be obtain in $\mathcal{O}(n^2)$ challenge-response pairs using Gaussian elimination. Based on our analysis in the previous sections we argue that other attack techniques are not likely to find the secret in feasible time.

The (modified) Foxtail protocol from [7] is as follows. Let n and k be defined as before. In the setup phase, the user and the server share a random binary vector $\mathbf{x}$ of Hamming weight k as a secret. In an authentication session, the server sends a random vector $\mathbf{c}$ from $\mathbb{Z}_4^n$.[7] The response from the user is the Foxtail function ft defined as

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

[6] This section contains some errors which are corrected in Appendix D. The same corrections apply to Section VI of the version of this paper titled *On the Linearization of Human Identification Protocols: Attacks based on Linear Algebra, Coding Theory and Lattices.*

[7] Note that this is the theoretical description of the protocol, which suffices for our discussion of security. The protocol can be implemented in a graphical way which is much more natural for humans as is shown in [7].

The server repeats the above process a fixed number of times (so that the probability of a successful random guess is low), and accepts the user if all the responses received are correct (according to the above function). Essentially, the Foxtail function maps the dot product between $\mathbf{c}$ and $\mathbf{x}$ modulo 4 to 0 if the result is in $\{0,1\}$, and to 1 if the result is in $\{2,3\}$. For a non-negative integer a, let $\mathsf{msb}(a)$ denote the most significant bit in the binary representation of a. For instance, if $a = 2$, where $2 = 10_2$ (in binary), we have that $\mathsf{msb}(a) = 1$. Our first observation is the following.

Lemma 1. *For $\mathbf{c} \in \mathbb{Z}_4^n$ and $\mathbf{x} \in \mathbb{Z}_2^n$,*

$$\mathsf{ft}(\mathbf{x}, \mathbf{c}) = \mathsf{msb}(\mathbf{c} \cdot \mathbf{x}^T \bmod 4).$$

Proof. If the dot product is in $\{0,1\}$, then the Foxtail function evaluates to 0. The most significant bit of both $0 = 00_2$ and $1 = 01_2$ is 0. On the other hand, if the dot product is in $\{2,3\}$ the Foxtail function evaluates to 1. The most significant bit of both $2 = 10_2$ and $3 = 11_2$ is 1, as required. □

Consider an element c_i of $\mathbf{c}$, where $1 \leq i \leq n$. Since $c_i \in \{0,1,2,3\}$, we can write it as

$$c_i = c_{i,0} + 2c_{i,1},$$

where $c_{i,0}$ and $c_{i,1}$ both belong to $\{0,1\}$. Then, we can write

$$\mathsf{ft}(\mathbf{x}, \mathbf{c}) = \left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right) + \frac{1}{2}\left(\sum_i 2c_{i,1}x_i \bmod 4\right)\right\rfloor. \tag{6}$$

We have the following lemma.

Lemma 2.
$$\frac{1}{2}\left(\sum_i 2c_{i,1}x_i \bmod 4\right) = \sum_i c_{i,1}x_i \bmod 2.$$

Proof. We have two possible cases pertaining to the product $\sum_i c_{i,1}x_i$:

- $\sum_i c_{i,1}x_i = 2m$ for some integer m. The left hand side in this case yields $\frac{1}{2}(2 \cdot 2m \bmod 4) = \frac{1}{2}(0) = 0$. The right hand side yields $2m \bmod 2 = 0$, which is equal to the left hand side.
- $\sum_i c_{i,1}x_i = 2m+1$ for some integer m. The left hand side gives us $\frac{1}{2}(4m + 2 \bmod 4) = \frac{1}{2}(2) = 1$. The right hand side evaluates to $2m + 1 \bmod 2 = 1$, again the same as the left hand side.

□

Using this result in Eq. (6), we get

$$\begin{aligned}\mathsf{ft}(\mathbf{x}, \mathbf{c}) &= \left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right) + \sum_i c_{i,1}x_i \bmod 2\right\rfloor \\ &= \left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right)\right\rfloor + \sum_i c_{i,1}x_i \bmod 2, \end{aligned} \tag{7}$$

where the last step is true since $\sum_i c_{i,1}x_i \bmod 2$ is always an integer. We have the following theorem.

Theorem 1.
$$\left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right)\right\rfloor = \sum_i \sum_{j>i} c_{i,0}c_{j,0}x_ix_j \bmod 2.$$

Proof. We again consider the possible cases related to the product $\sum_i c_{i,0}x_i$:

- $\sum_i c_{i,0}x_i = 4m$ for some non-negative integer m. The left hand side evaluates to $\lfloor\frac{1}{2}(4m \bmod 4)\rfloor = 0$. For the right hand side, first notice that since $c_{i,0}$ and x_i are both binary, the sum of the products being $4m$ means that exactly $4m$ of the $c_{i,0}x_i$'s are 1. This means that exactly $\binom{4m}{2}$ of the $c_{i,0}c_{j,0}x_ix_j$'s are 1. This implies that

$$\binom{4m}{2} = 2m(4m-1) \equiv 0 \bmod 2.$$

  Hence, the right hand side is the same.

- $\sum_i c_{i,0}x_i = 4m+1$ for some non-negative integer m. The left hand side evaluates to $\lfloor\frac{1}{2}(4m + 1 \bmod 4)\rfloor = 0$. For the right hand side we see that now we have exactly $\binom{4m+1}{2}$ of the terms in the summation equal to 1. So, the right hand side also gives us

$$\binom{4m+1}{2} = 2m(4m+1) \equiv 0 \bmod 2.$$

- $\sum_i c_{i,0}x_i = 4m+2$ for some non-negative integer m. The left hand side gives us $\lfloor\frac{1}{2}(4m + 2 \bmod 4)\rfloor = 1$. There are exactly $\binom{4m+2}{2}$ terms in the summation in the right hand side equal to 1. Thus the right hand side evaluates to

$$\binom{4m+2}{2} = 2(4m^2+3m)+1 \equiv 1 \bmod 2.$$

- $\sum_i c_{i,0}x_i = 4m+3$ for some non-negative integer m. The left hand side gives us $\lfloor\frac{1}{2}(4m + 3 \bmod 4)\rfloor = 1$. The summation in the right hand side has exactly $\binom{4m+3}{2}$ terms equal to 1. Thus, we get

$$\binom{4m+3}{2} = 2(4m^2+5m+1)+1 \equiv 1 \bmod 2.$$

  Again, the same as the left hand side.

□

The interpretation of the above result in light of Lemma 1 is as follows. First note that the left hand side above is equivalent to

$$\mathsf{msb}\left(\sum_i c_{i,0}x_i \bmod 4\right).$$

This equals the carry-bit in adding the binary numbers $c_{i,0}x_i$, where we define the carry-bit as the second digit from the right in the binary notation of the sum $\sum_i c_{i,0}x_i$, considering the digits to the left of the carry-bit as overflows (for instance, the carry bit of $5 = 101_2$ is 0). This can be easily seen by noting that the last two digits of the sum can be in one of the four possible configurations: $\{00, 01, 10, 11\}$, and the carry-bits are the first digits in these four configurations. It is easy to see that the carry-bit of n binary numbers is given by the sum modulo 2 of $\binom{n}{2}$ possible products of pairs of these numbers.

Using the result of Theorem 1 in Eq. (7) we get

$$\begin{aligned}\mathsf{ft}(\mathbf{x},\mathbf{c}) &= \left(\sum_i\sum_{j>i} c_{i,0}c_{j,0}x_ix_j + \sum_i c_{i,1}x_i\right) \bmod 2 \\ &= \sum_{i=1}^{\binom{n}{2}+n} c'_iy_i \bmod 2 \\ &= \mathbf{c}'\cdot\mathbf{y}^T \bmod 2 \qquad (8)\end{aligned}$$

where $\mathbf{c}'$ is the $\binom{n}{2}+n$ element binary vector whose first $\binom{n}{2}$ elements are the $c_{i,0}c_{j,0}$'s and the rest of the n elements are the $c_{i,1}$'s, and $\mathbf{y}$ is the $\binom{n}{2}+n$ element binary vector whose first $\binom{n}{2}$ elements are the x_ix_j's and the rest of the n elements are the x_i's. Since the Hamming weight of $\mathbf{x}$ is k, it follows that $\mathbf{y}$ has Hamming weight $\binom{k}{2}+k$. Given a challenge vector $\mathbf{c}$ one can easily construct the vector $\mathbf{c}'$. Let $n' = \binom{n}{2}+n$ and $k' = \binom{k}{2}+k$. Given m challenge-response pairs, we can thus construct a system of congruences

$$C'\mathbf{y} \equiv \mathbf{r} \bmod 2,$$

where C' is the $m \times n'$ matrix built from $\mathbf{c}_i'$'s as defined above, and $\mathbf{r}$ is the vector of m responses. Given $m = n'$ such pairs, we can solve this system uniquely using Gaussian elimination with high probability. For the parameter values $n = 140$, $k = 14$ considered in [7], this gives us $n' = 9870$ and $k' = 105$. For a given confidence level γ, we can then use this system for $\frac{\log_2 2}{\gamma}n' = \frac{n'}{\gamma} = \frac{9870}{\gamma}$ sessions before secret renewal. For $\gamma = 20$, we get about 493 sessions. This number is less than the 711 sessions mark obtained by the statistical attack from Yan et al. [3] on the original Foxtail, after which the protocol was modified in [7].

