Authentication Protocol using Trapdoored Matrices

by

Aikaterini Sotiraki

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Author ... Signature redacted ....
Department of Electrical Engineering and Computer Science
November 16, 2015

Certified by.................................. Signature redacted
Ronald L. Rivest
Professor
Thesis Supervisor

Accepted by .... Signature redacted
Leslie A. Kolodziejski
Graduate Officer, Department Committee on Graduate Theses

Authentication Protocol using Trapdoored Matrices

by

Aikaterini Sotiraki

Submitted to the Department of Electrical Engineering and Computer Science
on November 16, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In this thesis, we propose a new type of public-key authentication protocol, which is based on timing gaps. The honest user's secret key allows him to perform a task faster than any adversary. In our construction, the public key is a $n \times n$ matrix and the secret key is a trapdoor of this matrix. The task which an honest user has to perform in order to authenticate himself is a matrix-vector multiplication, where the vector is supplied by the verifier. We provide specific constructions of the trapdoor and analyze them both theoretically and practically.

Thesis Supervisor: Ronald L. Rivest
Title: Professor

Acknowledgments

This work was supported by Paris Kanellakis Fellowship and Akamai Fellowship.

I would like to express my gratitude to my supervisor, Ron Rivest, for the useful discussions and engagement through the learning process of this master thesis. Also, I would like to thank Yuval Peres for the very helpful feedback on the theory of Markov chains.

Contents

Chapter 1

Authentication protocols

1.1 Introduction

Authentication is a problem that arises in various settings and there are many different proposals for solving it. In this thesis, we consider authentication in the public-key setting. We start by explaining what authentication is and describing some previous protocols for achieving it. Afterwards, we provide our suggestion. In the following chapters, we present the necessary background for analyzing our protocol. We give a more concrete description of the protocol and we conclude with some considerations on its implementability.

In this chapter, we define what authentication is. We outline our proposal and describe some previous work on authentication.

1.1.1 Authentication Protocols

An interactive public-key authentication protocol is a protocol in which there are two parties, a prover and a verifier. The prover wants to authenticate herself to the verifier by proving the possession of the secret key connected to her public key. During the setup phase of the protocol, the verifier acquires some information that he trusts regarding the prover. For instance, a trusted authority might tie together the public key and the name of the prover in a certificate, which the verifier can then access, or the prover might add his public key to a public directory. If the execution of the protocol is successful, then the verifier knows that the information the prover provides is consistent with the trusted information.

Usually an authentication protocol consists of two stages. In the first one, the verifier poses a challenge to the prover. In the second one, the prover proves that she knows the secret key by answering the challenge correctly. The development of zero-knowledge proofs in this setting is very useful. The prover wants to authenticate herself, but at the same time she needs to make sure that the verifier, or any eavesdropper, learns nothing about her secret key, apart from the fact that she knows it. This is exactly what a zero-knowledge proof of knowledge offers [9].

The notion of authentication we use is the following [8]:

Definition 1. *A scheme is an authentication scheme if A can prove to B that she is A, but someone else cannot prove to B that she is A.*

The definition of a secure authentication system scheme is, then, similar to that of a secure identification system scheme of Feige et al. [7]:

Definition 2. *An authentication system scheme is secure if for any polynomial ensemble of users the probability of an impersonation event in a randomly chosen authentication system is negligible. In this case, an impersonation event is the event that at least once during the lifetime of the system one user succeeds in authenticating himself as some other good user, namely a user that throughout the lifetime of the system does not deviate from the protocol dictated by the scheme.*

Some examples of authentication protocols are a protocol by Fiat and Shamir [8] and a protocol by Schnorr [20]. Both of these protocols base their security on some number theoretic problems which are believed to be intractable. The prover's secret key is a quantity that no dishonest party can compute because of the intractability assumption. As usual, the authentication protocol is based on answering the challenge posed by the verifier correctly in zero-knowledge. The protocol is secure if no dishonest party can answer correctly the challenge in polynomial time.

Fiat-Shamir protocol

At CRYPTO'86 conference, Fiat and Shamir presented an authentication protocol based on the difficulty of factoring [8]. In their protocol, they assume the existence of a trusted center. This center publishes the universal parameters:

1. a large composite number n, which is the product of two primes, p and q,
2. a pseudo-random function, f, which maps arbitrary strings to the range $[0, n)$,
3. two small integers k and t.

The center, also, issues the private keys of the users with the following procedure:

- Create a string I with all the information about the user and the card.
- Compute the values $v_j = f(I, j)$ for small values of j.
- Pick k distinct values of j such that v_j is a quadratic residue modulo n.
- For each j that was picked, compute the smallest square root s_j of v_j^{-1}, i.e. the smallest s_j such that $s_j^2 \cdot v_j = 1 \pmod{n}$.
- Issue a smart card which contains I and the k values s_j with their indices.

Then, the authentication protocol is the following, where we assume that the k values of j used are $\{1, ..., k\}$:

1. The prover sends I to the verifier.
2. The verifier computes $v_j = f(I, j)$ for $j \in \{1, ..., k\}$.
3. Repeat the following steps for $i = 1, ..., t$:
   (a) The prover picks a random $r_i \in [0, n)$ and sends $x_i = r_i^2 \pmod{n}$ to the verifier.
   (b) The verifier sends a vector $(e_{i1}, ..., e_{ik}) \in \{0, 1\}^k$ to the prover.
   (c) The prover sends $y_i = r_i \prod_{e_{ij}=1} s_j \pmod{n}$ to the verifier.

   (d) The verifier checks if $x_i = y_i^2 \prod_{e_{ij}=1} v_j \pmod{n}$.

4. If the t checks are successful, the verifier accepts. Otherwise, he rejects.

Schnorr protocol

At CRYPTO'89 conference, Schnorr presented another authentication protocol based on the difficulty of computing the discrete logarithm [20]. In this protocol, the universal parameters published by the trusted center are the following:

1. two large primes, p and q such that $q|(p-1)$,
2. an $a \in \mathbb{Z}_p$ with order q,
3. a small integer t.

The center has its own public/private key pair for signing the users' public keys. It registers a user by creating a string I with all the information about the user and the card and signing (I, v), where v is the user's public key.

Each user generates his private key s and his public key v by himself:

- The private key s is a random number in $\{1, 2, ..., q\}$.
- The public key v is the number $v = a^{-s} \pmod{p}$.

Then, the authentication protocol is the following:

1. The prover sends I, v and the trusted center's signature of (I, v) to the verifier.
2. The verifier verifies the center's signature transmitted by the prover.
3. The prover picks a random $r \in \{1, ..., q-1\}$ and sends $x = a^r \pmod{p}$ to the verifier.
4. The verifier sends a random number $e \in \{0, ..., 2^t - 1\}$ to the prover.
5. The prover sends $y = r + s \cdot e \pmod{p}$ to the verifier.
6. The verifier checks if $x = a^y \cdot v^e \pmod{p}$.
7. If the check is successful, the verifier accepts. Otherwise, he rejects.

1.2 Time-based authentication protocols

In this work, we present an alternative idea for creating authentication protocols. The major component in our protocol is not the response to a challenge, but the time needed to provide this response. The verifier keeps track of the time that the prover needs in order to respond correctly. Only if this time is small enough and the answer is correct does the prover authenticates herself successfully. More precisely, the secret key is a trapdoor that gives the honest party a polynomial time advantage in answering the challenge.

Time plays an important role in many cryptographic protocols, especially as far as their security is concerned. Many protocols, such as Diffie-Hellman, RSA and DSS, can be broken using timing attacks [11]. In these cases, measuring the difference between the time needed to perform different computations gives the adversary a non-negligible advantage and breaks the security of the protocol. Other identification protocols (e.g. [5]) use the timing delay between the challenge and the response as a "distance bounding" technique. This technique is then integrated in an identification protocol, so that the verifier can verify not only the identity of the prover, but also that she is close by. A case in which this is helpful is the identification at the entrance to a building, where the prover has to show that she is present there.

In contrast to other authentication protocols, in our protocol, the dishonest party is able to find the correct answer to the challenge, but to do so he needs an unacceptable amount of time. A polynomial gap between the time of the honest party and the best time of the adversary is enough to make our protocol work. This notion of polynomial gap is not new in cryptography. For example, Merkle's puzzles [13], an early proposal related to public-key cryptography, were based in the quadratic difference in the complexity for the honest and the dishonest party.

Informally, an authentication protocol based on timing gaps is secure if, given the public key of a user, no polynomial adversary can find a way to accelerate its computation enough to fool the verifier. To facilitate the description and construction of such protocol, we need to make some assumptions. As mentioned before, a very important assumption is that the verifier has obtained the prover's correct public key. Then, he checks that the prover can authenticate herself in a way consistent with the trusted information. Furthermore, we assume that the person authenticating herself with a time-based protocol has an implementation whose running time is independent of the starting time of the protocol execution. This means that we can consider that the time the prover needs to answer the challenge is constant over time. Similarly, we assume that the time needed to complete the protocol is independent of the trapdoor and the challenge. So, no timing attacks can be applied to our protocol.