How about the case when $m < n'$? We see that the lattice-based attacks described in Section 4.3 are not likely to find the secret since n' is way above the 100 mark. Information set decoding might be able to find the solution when $m \geq \log_2(\binom{n'}{k'}) \approx 834$. For smaller m, however, the work required will be infeasible. As m grows we expect a corresponding decrease in the work factor. Figure 5 shows the work factor WF in $\log_2$-scale, which is the work factor obtained in [34]. We saw through the simulation that plain information set decoding performs better after $m > 1956$. To achieve a work factor of less than 2^{50}, $m \geq 7110$ is required. Notice that a work factor of 2^{50} is not a bad choice for infeasibility, as we should also take into consideration the time required in transforming such a large matrix.

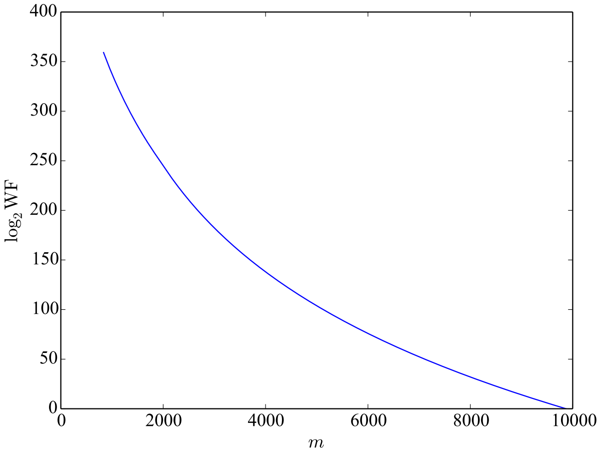


Figure 5: The logarithmic plot of the work factor (WF) of the information set decoding algorithm described in [34] against m on the system of congruences derived from the Foxtail protocol.

7 Discussion

A central design criterion for the two protocols considered in this paper can be thought of as introducing *non-linearity* into the responses to avoid representation as a system of linear congruences. Viewed in this way, they can be put in the category of the protocols from Hopper and Blum [2], namely the HB and the sum of k mins protocol. Briefly, in the HB protocol a wrong response is sent with a fix probability $\eta \in (0, \frac{1}{2})$. The resulting protocol can then be shown to be based on the NP-Hard problem of learning parity in the presence of the noise thus generated. In the sum of k mins protocol, the secret is a set of *pairs* of indices of $\mathbf{x}$. Upon receipt of a challenge, the user chooses the minimum of the two digits in the challenge vector for each pair of secret indices and sends the response as the modulo d sum of the resulting digits. For this protocol, Hopper and Blum recommended $d = 10$. They also showed that the protocol can be represented as a system of linear congruences in $\binom{n}{2}$ unknowns.

Thus, in light of our results, both the Foxtail protocol and the sum of k mins protocol can be *conjectured* to achieve security against $\mathcal{O}(n^2)$ observed challenge-response pairs.[8] This is an improvement over the $\mathcal{O}(n)$ challenge-response pairs obtained from the basic protocol described in Section 3. Another illuminating way to look at the underlying mathematical problem in the two protocols discussed in this paper is as a form of the Learing with Errors (LwE) problem [40]. Consider the Foxtail function first. We can write it as

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

where $\mathbf{r}$ is the response vector from m challenges and $\mathbf{e}$ is an error vector. Rearranging, we get

$$C\mathbf{x} + \mathbf{e}' \equiv \mathbf{r}' \bmod 4,$$

where $\mathbf{e}' = 3\mathbf{e}$ and $\mathbf{r}' = 2\mathbf{r}$. This then can be considered as an LwE problem with the error vector $\mathbf{e}'$ having the distribution χ, such that $\chi(0) = \chi(3) = \frac{1}{2}$ and $\chi(1) = \chi(2) = 0$. Similarly, if we consider the matrix C as composed of m challenges from the CG protocol, we arrive at

$$C\mathbf{x} + \mathbf{e} \equiv \mathbf{r} \bmod 2,$$

where $\mathbf{r}$ is the m-element response vector, each element of which is obtained as a sum of responses from the protocol. The error vector $\mathbf{e}$ has the trivial probability distribution $\chi(\frac{k}{2}) = 1$ and 0 otherwise.

More precisely the two problems can be considered as LwE with *structured noise*, where in the CG protocol we have the additional constraint that the rows of the matrix C have Hamming weight precisely $\frac{n}{2}$. The fact that we show polynomial time attacks on the two protocols after observing $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$ samples in the two protocols, respectively, is consistent with the findings of Arora and Ge [41] and Ding [42] about learning in the presence of structured noise. Both these works leverage structured noise to represent the gathered samples (challenge-response pairs in our terminology) as a system of polynomials of degree $D < d$, where D is the set of noise values with non-zero probability, and use linearization to obtain a solution with $\mathcal{O}(n^D)$ samples. Therefore, by formulating the underlying problem in the two protocols as LwE with structured noise, we may also use the attack from [41] and [42] to obtain a solution with the same sample complexity as our attacks. Note that the attack from Arora and Ge considers a broader definition of structured noise, e.g., at most 3 out of the given m samples are noisy, which also covers the pattern of structured noise considered here. For instance, the noise in the formulation of Foxtail as an LwE instance can be represented by the (quadratic) noise polynomial $P(\eta) = \eta(\eta - 3)$, which only evaluates to 0 when the noise η is either 0 or 3, i.e., the acceptable noise pattern [41].

8 Conclusion

We have presented a detailed analysis of the feasible ways to attack a system of linear congruences. We do not claim that the analysis is comprehensive, as there might be other ways to attack such a system. We have further taken two human identification protocols from literature and shown how they can be represented as a system of linear congruences. Using our analysis on attacks on such systems of congruences, we have shown how the security of these two protocols is reduced in terms of the number of allowable sessions before secret renewal. We have also put the protocols in context with

[8] We emphasize that this is merely a conjecture because there could possibly be other as yet unknown attacks that may reduce this bound further.

other human identification protocols proposed in literature in terms of the number of sessions allowed before secret renewal as a function of the protocol parameter. Moreover, we have shown how the underlying mathematical structure of the two protocols can be thought of as a contrived learning with errors (LwE) problem. An interesting open question is to construct a protocol that can resist $\mathcal{O}(n^3)$ or more observations while keeping the cardinality of the secret to k. This will be an improvement over the (conjectured) $\mathcal{O}(n^2)$ bound reached by the Foxtail protocol discussed in this paper. As the state-of-the-art protocols are arguably not practical for human authentication, an open research question is to find versions of these protocols secure against active attackers for authentication of resource constrained devices; a practice similar to the application of the HB protocol for RFID authentication.

Acknowledgments

The work of Ron Steinfeld was partially supported by ARC Discovery Grant DP150100285. We thank Matthieu Monteiro for pointing out errors in Section 6 which are corrected in Appendix D.

References

[1] T. Matsumoto and H. Imai. Human identification through insecure channel. In *EUROCRYPT '91*, pages 409–421. Springer-Verlag, 1991.

[2] N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *ASIACRYPT '01*, pages 52–66. Springer-Verlag, 2001.

[3] Qiang Yan, Jin Han, Yingjiu Li, and Robert H. Deng. On Limitations of Designing Leakage-Resilient Password Systems: Attacks, Principals and Usability. In *19th Annual Network and Distributed System Security Symposium*, NDSS '12. The Internet Society, 2012.

[4] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO '05, pages 293–308, Berlin, Heidelberg, 2005. Springer-Verlag.

[5] S. Li and H.-Y. Shum. Secure Human-Computer Identification (Interface) Systems against Peeping Attacks: SecHCI. IACR's Cryptology ePrint Archive: Report 2005/268, http://eprint.iacr.org/2005/268, 2005.

[6] Luigi Catuogno and Clemente Galdi. A Graphical PIN Authentication Mechanism with Applications to Smart Cards and Low-Cost Devices. In *WISTP*, volume 5019 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2008.

[7] Hassan Jameel Asghar, Shujun Li, Ron Steinfeld, and Josef Pieprzyk. Does Counting Still Count? Revisiting the Security of Counting based User Authentication Protocols against Statistical Attacks. In *20th Annual Network and Distributed System Security Symposium*, NDSS '13. The Internet Society, 2013.