Our proposal suggests that even a polynomial gap between the time needed to successfully complete a challenge with and without the secret key is enough to construct an authentication protocol. Such a protocol could be used in authentication via smart cards. For instance, the verifier could be an ATM and the prover could be the smart card that a customer uses.

1.2.1 General form of the protocol

The general form of a scheme based of a polynomial timing gap is shown in Figure 1-1. In our case, we are not interested in problems that are solvable to someone that possesses certain information, but are intractable without this information. We are interested in problems that can be solved by anyone, but *faster* with certain trapdoor information than without it.

1.2.2 Protocol based on computing $a^{2^t} \pmod{n}$

We demonstrate how a time-based authentication protocol works with a protocol that is based on the ideas about time-lock puzzles [19]. A time-lock puzzle is a puzzle which no one, except maybe its creator, can solve without running an algorithm for at least a certain amount of time. In our case, the creator of the puzzle is the prover; this gives her the ability to find the solutions faster than anyone else.

The protocol is the following:

- The prover generates a composite modulo $n = pq$ and published it as her public

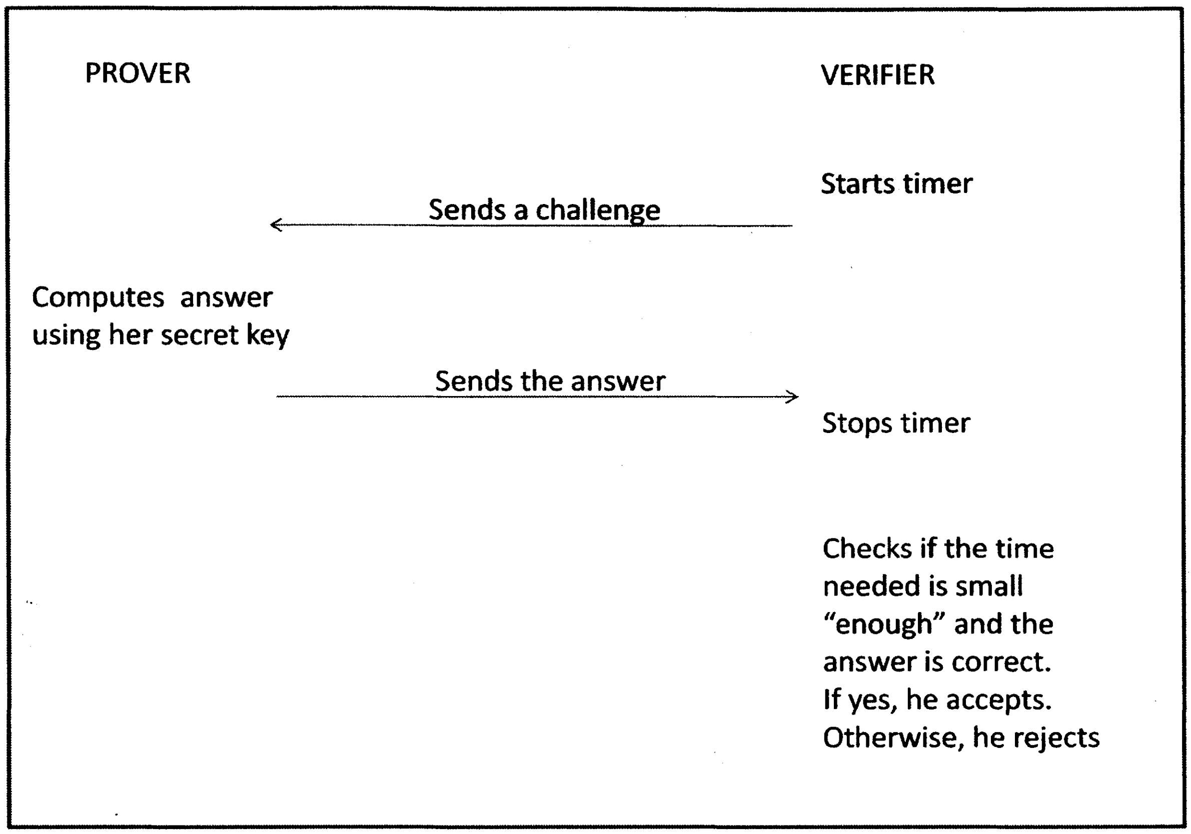
PROVER
VERIFIER
Starts timer
Sends a challenge
Computes answer using her secret key
Sends the answer
Stops timer
Checks if the time needed is small "enough" and the answer is correct. If yes, he accepts. Otherwise, he rejects

Figure 1-1: Time-based authentication protocol.

key. The prime numbers p and q are the secret key.