[8] Chih-Hung Wang, Tzonelih Hwang, and Jiun-Jang Tsai. On the Matsumoto and Imai's Human Identification Scheme. In *Proceedings of the 14th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '95, pages 382–392, Berlin, Heidelberg, 1995. Springer-Verlag.

[9] Daphna Weinshall. Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In *IEEE Symposium on Security and Privacy*, SP '06, pages 295–300. IEEE Computer Society, 2006.

[10] 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):4367–4375, 2008.

[11] L. Sobrado and J.-C. Birget. Graphical Passwords. *The Rutgers Scholar*, 4, 2002.

[12] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme. In *AVI '06*, pages 177–184. ACM, 2006.

[13] X. Bai and W. Gu and S. Chellappan and X. Wang and D. Xuan and B. Ma. PAS: Predicate-Based Authentication Services against Powerful Passive Adversaries. In *ACSAC '08*, pages 433–442. IEEE Computer Society, 2008.

[14] Philippe Golle and David Wagner. Cryptanalysis of a Cognitive Authentication Scheme (Extended Abstract). In *IEEE Symposium on Security and Privacy*, SP '07, pages 66–70. IEEE Computer Society, 2007.

[15] S. Li, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz. Breaking Randomized Linear Generation Functions based Virtual Password System. In *ICC '10*. IEEE, 2010.

[16] H. J. Asghar, S. Li, J. Pieprzyk, and H. Wang. Cryptanalysis of the Convex Hull Click Human Identification Protocol. In *ISC '10*, pages 24–30. Springer-Verlag, 2010.

[17] S. Li, H. J. Asghar, J. Pieprzyk, A.-R. Sadeghi, R. Schmitz, and H. Wang. On the Security of PAS (Predicate-based Authentication Service). In *ACSAC '09*, pages 209–218. IEEE Computer Society, 2009.

[18] T. Matsumoto. Human-Computer Cryptography: An Attempt. In *CCS '96*, pages 68–75. ACM, 1996.

[19] Hassan Jameel Asghar, Josef Pieprzyk, and Huaxiong Wang. A New Human Identification Protocol and Coppersmith's Baby-step Giant-step Algorithm. In *Proceedings of the 8th International Conference on Applied Cryptography and Network Security*, ACNS '10, pages 349–366, Berlin, Heidelberg, 2010. Springer-Verlag.

[20] Mario Čagalj and Toni Perković. Timing Attacks on Cognitive Authentication Schemes. *IEEE Transactions on Information Forensics and Security*, PP(99):1, 2014.

[21] E. Berlekamp, R. McEliece, and H. Van Tilborg. On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3):384–386, 1978.

[22] Alexander Barg. Complexity Issues in Coding Theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46), 1997.

[23] Rod G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The Parametrized Complexity of Some Fundamental Problems in Coding Theory. *SIAM J. Comput.*, 29(2):545–570, 1999.

[24] Hassan Jameel Asghar, Josef Pieprzyk, and Huaxiong Wang. On the Hardness of the Sum of k Mins Problem. *Comput. J.*, 54(10):1652–1660, 2011.

[25] Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. On learning k-parities with and without noise. arXiv.org e-Print archive, arXiv:1502.05375 [cs.DS], http://arxiv.org/abs/1502.05375, 2015.

[26] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with Small Parameters. In *CRYPTO '13*, pages 21–39. Springer, 2013.

[27] Yanbin Pan and Feng Zhang. A Note on the Density of the Multiple Subset Sum Problems. Cryptology ePrint Archive, Report 2011/525, 2011. http://eprint.iacr.org/.

[28] Jeffrey C. Lagarias and Andrew M. Odlyzko. Solving Low-density Subset Sum Problems. *Journal of the ACM*, 32(1):229–246, 1985.

[29] H.W. Lenstra, A.K. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[30] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved Low-Density Subset Sum Algorithms. *Computational Complexity*, 2:111–128, 1992.

[31] C. P. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. *Math. Program.*, 66(2):181–199, 1994.

[32] Christiane Peters. Information-set Decoding for Linear Codes over $\mathbf{F}_q$. In *Proceedings of the Third International Conference on Post-Quantum Cryptography*, PQCrypto '10, pages 81–94, Berlin, Heidelberg, 2010. Springer-Verlag.

[33] Eugene Prange. The Use of Information Sets in Decoding Cyclic Codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[34] Robert Niebuhr, Pierre-Louis Cayrel, Stanislav Bulygin, and Johannes Buchmann. On lower bounds for Information Set Decoding over $\mathbb{F}_q$. In *Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography*, SCC '10, pages 143–157, 2010.

[35] Alexander Meurer. A Coding-Theoretic Approach to Cryptanalysis. PhD Dissertation. Available online at http://www.cits.rub.de/imperia/md/content/diss.pdf, 2012.

[36] Jacques Stern. A Method for Finding Codewords of Small Weight. In *Proceedings of the 3rd International Colloquium on Coding Theory and Applications*, pages 106–113, London, UK, 1989. Springer-Verlag.

[37] Luigi Catuogno and Clemente Galdi. On the Security of a Two-factor Authentication Scheme. In *Proceedings of the 4th IFIP WG 11.2 International Conference on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices*, WISTP '10, pages 245–252, Berlin, Heidelberg, 2010. Springer-Verlag.

[38] Luigi Catuogno and Clemente Galdi. Analysis of a two-factor graphical password scheme. *International Journal of Information Security*, pages 1–17, 2014.

[39] Richard A. Brualdi. *Introductory Combinatorics.* Pearson Education, Inc., New Jersey, USA, 4th edition, 2004.

[40] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[41] Sanjeev Arora and Rong Ge. Learing Parities with Structured Noise. Hasso-Plattner-Institut: TR10-066, http://eccc.hpi-web.de/report/2010/066/, 2010.

[42] Jintai Ding. Algebraic Solvers for Certain Lattice-related Problems. In *Proceedings of the IEEE Information Theory Workshop*, ITW '11, pages 405–409. IEEE, 2011.

[43] Jean Gallier. Fundamentals of Linear Algebra and Optimization – CIS515, Some Notes. Available online at http://www.seas.upenn.edu/~cis515/linalg.pdf, 2012.

[44] Neal H. McCoy. Rings and Ideals. *Carus Mathematical Monographs*, 8, 1948.

[45] Wai-Sin Ching. Linear Equations over Commutative Rings. *Linear Algebra and its Applications*, 18(3):257–266, 1977.

[46] Marcus Nilsson and Robert Nyqvist. Number of Solutions of Linear Congruence Systems. arXiv.org e-Print archive, arXiv:1208.3550 [math.NT], http://arxiv.org/abs/1208.3550, 2012.

[47] Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, New York, USA, 2nd edition, 2009.

[48] D. Bollman and H. Ramirez. On the Enumeration of Matrices Over Finite Commutative Rings. *The American Mathematical Monthly*, 76(9):1019–1023, 1969.

[49] Guy Robin. Estimation de la fonction de Tchebychef θ sur le k-ième nombre premier et grandes valeurs de la fonction $\omega(n)$ nombre de diviseurs premiers de n. *Acta Arithmetica*, 42(4):367–389, 1983.

A Random Square Matrices over $\mathbb{Z}_d$

Let C be a random $n \times n$ matrix with entries from $\mathbb{Z}_d$, where $d \geq 2$. Let $\mathbf{x}$ be a random binary vector of n-elements.[9] Let $\mathbf{r}$ be an n-element vector defined as

$$C\mathbf{x} \equiv \mathbf{r} \bmod d, \tag{9}$$

then we are interested in finding when Gaussian elimination can be used to find the unique solution (i.e., a unique $\mathbf{x}$) to the above system of linear congruences. The set $\mathbb{Z}_d$, together with the usual operations of addition and multiplication, is a field when d is a prime, and a commutative ring with unity, otherwise. The two cases are therefore considered separately.

A.1 When d is Prime

When d is a prime then Eq. (9) has a unique solution if only if $\mathsf{det}(C) \neq 0$ [43, §5.5, p. 189]. Now $\mathsf{det}(C) \neq 0$ if and only if the n columns of C are linearly independent [43, §5.4, p. 188]. Equivalently, $\mathsf{det}(C) \neq 0$ if and only if C has an inverse. Thus, if the matrix C is invertible the above system of linear equations has a unique solution. This solution can then be found through Gaussian elimination. We therefore find the probability that C is invertible.

Let $\eta(n, d)$ denote the number of $n \times n$ invertible matrices over $\mathbb{Z}_d$. Recall that an invertible C is the same as having n linearly independent rows. For the first row vector we have $d^n - 1$ choices, since the only constraint on this vector is that it should not be the zero vector. The second vector should not be generated as a linear combination of the first vector, which gives us $d^n - d$ choices for this vector. Generalizing, the ith vector, where $1 \leq i \leq n$, should not be the linear combination of the $i - 1$ previous vectors, implying that we have $d^n - d^{i-1}$ choices for this vector. Thus,

$$\eta(n, d) = \prod_{i=1}^{n} (d^n - d^{i-1}).$$

Now, to calculate the invertibility probability, p_{inv}, we see that there are a total of d^{n^2} $n \times n$ matrices over $\mathbb{Z}_d$. Therefore,

$$p_{\mathsf{inv}} = \frac{\prod_{i=1}^{n} (d^n - d^{i-1})}{d^{n^2}} = \prod_{i=1}^{n} (1 - d^{i-1-n}) = \prod_{i=1}^{n} (1 - d^{-i}).$$

[9] For this section, we do not impose any restriction on the Hamming weight of $\mathbf{x}$.

We would like to emphasize that the above result is not new, but we nevertheless present the derivation here for completeness. Figure 6 shows p_{inv} against d as the latter ranges from 2 to 25. For now, consider only the prime values of d. We see that p_{inv} is non-negligible. However, since it is less than 1 there is a good chance that C is non-invertible. We are therefore also interested in finding the expected number of calls required to an oracle that returns a random n-element vector from $\mathbb{Z}_d$, before we have a set of n linearly independent vectors. From the perspective of human identification protocols, this gives us the expected number of challenge-response pairs required to obtain a unique solution through Gaussian elimination. As p_{inv} is quite high, intuitively this number should be very close to n. We are interested in finding a bound for the expected number of calls that is linear in n.

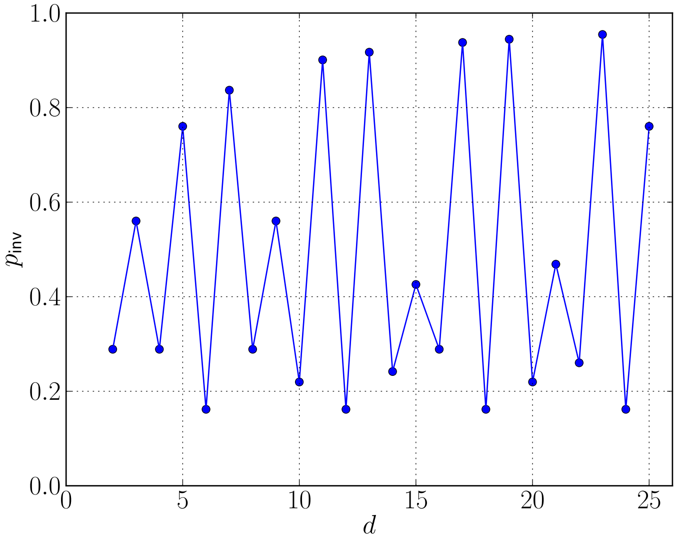


Figure 6: The probability that an $n \times n$ matrix over $\mathbb{Z}_d$ is invertible.

We use the following simple strategy to obtain n linearly independent vectors: We retain the first non-zero vector. Subsequently, if the ith vector is linearly dependent on the set of $i-1$ vectors thus obtained, we discard it. Otherwise we include it in the set. Notice that this strategy is optimal in terms of the number of oracle calls provided the probability of obtaining the ith linearly independent vector does not depend on a specific set of $i-1$ linearly independent vectors. It is easy to see the strategy is optimal for obtaining the first vector. For the second vector, we see that if it is linearly dependent on the first vector, then replacing it with the first vector has no influence on the probability of the second vector being linearly independent. For the third vector, if it is dependent on one of the two vectors, replacing it with one of the vectors still gives us only two linearly dependent vectors and we are at the same place. Replacing it with both the vectors will give us one vector less and the probability to obtain 3 linearly independent vectors will be less than the case in which we discard this third vector. Continuing on this way, we see that this strategy is indeed optimal.

Let X_i denote the number of oracle calls required to obtain the ith linearly independent vector given that we have $i-1$ linearly independent vectors. Then X_i is a geometric random variable whose expected value is

$$\mathrm{E}[X_i] = \frac{d^n}{d^n - d^{i-1}},$$

which is the reciprocal of the probability that the ith vector is linearly independent. Let X denote the total number of oracle calls required to obtain n linearly independent vectors. Then

$$X = X_1 + X_2 + \cdots + X_n.$$

And therefore, the expected value is

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[X_1] + \mathrm{E}[X_2] + \cdots + \mathrm{E}[X_n] \\
&= \frac{d^n}{d^n - d^0} + \frac{d^n}{d^n - d^1} + \cdots + \frac{d^n}{d^n - d^{n-1}} \\
&= \frac{d^n}{d^n}\left(\frac{1}{1-\frac{1}{d^n}} + \frac{1}{1-\frac{d}{d^n}} + \cdots + \frac{1}{1-\frac{d^{n-1}}{d^n}}\right) \\
&= \frac{1}{1-\frac{1}{d}} + \frac{1}{1-\frac{1}{d^2}} + \cdots + \frac{1}{1-\frac{1}{d^n}} \\
&= \frac{d}{d-1} + \frac{d^2}{d^2-1} + \cdots + \frac{d^n}{d^n-1} \\
&= \frac{d-1+1}{d-1} + \frac{d^2-1+1}{d^2-1} + \cdots + \frac{d^n-1+1}{d^n-1} \\
&= \left(1+\frac{1}{d-1}\right) + \left(1+\frac{1}{d^2-1}\right) + \cdots + \left(1+\frac{1}{d^n-1}\right) \\
&= n + \sum_{i=1}^{n} \frac{1}{d^i - 1} \qquad (10)
\end{aligned}$$

Now, we show by induction that

$$d^i - 1 \geq d^{i-1} \qquad (11)$$

for all $i \geq 1$. First, let $i = 1$. This gives us $d^1 - 1 \geq d^0 \Rightarrow d - 1 \geq 1 \Rightarrow d \geq 2$, which is true by the definition of d. Now assume that for all positive integers less than r, $d^r - 1 \geq d^{r-1}$. Then

$$\begin{aligned}
d^r &\geq d^{r-1} + 1 \\
\Rightarrow d^{r+1} &\geq (d^{r-1} + 1)d \\
\Rightarrow d^{r+1} &\geq d^{r-1}d + d \\
\Rightarrow d^{r+1} &\geq d^r + d > d^r + 1 \\
\Rightarrow d^{r+1} - 1 &> d^r,
\end{aligned}$$

which proves the Inequality (11). Using this result in Eq. (10), gives us

$$\mathrm{E}[X] \leq n + \sum_{i=1}^{n} \frac{1}{d^{i-1}}.$$

By changing the variable i to $j = i - 1$, we see that the sum above is translated into the sum of the first n terms of a geometric series with the first term 1 and common ratio $\frac{1}{d}$. Thus, the sum above is

$$\frac{1-\left(\frac{1}{d}\right)^n}{1-\frac{1}{d}} = \frac{d^n - 1}{d^n}\frac{d}{d-1}.$$

Therefore,

$$\mathrm{E}[X] \leq n + \frac{d^n-1}{d^n}\frac{d}{d-1} < n + \frac{d}{d-1}, \qquad (12)$$

where we have used the fact that $\frac{d^n-1}{d^n} < 1$ for all $d > 1$ and $n \geq 0$. Figure 7 shows the bounded versus actual expected values $\mathrm{E}[X]$ against d. The next section explains how the values are obtained for composite d.

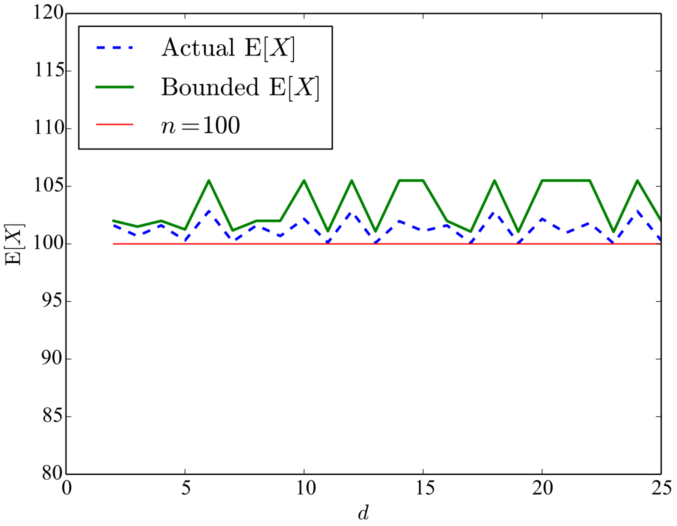


Figure 7: The actual versus bounded expected number of calls E[X] to an oracle required to obtain n linearly independent vectors in $\mathbb{Z}_d^n$.