- Whenever the prover wants to authenticate herself to the verifier, she asks for a number $t \in \mathbb{N}$.
- The verifier generates a random, but of appropriate size, $t \in \mathbb{N}$ and a random $a \in \mathbb{Z}_n^*$ and sends them to the prover.
- The prover computes quickly, using the factorization of n, the quantity $c = 2^t \pmod{\phi(n)}$ then she sends back to the verifier $y = a^c \pmod{n}$.
- The verifier verifies that $y = a^{2^t} \pmod{n}$ by performing the usual exponentiation, using the publicly available modulo n.

- If y is correct and the time the prover needed is "sufficiently small", then the verifier accepts. Otherwise he rejects.

Without knowing the factorization of n, the fastest way to compute $a^{2^t} \pmod{n}$ seems to be to start with a and perform t squarings sequentially. So, the fact that the prover knows the factors p and q gives her an advantage in the time needed to find the desired result.

In the above protocol, we mention that the number t that the verifier chooses should be of appropriate size. If t is too small, then the time needed by the prover does not differ a lot by the time an adversary needs, which might make it difficult for the verifier to distinguish honest from dishonest parties. On the other hand, if t is too large, then the time needed by the verifier in order to compute $a^{2^t} \pmod{n}$ might be too much, which again is a problem since the verifier cannot accept or reject before computing this quantity by himself.

1.2.3 Protocol based on Trapdoored Matrices

Another suggestion for implementing time-based authentication is through trapdoored matrices. This is the type of protocol we are concerned in the next chapters.

In this protocol, a user's public key consists of an $n \times n$ matrix A over $GF(q)$, where q is fixed, and her secret key is a trapdoor of A. The challenge is to perform the computation Ax for a random n-vector x as fast as possible. The protocol is of the following form:

- The prover creates a trapdoored $n \times n$ matrix A over $GF(q)$ and publishes it as her public key.

- Whenever the prover wants to authenticate herself to the verifier, she asks for a n-vector x with elements over $GF(q)$.

- The verifier generates a random vector $x \in GF(q)^n$ and sends it to the prover.

- The prover computes very quickly, using the trapdoor, the quantity $y = Ax$ and sends it back to the verifier.

- The verifier verifies whether $y = Ax$ in $\Theta(n^2)$ time with the naïve matrix-vector multiplication using the public matrix A.
- If y is correct and the time the prover needed is "sufficiently small", then the verifier accepts. Otherwise, he rejects.

We denote the trapdoor of matrix A by $T(A)$. This trapdoor must give a significant advantage to the prover in terms of the time she needs in order to perform the operation Ax. For instance, the trapdoor could be a compact straight-line program for computing Ax. Simultaneously, it should be difficult for an adversary to find a trapdoor, which would allow him to perform this operation quickly enough to trick the verifier.

In the next chapters, we describe a candidate construction of the trapdoored matrix A.

Chapter 2

Theoretical background

In the chapter, we give some necessary background for analyzing the construction of trapdoored matrices. For this construction we need some basic notions of linear algebra and of Markov chains. We start this chapter with some definitions and facts of linear algebra. We continue with an overview of Markov chains, focusing especially on methods we will use afterwards.

In the next chapter, we present some analysis of the asymptotic complexity of matrix-vector multiplication over $GF(q)$ using the trapdoor. We want this complexity to be smaller than the best known algorithm for matrix-vector multiplication. So, we conclude this chapter with a brief presentation of the previous works on measuring the complexity of matrix-vector multiplication.

2.1 Linear Algebra

In the proposed protocol, we pick our trapdoored matrix to be full rank. Even though we do not really need invertibility, the complexity of the matrix-vector multiplication for full rank matrices is at least as large as that for those with lower rank. Since we aim at distinguishing honest from dishonest parties based on small time differences in computing a matrix-vector multiplication, we want to minimize the time needed by the honest user. This is done by using the best possible trapdoor. Simultaneously, we want to maximize the time needed by any adversary. Therefore, it makes sense to restrict our attention to full rank matrices.