A.2 When d is Composite

Now, when $\mathbb{Z}_d$ is not a field, i.e., d is composite, the matrix C is invertible if and only if $\mathsf{det}(C)$ is invertible, which is equivalent to saying that $\mathsf{det}(C)$ is a unit of the commutative ring $\mathbb{Z}_d$ [44, 45]. Recall that a unit is an element of the ring which has an inverse. For instance, in $\mathbb{Z}_4$, the only units are 1 and 3. In the case of matrices over commutative rings too, the existence of an invertible matrix C is equivalent to saying that a unique solution exists [44,45]. Thus, the above system of congruences has a unique solution if and only if C is invertible in $\mathbb{Z}_d$. If that is the case, one can apply Gaussian elimination to find the vector $\mathbf{x}$. Notice that Gaussian elimination can be modified slightly to be used in the ring $\mathbb{Z}_d$, where d is composite. Namely, by avoiding multiplication of rows by zero divisors. See for instance [46] and [47, §14.5, p. 397]. Recall that a zero divisor a is a non-zero element of the commutative ring such that $ab = 0$ for some non-zero element b of the ring. For instance, 2 is a zero divisor of the ring $\mathbb{Z}_4$.

In case $\mathbb{Z}_d$ is a commutative ring with 1, the number of invertible matrices are [48]

$$\eta(n, d) = d^{n^2} \prod_{i=1}^{k} \prod_{j=1}^{n} (1 - p_i^{-j}).$$

The invertibility probability p_{inv} is then

$$p_{\mathsf{inv}} = \frac{d^{n^2} \prod_{i=1}^{k} \prod_{j=1}^{n} (1 - p_i^{-j})}{d^{n^2}} = \prod_{i=1}^{k} \prod_{j=1}^{n} (1 - p_i^{-j}).$$

Notice that when d is a prime, the only prime factor of d is d itself. Therefore, the above formula agrees with the previous case. Looking at Figure 6, we see that while p_{inv} is less than the case when d is a prime, it is still non-negligible. We again want to find a bound on the expected number of oracle calls required to obtain n linearly independent vectors. To obtain the bound we use a lemma from [48] about the rank of an $n \times n$ matrix over $\mathbb{Z}_d$. Let C be an $m \times n$ matrix where $m \leq n$. Let $\mathsf{rank}(C)$ denote the rank of the matrix C which is defined as follows: The rank of C is 0 if and only if there exists an $a \in \mathbb{Z}_d$ such that $ac_{i,j} \equiv 0 \bmod d$ for all $c_{i,j} \in C$. If this is not the case then the rank of C is the greatest integer $r \geq 1$ such that $a \cdot \mathsf{det}(C_{i,j}) \equiv 0 \bmod d$ implies that $a = 0$ for each $r \times r$ minor $C_{i,j}$ of C [48]. This is sometimes referred to as the McCoy rank [45]. Notice that if d is a prime, this corresponds to the usual definition of rank. We shall use the following theorem which can be traced back to McCoy [44, 45]:

Theorem 2. *Let C be an $n \times n$ matrix over $\mathbb{Z}_d$. Then $\mathsf{rank}(C) = n$ if and only if $\mathsf{det}(C)$ is a unit in $\mathbb{Z}_d$ if and only if the n rows (or columns) of C are linearly independent.*

Note that this theorem establishes the link between linear independence and full rank of matrices over $\mathbb{Z}_d$ when d is composite. Assume C is an $m \times n$ matrix, then $\pi_a(C)$ is defined as the $m \times n$ matrix obtained by replacing each element of C by its residue modulo a, where $a \leq d$. We say that $\pi_a(C)$ is the projection of C modulo a. The following is a rewording of the lemma from [48] for the specific ring $\mathbb{Z}_d$:

Lemma 3. *Let $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ be the prime factorization of d. Let C be an $n \times n$ matrix over $\mathbb{Z}_d$, then*

$$\mathsf{rank}(C) = \min\{\mathsf{rank}(\pi_{p_1^{\alpha_1}}(C)), \mathsf{rank}(\pi_{p_2^{\alpha_2}}(C)), \ldots, \mathsf{rank}(\pi_{p_k^{\alpha_k}}(C))\}.$$

Finally, we shall use the following lemma:

Lemma 4. *Let p be a prime and let $\alpha \geq 1$ be a positive integer. Let C be an $n \times n$ matrix over $\mathbb{Z}_{p^\alpha}$, then $\mathsf{rank}(C) = n$ if and only if $\mathsf{rank}(\pi_p(C)) = n$.*

Proof. First note that if $\alpha = 1$ then the relation trivially holds. Therefore, we may assume $\alpha \geq 2$. First suppose $\mathsf{rank}(C) = r$ and $\mathsf{rank}(\pi_p(C)) = n$ where $r < n$. Since the rank of $\pi_p(C)$ is n, its n rows $\mathbf{c}_1, \ldots, \mathbf{c}_n$ are linearly independent vectors. Their counterparts in C can be written uniquely as

$$\mathbf{c}'_i = \mathbf{q}_i p + \mathbf{c}_i,$$

where $\mathbf{q}_i$ is the quotient vector. Since the rank of C is less than n, by Theorem 2 these n vectors must be linearly dependent modulo p^α. Therefore,

$$\begin{aligned} \sum_{i=1}^{n} a_i \mathbf{c}'_i &\equiv \mathbf{0} \bmod p^\alpha \\ \Rightarrow \sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i=1}^{n} a_i \mathbf{c}_i &\equiv \mathbf{0} \bmod p^\alpha, \end{aligned}$$

for some $a_i \in \mathbb{Z}_{p^\alpha}$ not all congruent to zero modulo p^α. For some quotient vector $\mathbf{q}$, the above congruence can be written as

$$\begin{aligned} \Rightarrow \sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i=1}^{n} a_i \mathbf{c}_i &= \mathbf{q} p^\alpha \qquad (13) \\ \Rightarrow \sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i=1}^{n} a_i \mathbf{c}_i &= \mathbf{q} p^{\alpha-1} p \\ \Rightarrow \sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i=1}^{n} a_i \mathbf{c}_i &\equiv \mathbf{0} \bmod p \\ \Rightarrow \sum_{i=1}^{n} a_i \mathbf{c}_i &\equiv \mathbf{0} \bmod p. \end{aligned}$$

As these vectors are linearly independent in $\pi_p(C)$, all non-zero a_i's should be multiples of p. Thus $a_i = b_i p^{\beta_i}$ for all non-zero a_i's, where $b_i \in \{1, \dots, p-1\}$ and $1 \leq \beta_i < \alpha$. Let β_j be the minimum of such β_i's, where $j \in \{1, \dots, n\}$. Now, Eq. (13) can be written as

$$\sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i \neq j} a_i \mathbf{c}_i + b_j p^{\beta_j} \mathbf{c}_j = \mathbf{q} p^{\alpha}$$
$$\Rightarrow \sum_{i=1}^{n} a_i \mathbf{q}_i p + \sum_{i \neq j} a_i \mathbf{c}_i + b_j p^{\beta_j} \mathbf{c}_j = \mathbf{q} p^{\alpha - \beta_j - 1} p p^{\beta_j}.$$

Now, clearly p^{β_j} divides the right hand side. Since β_j is the minimum of the β_i's it divides the left hand side as well (trivially dividing the zero a_i's). Therefore, we can cancel p_{β_j} on both sides, and write the above as a congruence modulo p. Note that the left most term being a multiple of p vanishes as a result. In the resulting congruence, b_j is non-zero by definition. Therefore, we reach a contradiction that the set of n vectors are linearly dependent in $\pi_p(C)$.

Now assume $\mathsf{rank}(C) = n$ and $\mathsf{rank}(\pi_p(C)) = r$, where $r < n$. Let $\mathbf{c}_1, \dots, \mathbf{c}_n$ be the set of n linearly independent rows of C guaranteed by Theorem 2. Define:

$$\mathbf{c}_i = \mathbf{q}_i p + \mathbf{c}'_i,$$

where $\mathbf{q}_i$ is the quotient vector and $\mathbf{c}'_i \in \mathbb{Z}_p^n$, for all $i \in \{1, \dots, n\}$. Since the rank of $\pi_p(C)$ is less than n, the n vectors $\mathbf{c}'_i$ should be linearly dependent modulo p. In other words

$$\sum_{i=1}^{n} a_i \mathbf{c}'_i \equiv \mathbf{0} \bmod p,$$

for some $a_i \in \mathbb{Z}_p$ not all congruent to zero modulo p. The above congruence can be written as the following equality for some quotient vector $\mathbf{q}$,

$$\sum_{i=1}^{n} a_i (\mathbf{c}_i - \mathbf{q}_i p) = \mathbf{q} p$$
$$\Rightarrow \sum_{j=1}^{n} a_i \mathbf{c}_i - \sum_{i=1}^{n} a_i p \mathbf{q}_i = \mathbf{q} p.$$

Multiplying the above by $p^{\alpha-1}$, we obtain

$$\sum_{j=1}^{n} a_i p^{\alpha-1} \mathbf{c}_i - \sum_{i=1}^{n} a_i p^{\alpha} \mathbf{q}_i = \mathbf{q} p^{\alpha}$$
$$\Rightarrow \sum_{j=1}^{n} a_i p^{\alpha-1} \mathbf{c}_i \equiv \mathbf{0} \bmod p^{\alpha}.$$

Since the $\mathbf{c}_i$'s are linearly independent modulo p^{α}, it must be that $a_i p^{\alpha-1}$ is zero modulo p^{α} for all i. On the other hand a_i is non-zero modulo p for at least one $i \in \{1, \dots, n\}$. Let j be that i. Then for some non-zero b

$$a_j p^{\alpha-1} = b p^{\alpha} \Rightarrow a_j p^{\alpha-1} = b p^{\alpha-1} p \Rightarrow a_j = bp$$

which implies that a_j is zero modulo p. Again, a contradiction. The result follows. □

Given an oracle that returns a random vector from $\mathbb{Z}_d^n$, where d is composite and has the factorization $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, we have the following optimal strategy (in terms of number of oracle calls) to obtain a matrix C of rank n. We retain the first non-zero vector. For the ith vector, for each p_j in the factorization of d, we see if the projection of the matrix $i \times n$ modulo p_j is of full rank. If for all p_j, the rank is i, we add the vector in the collection. Otherwise, we discard it. By the time we have n such vectors, we obtain a square matrix C over $\mathbb{Z}_d$ of rank n which is equivalent to saying that the determinant of this matrix is a unit in $\mathbb{Z}_d$ (Theorem 2). This in turn means that Gaussian elimination will yield a unique solution of Equation 9. By the same argument as in the case of a prime d, we see that such a strategy is indeed optimal.

Now, the probability that the projection of the ith vector is linearly independent modulo each p_j is given by

$$\frac{\prod_{j=1}^{k} p_j^{(\alpha_j - 1)n}(p_j^n - p_j^{i-1})}{\prod_{j=1}^{k} p_j^{\alpha_j n}}.$$

This is true since each vector in $\mathbb{Z}_{p_j}^n$ corresponds to $p_j^{(\alpha-1)n}$ different vectors in $\mathbb{Z}_d^n$. Furthermore, a random vector from $\mathbb{Z}_d^n$ is also random modulo p_j. Therefore, the individual probabilities are independent for all p_j. Finally the total number of vectors is d^n, giving us the denominator in the above probability. Let X_i be the number of oracle calls required to obtain the ith vector such that its projection is linearly independent modulo each p_j, given that we have $i-1$ such vectors. As before, X_i is a geometric random variable whose expected value is the reciprocal of the above defined probability. Therefore

$$\begin{aligned}
\mathrm{E}[X_i] &= \frac{\prod_{j=1}^{k} p_j^{\alpha_j n}}{\prod_{j=1}^{k} p_j^{(\alpha_j - 1)n}(p_j^n - p_j^{i-1})} \\
&= \frac{\prod_{j=1}^{k} p_j^{\alpha_j n}}{\prod_{j=1}^{k} p_j^{\alpha_j n} p_j^{-n}(p_j^n - p_j^{i-1})} \\
&= \frac{\prod_{j=1}^{k} p_j^{\alpha_j n}}{\prod_{j=1}^{k} p_j^{\alpha_j n} \prod_{j=1}^{k}(1 - p_j^{-n+i-1})} \\
&= \frac{1}{\prod_{j=1}^{k}(1 - \frac{1}{p_j^{n-i+1}})}
\end{aligned}$$

Now let X denote the total number of oracle calls required to obtain n linearly independent vectors. Then

$$X = X_1 + X_2 + \cdots + X_n.$$

And the expected value is

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \sum_{i=1}^{n} \frac{1}{\prod_{j=1}^{k}(1 - \frac{1}{p_j^{n-i+1}})} \\
&= \sum_{i=1}^{n} \frac{1}{\prod_{j=1}^{k}(1 - \frac{1}{p_j^{i}})} \\
&= \sum_{i=1}^{n} \prod_{j=1}^{k} \left(\frac{p_j^i}{p_j^i - 1} \right) \\
&= \sum_{i=1}^{n} \prod_{j=1}^{k} \left(\frac{p_j^i - 1 + 1}{p_j^i - 1} \right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} \prod_{j=1}^{k} \left(1 + \frac{1}{p_j^i - 1}\right) \\
&\leq \sum_{i=1}^{n} \prod_{j=1}^{k} \left(1 + \frac{1}{p_j^{i-1}}\right)
\end{aligned}
$$

where in the last step we have used Inequality (11). Now, $p_j \geq 2$. This implies that

$$1 + \frac{1}{p_j^{i-1}} \leq 1 + \frac{1}{2^{i-1}}$$

Putting this in the above, we get

$$
\begin{aligned}
\mathrm{E}[X] &\leq \sum_{i=1}^{n} \prod_{j=1}^{k} \left(1 + \frac{1}{2^{i-1}}\right) \\
&= \sum_{i=1}^{n} \left(1 + \frac{1}{2^{i-1}}\right)^k \\
&= \sum_{i=1}^{n} \sum_{j=0}^{k} \binom{k}{j} \left(\frac{1}{2^{i-1}}\right)^j \\
&= \sum_{j=0}^{k} \binom{k}{j} \sum_{i=1}^{n} \left(\frac{1}{2^{i-1}}\right)^j \\
&= \sum_{j=0}^{k} \binom{k}{j} \sum_{i=1}^{n} \left(\frac{1}{2^j}\right)^{i-1} \\
&= \binom{k}{0} \sum_{i=1}^{n} \left(\frac{1}{2^0}\right)^{i-1} + \sum_{j=1}^{k} \binom{k}{j} \sum_{i=1}^{n} \left(\frac{1}{2^j}\right)^{i-1} \\
&= n + \sum_{j=1}^{k} \binom{k}{j} \sum_{i=1}^{n} \left(\frac{1}{2^j}\right)^{i-1}
\end{aligned}
$$

With the change of variable $l = i - 1$, we see that the inner sum is the sum of first n terms of the geometric series whose first term is 1 and common ratio is $\frac{1}{2^j}$. This sum is

$$\frac{1 - \frac{1}{2^{jn}}}{1 - \frac{1}{2^j}} = \frac{2^{jn} - 1}{2^{jn}} \frac{2^j}{2^j - 1} < \frac{2^j}{2^j - 1} = \frac{2^j - 1 + 1}{2^j - 1} = 1 + \frac{1}{2^j - 1},$$

where we have used the observation that $\frac{2^{jn}-1}{2^{jn}} < 1$ for all $n \geq 0$. Thus,

$$
\begin{aligned}
\mathrm{E}[X] &< n + \sum_{j=1}^{k} \binom{k}{j} \left(1 + \frac{1}{2^j - 1}\right) \\
&= n + \sum_{j=1}^{k} \binom{k}{j} + \sum_{j=1}^{k} \binom{k}{j} \frac{1}{2^j - 1} \\
&= n + 2^k - 1 + \sum_{j=1}^{k} \binom{k}{j} \frac{1}{2^j - 1},
\end{aligned}
$$

where above we have used the identity

$$\sum_{j=0}^{k}\binom{k}{j} = 2^k \Rightarrow \sum_{j=1}^{k}\binom{k}{j} = 2^k - 1.$$

Now, again invoking Inequality (11) we see that

$$\begin{aligned} \mathrm{E}[X] &< n + 2^k - 1 + \sum_{j=1}^{k}\binom{k}{j}\frac{1}{2^{j-1}} \\ &= n + 2^k - 1 + 2\sum_{j=1}^{k}\binom{k}{j}\frac{1}{2^{j}} \\ &= n + 2^k - 1 + 2\left(\left(1+\frac{1}{2}\right)^k - 1\right) \\ &= n + 2^k + 2\left(\frac{3}{2}\right)^k - 3, \end{aligned} \tag{14}$$

where we have used the identity

$$\sum_{j=0}^{k}\binom{k}{j}\frac{1}{2^j} = \left(1+\frac{1}{2}\right)^k \Rightarrow \sum_{j=1}^{k}\binom{k}{j}\frac{1}{2^j} = \left(1+\frac{1}{2}\right)^k - 1.$$

For completeness, one may use the following bound from Robin [49, p. 380] for the number of unique prime factors k of d to obtain a bound in n and d only:

$$k \le 1.38402\frac{\ln d}{\ln\ln d}.$$

However, this bound is rather loose for small values of d. Figure 7 shows the actual and bounded expected values $\mathrm{E}[X]$. The bounded values of $\mathrm{E}[X]$ for composite d are obtained by first finding the number of unique prime factors of d and then using the bound from Inequality (14).