We denote the i-th row of an $n \times n$ matrix A by A_i. Then, from elementary linear algebra:

Definition 3. *An* elementary row operation *on a matrix A is one of the following:*

1. *Row switching: Switch row i and row j of the matrix ($A_i \leftrightarrow A_j$)*
2. *Row multiplication: Replace row i by its multiplication with a non-zero k ($A_i \leftarrow k \cdot A_i$)*
3. *Row addition: Substitute row i by its sum with a multiple of row j ($A_i \leftarrow A_i + k \cdot A_j$)*

Definition 4. *An* elementary matrix *is a matrix that is produced by performing a single elementary row operation on the identity matrix.*

Theorem 5. *An $n \times n$ matrix is invertible if and only if it can be written as a product of elementary matrices.*

The representation of an invertible matrix as a product of elementary matrices is closely related to Gaussian elimination. Each step of Gaussian elimination is a single elementary row operation, so it can be represented as a multiplication with an elementary matrix. This means that every invertible matrix can be written as a product of at most $O(n^2)$ elementary matrices.

For our purpose we use a slightly different notion of elementary row operation:

Definition 6. *A* general row operation *on a matrix A is the substitution of row A_i by $a \cdot A_i + b \cdot A_j$ where j is another row and $a \in GF(q) \setminus \{0\}$, $b \in GF(q)$.*

Similarly to the above, we define the notion of a *general elementary matrix.*

Each general row operation can be expressed as a sequence of at most 2 elementary row operations, a row multiplication (if $a \neq 1$) and a row addition (if $b \neq 0$). Also:

Lemma 7. *Every elementary row operation on a matrix A can be expressed as sequence of at most 3 general row operations on A.*

Proof.
- Row switching:
  1. $A_i \leftarrow 1 \cdot A_i - 1 \cdot A_j$
  2. $A_j \leftarrow 1 \cdot A_j + 1 \cdot A_i$
  3. $A_i \leftarrow -1 \cdot A_i + 1 \cdot A_j$
- Row multiplication: $A_i \leftarrow k \cdot A_i + 0 \cdot A_j$
- Row addition: $A_i \leftarrow 1 \cdot A_i + k \cdot A_j$

□

So, from Theorem 5 and Lemma 7 we have that

Proposition 8. *A matrix can be written as a product of $O(n^2)$ general elementary matrices if and only if it is invertible.*

Theorem 8 is the basic tool we use for creating the trapdoor. Each trapdoored matrix is the product of $L(n)$ general elementary matrices. The prover knows these matrices and their order in the multiplication, while the verifier only knows the resulting product.

In next chapters, we also use the following proposition:

Proposition 9. *A general elementary matrix is invertible and its inverse is also a general elementary matrix.*

Proof. If A is a general elementary matrix corresponding to the row operation $A_i \leftarrow a \cdot A_i + b \cdot A_j$ with $a \neq 0$, then the general elementary matrix corresponding to the row operation $A_i \leftarrow a^{-1} \cdot A_i - a^{-1} \cdot b \cdot A_j$ is its inverse. □

2.2 Markov Chains

As we mentioned in the previous section, each public key is the product of general elementary matrices. We denote the number of elementary matrices on the product, which also determines the size of the trapdoor, by $L(n)$. We know that each invertible matrix can be written as a product of general elementary matrices (Proposition 8). So, in order to analyze the security of the protocol, we need to find the size of $L(n)$ for which it is hard to distinguish a trapdoored matrix produced as a product of $L(n)$ random general elementary matrices from a completely random invertible matrix. This question is strongly related to the theory of Markov chains. This section mentions the basic definitions and methods we need ([14], [2], [12]).

2.2.1 Basic Definitions and Theorems

We start by defining what a Markov chain is:

Definition 10. *A sequence of random variables X_0, X_1, X_2, ... is a* Markov chain *with state space Ω if for all $a_0, a_1, ..., a_t \in \Omega$ it satisfies the Markov property:*

$$Pr(X_t = a_t | X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, , X_0 = a_0) = Pr(X_t = a_t | X_{t-1} = a_{t-1})$$

If Ω is finite, then the Markov chain is called finite Markov chain.

For the rest of the section, we consider only finite Markov chains.