B Binary Matrices with Fixed Row Weight

In this section we assume that a vector means a row vector unless otherwise implied. Let $n \ge 2$ and $0 < r < n$. In the following, all computations are done in the vector space $\mathbb{Z}_2^n$. We are interested in the set

$$X = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}_2^n \text{ and } \mathsf{wt}(\mathbf{x}) = r\}.$$

First notice that X is not a subspace of the vector space $\mathbb{Z}_2^n$. To see this, consider $n = 4$ and $r = 2$ and the two vectors

$$\begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix} \text{ and } \begin{pmatrix}0 & 1 & 1 & 0\end{pmatrix}.$$

Then their sum is the vector $\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}$ which is not contained in X. We therefore cannot apply any results about subspaces directly to X. For a subset S of X, we define the span of S, denoted $\mathsf{span}(S)$ as the set of all linear combinations of elements of S. We say that the set S spans X if $\mathsf{span}(S) \supseteq X$. If this is true, we call S a spanning set. Linear dependency of any set of elements in S follows the usual definition.

We now show that if r is even, then any set of $n-1$ linearly independent vectors from X spans X. On the other hand, when r is odd, any set of n linearly independent vectors from X spans X. We consider even and odd r separately. First, suppose r is even. We take $n = 8$ and $r = 4$ as an example. Consider the row vectors of the matrix below

$$S_{\mathsf{even}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{15}$$

This is a 7×8 matrix. We argue that any element of X can be represented by a linear combination of the rows of S_{even}, and therefore the rows of S_{even} span X. Consider a putative vector $\mathbf{x}$ from X that cannot be represented as a linear combination of the rows of S. In the following it is convenient to imagine a partition between the first 3 and last 5 columns of S_{even}. We shall refer to the two partitions as C_3 and C_5 respectively. For a vector $\mathbf{x}$ we say it has i ones in C_3 and $r - i$ ones in C_5 to mean that $\mathbf{x}$ has i ones in the first 3 columns (or positions) and $r - i$ ones in the last 5 columns (or positions). Note that any vector $\mathbf{x}$ in our example has $r = 4$ ones. There are four possible cases.

1. The vector $\mathbf{x}$ has 3 ones in C_3 and the last one in C_5. In this case we can represent $\mathbf{x}$ as one of the first 5 rows.

2. The vector $\mathbf{x}$ has 2 ones in C_3 and the remaining two in C_5. First assume that the 2 ones in C_3 are arranged in either of the two configurations corresponding to the last two rows of S_{even}. Clearly, the two 2 ones of $\mathbf{x}$ in C_5 cannot be in positions 4 and 5 (otherwise it is one of the last two vectors). For other cases, we pick the two rows from S whose one in C_5 corresponds to the ones of $\mathbf{x}$ in C_5 and add them. This gives us a vector which has 2 ones in the correct position in C_5. The rest of the vector has all zeros. We then add to this vector row 6 or 7 depending on which has the same configuration as $\mathbf{x}$ in C_3. Depending on the position of the 2 ones of $\mathbf{x}$ in C_5, this addition can disturb the configuration in C_5 by either adding two additional ones or flipping one of the desired ones and adding another one. In any case, we can select two appropriate rows from the first 5 rows of S_{even} and add them to this vector to obtain the desired configuration. The only case left is when the configuration of $\mathbf{x}$ in C_3 is $\begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$. Such a vector can be obtained by adding the last two rows of S_{even}, which gives us the correct configuration in C_3. The remaining two ones can be obtained by adding 2 appropriate rows from the first 5 rows.

3. The vector $\mathbf{x}$ has 1 one C_3 and the remaining 3 in C_5. We pick 3 rows from the first 5 in S_{even} which have ones in the corresponding positions in C_5. This gives us the configuration $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$ in C_3. If the desired configuration in C_3 is $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ or $\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$, we can add the row from S_{even} having $\begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$ as its configuration in C_3, respectively. Finally we add two of the first 5 rows to return to the correct configuration in C_5. If the desired configuration in C_3 is $\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$, we simply add the last two rows together to obtain the desired $\mathbf{x}$.

4. The vector $\mathbf{x}$ has all 4 ones in C_5. In this case we add the 4 rows from the first 5 rows of S_{even} which have ones in the correct positions giving us the vector $\mathbf{x}$. This sum will have all zeros in C_3.

It is easy to see that the rows of S_{even} are linearly independent. For, the first 5 rows cannot be written as the linear combination of each other due to their configuration in C_5. Row 6 cannot be written as the linear combination of the first 5 due to its configuration in C_3. The same applies for the last row.

For odd r, we consider the example with $n = 8$ and $r = 3$. We have the following 8×8 matrix whose rows span X:

$$S_{\mathsf{odd}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{16}$$

This can be proved in a similar manner as in the case with $n = 8$ and $r = 4$. More specifically, we imagine a split between the first 2 and the last 6 columns of S_{odd}, and consider the different cases.

Now, we show why the discrepancy between odd and even r exists. We begin with the case of odd r. Consider the $i \times (i + j)$ matrix of the form

$$S_1 = \begin{pmatrix} 1 & 1 & \cdots & 1 & 0 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ & & & & \vdots & \vdots & & & & \\ 1 & 0 & \cdots & 1 & 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 1 & 1 & 1 & 1 & 0 & \cdots & 0 \end{pmatrix}$$

Consider also the $j \times (i + j)$ matrix of the form

$$S_2 = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 & 1 & 0 & 1 & 0 & \cdots & 0 \\ & & & & \vdots & \vdots & & & & \\ 1 & 1 & \cdots & 1 & 1 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Pick any row $\mathbf{s}$ in S_1. If $i - 1$ is even then the sum of the $i - 1$ rows of S_1 excluding $\mathbf{s}$, denoted $\mathbf{s}'$, agrees with $\mathbf{s}$ in C_i. This is true since for the position of the 0 of $\mathbf{s}$ in C_i, there are $i - 1$ ones in the sum, which is an even number. On the other hand, for the rest of the positions in C_i there are $i - 2$ ones, an odd number. The positions $i + 1$ and $i + 2$ have zeros, since $i - 1$ is even. We can then add the first two rows of S_2 to this sum to obtain $\mathbf{s}$.

On the other hand if $i-1$ is odd then $\mathbf{s}'$ is a vector which is an inverted form of $\mathbf{s}$ in C_i. Furthermore, it has a 1 in each of the positions $i + 1$ and $i + 2$. To convert this to the correct form, we need to add to it one of the vectors from S_2. But such a vector also has another 1 in C_j. Adding such a vector does not give us the desired configuration. The same is true for any combination of vectors from S_2.

We have the following theorem whose proof is a straightforward generalization of above.

Theorem 3. *Let $n \geq 2$ and let $0 < r < n$. Let S_1 be the $(n - r + 1) \times n$ matrix*

$$\begin{pmatrix} \mathbf{1}_{n-r+1,r-1} & I_{n-r+1} \end{pmatrix},$$

and let S_2 be the $(r - 1) \times n$ matrix

$$\begin{pmatrix} \mathbf{1}_{r-1} - I^M_{r-1} & \mathbf{1}_{r-1,2} & \mathbf{0}_{r-1,n-r-1} \end{pmatrix}.$$

Further, let S' be the $n \times n$ matrix

$$\begin{pmatrix} S_1 \\ S_2 \end{pmatrix}.$$

Then, if r is odd, the rows of the matrix $S = S'$ are linearly independent and span X. If r is even, the rows of the matrix S, obtained by removing the last row of S', are linearly independent rows and span X.

The following corollary can be proved with the standard techniques in linear algebra.

Corollary 1. *Let S be the spanning set of X as defined in the above theorem. Then, any spanning set of X with vectors from X has $|S|$ vectors.*

From this it follows that

Corollary 2. *Let C be a random $n \times n$ binary matrix each row of which has Hamming weight $r = \lfloor \frac{n}{2} \rfloor$. Then $\mathsf{rank}(C) \leq n-1$ if r is even and $\mathsf{rank}(C) \leq n$ otherwise.*

C Solution to the Recurrence Relation $P(r, c)$

To find the solution to Eq. (4) we use the notion of generating functions [39, §7].[10] Let $G(x, y)$ be the generating function

$$G(x,y) = \sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c. \tag{17}$$

Multiplying the above by $1 - x - y$, we get

$$\begin{aligned} G(x,y)(1-x-y) &= \sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c \qquad (18)\\ &\quad - x\sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c \qquad (19)\\ &\quad - y\sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c. \qquad (20)\end{aligned}$$

Now,

$$\begin{aligned}\sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c &= \sum_{r\geq 1}\sum_{c\geq 1} P(r,c)x^r y^c + \sum_{r\geq 1} P(r,0)x^r \\ &\quad + \sum_{c\geq 1} P(0,c)y^c + P(0,0). \end{aligned}\tag{21}$$

This implies that

$$\begin{aligned} x\sum_{r\geq 0}\sum_{c\geq 0} P(r,c)x^r y^c &= x\sum_{r\geq 1}\sum_{c\geq 1} P(r,c)x^r y^c + x\sum_{r\geq 1} P(r,0)x^r \\ &\quad + x\sum_{c\geq 1} P(0,c)y^c + xP(0,0). \end{aligned}\tag{22}$$

Now note that

$$x\sum_{r\geq 1} P(r,c)x^r = \sum_{r\geq 1} P(r-1,c)x^r - xP(0,c).$$

Thus,

$$\begin{aligned} x\sum_{r\geq 1}\sum_{c\geq 1} P(r,c)x^r y^c &= \sum_{c\geq 1} y^c \left(x\sum_{r\geq 1} P(r,c)x^r \right) \\ &= \sum_{c\geq 1}\sum_{r\geq 1} P(r-1,c)x^r y^c - x\sum_{c\geq 1} P(0,c)y^c. \end{aligned}$$

[10] The derivation in this section follows the sketch described in http://math.stackexchange.com/questions/206158.

In particular

$$x\sum_{r\geq 1}P(r,0)x^r=\sum_{r\geq 1}P(r-1,0)x^r-xP(0,0).$$

Putting these results in Eq. (22), we get

$$\begin{aligned}x\sum_{r\geq 0}\sum_{c\geq 0}P(r,c)x^ry^c&=\sum_{c\geq 1}\sum_{r\geq 1}P(r-1,c)x^ry^c-x\sum_{c\geq 1}P(0,c)y^c\\&+\sum_{r\geq 1}P(r-1,0)x^r-xP(0,0)\\&+x\sum_{c\geq 1}P(0,c)y^c+xP(0,0)\\&=\sum_{c\geq 1}\sum_{r\geq 1}P(r-1,c)x^ry^c+\sum_{r\geq 1}P(r-1,0)x^r.\end{aligned}\tag{23}$$

By symmetry

$$y\sum_{r\geq 0}\sum_{c\geq 0}P(r,c)x^ry^c=\sum_{c\geq 1}\sum_{r\geq 1}P(r,c-1)x^ry^c+\sum_{c\geq 1}P(0,c-1)y^c.\tag{24}$$

Putting in the values from Equations 21, 23 and 24 into 18, 19 and 20 respectively, we get

$$\begin{aligned}G(x,y)(1-x-y)&=\sum_{r\geq 1}\sum_{c\geq 1}P(r,c)x^ry^c+\sum_{r\geq 1}P(r,0)x^r\\&+\sum_{c\geq 1}P(0,c)y^c+P(0,0)\\&-\sum_{c\geq 1}\sum_{r\geq 1}P(r-1,c)x^ry^c-\sum_{r\geq 1}P(r-1,0)x^r\\&-\sum_{c\geq 1}\sum_{r\geq 1}P(r,c-1)x^ry^c-\sum_{c\geq 1}P(0,c-1)y^c\\&=\sum_{r\geq 1}\sum_{c\geq 1}\left(P(r,c)-P(r-1,c)-P(r,c-1)\right)x^ry^c\\&+\sum_{r\geq 1}\left(P(r,0)-P(r-1,0)\right)x^r\\&+\sum_{c\geq 1}\left(P(0,c)-P(0,c-1)\right)y^c\\&+P(0,0).\end{aligned}$$

Now, by definition $P(r,c)=P(r-1,c)+P(r,c-1)$, $P(r,0)=1$ for all $r\geq 0$ and $P(0,c)=1$ for all $c\geq 0$. Therefore, all the terms on the right hand side above vanish except $P(0,0)=1$. We therefore get

$$G(x,y)=\frac{1}{1-x-y}.\tag{25}$$

Now assume that $x + y < 1$, then the right hand side above is the sum of the geometric series whose first term is 1 and common ratio is $x + y$. Therefore,

$$
\begin{aligned}
\frac{1}{1-x-y} &= \sum_{i\geq 0}(x+y)^i \\
&= \sum_{i\geq 0}\sum_{j=0}^{i}\binom{i}{j}x^{i-j}y^j \\
&= \sum_{j=0}^{0}\binom{0}{j}x^{0-j}y^j + \sum_{j=0}^{1}\binom{1}{j}x^{1-j}y^j + \sum_{j=0}^{2}\binom{2}{j}x^{2-j}y^j + \cdots \\
&= \binom{0}{0}x^0y^0 \\
&+ \binom{1}{0}x^1y^0 + \binom{1}{1}x^0y^1 \\
&+ \binom{2}{0}x^2y^0 + \binom{2}{1}x^1y^1 + \binom{2}{2}x^0y^2 \\
&+ \cdots \\
&= \binom{0}{0}x^0y^0 + \binom{1}{1}x^0y^1 + \binom{2}{2}x^0y^2 + \cdots \\
&+ \binom{1}{0}x^1y^0 + \binom{2}{1}x^1y^1 + \binom{3}{2}x^1y^2 + \cdots \\
&+ \binom{2}{0}x^2y^0 + \binom{3}{1}x^2y^1 + \binom{4}{2}x^2y^2 + \cdots \\
&+ \cdots \\
&= \sum_{j\geq 0}\binom{j}{j}x^0y^j + \sum_{j\geq 0}\binom{j+1}{j}x^1y^j + \sum_{j\geq 0}\binom{j+2}{j}x^2y^j + \cdots \\
&= \sum_{i\geq 0}\sum_{j\geq 0}\binom{i+j}{j}x^iy^j. \qquad (26)
\end{aligned}
$$

Thus, Eq. (25) becomes

$$G(x,y) = \sum_{i\geq 0}\sum_{j\geq 0}\binom{i+j}{j}x^iy^j. \qquad (27)$$

Comparing Eq. (17) with Eq. (27), we finally get

$$P(r,c) = \binom{r+c}{c} = \binom{r+c}{r}.$$

D Corrections to Section 6

This section contains corrections to Section 6. The corrections do not change the final result of the section, but are essential to show that the linearization of Foxtail still works.[11] First note that Eq. 6

[11] The same corrections also apply to Section VI of the version of this paper titled *On the Linearization of Human Identification Protocols: Attacks based on Linear Algebra, Coding Theory and Lattices.*

is written incorrectly. This should be re-written as:

$$\begin{aligned}\mathsf{ft}(\mathbf{x},\mathbf{c}) &= \left\lfloor \frac{\mathbf{c}\cdot\mathbf{x}^T \bmod 4}{2} \right\rfloor \\ &= \left\lfloor \frac{1}{2}\left(\left(\sum_i c_{i,0}x_i + \sum_i 2c_{i,1}x_i\right) \bmod 4\right)\right\rfloor \\ &= \left\lfloor \frac{1}{2}\left(\left(\sum_i c_{i,0}x_i \bmod 4 + \sum_i 2c_{i,1}x_i \bmod 4\right) \bmod 4\right)\right\rfloor . \end{aligned} \tag{28}$$

We now prove the following lemma.

Lemma 5. *Let $a \in \{0,1,2,3\}$ and $b \in \{0,2\}$. Then*

$$\left\lfloor \frac{1}{2}\left((a+b) \bmod 4\right)\right\rfloor = \left(\left\lfloor \frac{a}{2}\right\rfloor + \left\lfloor \frac{b}{2}\right\rfloor\right) \bmod 2.$$

Proof. By simply enumerating all possible combinations of values of a and b and evaluating the left hand and the right hand sides (denoted LHS and RHS, respectively), we get the following table

a	b	LHS	RHS
0	0	0	0
0	2	1	1
1	0	0	0
1	2	1	1
2	0	1	1
2	2	0	0
3	0	1	1
3	2	0	0

which shows that LHS = RHS as required. □

Now by substituting $a = \sum_i c_{i,0}x_i \bmod 4$ and $b = \sum_i 2c_{i,1}x_i \bmod 4$ from Eq. 28 into the statement of the above lemma,[12] we get

$$\begin{aligned}\mathsf{ft}(\mathbf{x},\mathbf{c}) &= \left\lfloor \frac{1}{2}\left(\left(\sum_i c_{i,0}x_i \bmod 4 + \sum_i 2c_{i,1}x_i \bmod 4\right) \bmod 4\right)\right\rfloor \\ &= \left(\left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right)\right\rfloor + \left\lfloor \frac{1}{2}\left(\sum_i 2c_{i,1}x_i \bmod 4\right)\right\rfloor\right) \bmod 2 \\ &= \left(\left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right)\right\rfloor + \left\lfloor \sum_i c_{i,1}x_i \bmod 2\right\rfloor\right) \bmod 2 && \text{By Lemma 2} \\ &= \left(\left\lfloor \frac{1}{2}\left(\sum_i c_{i,0}x_i \bmod 4\right)\right\rfloor + \sum_i c_{i,1}x_i \bmod 2\right) \bmod 2 && \text{Being an integer} \\ &= \left(\sum_i \sum_{j>i} c_{i,0}c_{j,0}x_ix_j \bmod 2 + \sum_i c_{i,1}x_i \bmod 2\right) \bmod 2 && \text{By Theorem 1} \\ &= \left(\sum_i \sum_{j>i} c_{i,0}c_{j,0}x_ix_j + \sum_i c_{i,1}x_i\right) \bmod 2, \end{aligned}$$

from which Eq. 8 follows.

[12] Note that $\sum_i 2c_{i,1}x_i \bmod 4 \in \{0,2\}$.