Definition 11. *The transition probability*

$$P_{i,j} = Pr(X_t = j | X_{t-1} = i)$$

is the probability that the process moves from state i to j in one step.
The Markov property implies that the Markov chain is uniquely defined by the one-step transition matrix:

$$P = \begin{pmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,j} & \dots & P_{1,n} \\ P_{2,1} & P_{2,2} & \dots & P_{2,j} & \dots & P_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{i,1} & P_{i,2} & \dots & P_{i,j} & \dots & P_{i,n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{n,1} & P_{n,2} & \dots & P_{n,j} & \dots & P_{n,n} \end{pmatrix}$$

This matrix is called the transition matrix *of the Markov chain.*

Definition 12. *A Markov chain is* irreducible *if all states belong to one communicating class. Namely, for any two states $i, j \in \Omega$ there exists an integer t such that $P^t(i,j) > 0$.*

Definition 13. *A state i in a discrete time Markov chain is* periodic *if there exists an integer $K > 1$ such that $P(X_{t+s} = i | X_t = i) = 0$ unless s is divisible by K.*
A discrete time Markov chain is periodic *if any state in the chain is periodic.*
A state or a chain that is not periodic is aperiodic.

A very useful notion in Markov chains is that of the stationary distribution, which shows the limiting distribution over the state space of the chain. We are interested in finding the stationary distribution of specific chains and arguing that there is only one stationary distribution. So, the following propositions are helpful.

Definition 14. *A* stationary distribution *(also called an* equilibrium distribution*) π of a Markov chain is a probability distribution π such that:*

$$\pi = \pi \cdot P$$

Definition 15. *A matrix is called* doubly stochastic *if its row sums and its column sums are 1.*

Theorem 16. *The stationary distribution of a Markov chain with doubly stochastic transition matrix is the uniform distribution over Ω.*

Definition 17. *A state i is called* absorbing *if the chain never leaves i once it first visits i. Namely, $P_{i,i} = 1$ and, of course, $P_{i,j} = 0$, for all $j \neq i$.*

If every state can reach an absorbing state in a finite number of steps, then the chain is called absorbing Markov chain.

Proposition 18. *([12] Proposition 1.26) If an absorbing Markov chain has a unique absorbing state i, then it has as unique stationary distribution the degenerate distribution at i.*

Another very useful notion is that of mixing time, which shows how quickly a Markov chain approaches its stationary distribution. We formally define this notion and in the next section, we describe famous methods for bounding it.

Definition 19. *The* variation distance *between two distributions D_1 and D_2 on a countable state space Ω is given by*

$$||D_1 - D_2|| = \frac{1}{2}\sum_{i\in\Omega} |D_1(i) - D_2(i)|$$

Definition 20. *Let π be the stationary distribution of a Markov chain with state space Ω. Let P_i^t represent the distribution of the state of the chain starting at state i after t steps. We define*

$$\Delta_x(t) = ||P_i^t - \pi||$$

and

$$\Delta(t) = \max_{i\in\Omega} \Delta_i(t).$$

That is, $\Delta_i(t)$ is the variation distance between the stationary distribution and P_i^t, and $\Delta(t)$ is the maximum of these values over all starting states i.
We also define

$$\tau_i(\epsilon) = \min\{t : \Delta_i(t) \leq \epsilon\};$$

and

$$\tau(\epsilon) = \max_{i\in\Omega} \tau_i(\epsilon).$$

That is, $\tau_i(\epsilon)$ is the first step t at which the variation distance between P_i^t and the stationary distribution is less than ϵ, and $\tau(\epsilon)$ is the maximum of these values over all states i.

Definition 21. *When $\tau(\epsilon)$ is considered as a function of ϵ, it is called the* mixing time *of the Markov chain.*

We call (worst case) mixing time the quantity $\tau_{mix} = \tau(1/4)$.

Theorem 22. *(Convergence Theorem [12]) Suppose that P is irreducible and aperiodic. Then, it has a unique stationary distribution π. Also, there exist constants $\alpha \in (0,1)$ and $C > 0$ such that*

$$\max_{i\in\Omega} ||P_i^t - \pi||_{TV} \leq C\alpha^t.$$

2.2.2 Strong Stationary Time

One of the basic tools to analyze the mixing time of a Markov chain is the notion of strong stationary time. This section provides the basic definitions and theorems we need about the strong stationary time.

Definition 23. *Given a sequence $(X_t)_{t=0}^{\infty}$ of Ω-valued random variables, a $\{0,1,2,...,\infty\}$-valued random variable τ is a* stopping time *for (X_t) if, for each $t \in \{0,1,...\}$, there is a set $B_t \subset \Omega^{t+1}$ such that*

$$\{\tau = t\} = \{(X_0, X_1, ..., X_t) \in B_t\}.$$

In other words, a random time τ is a stopping time if and only if the indicator function $\mathbb{1}_{\tau=t}$ is a function of the vector $(X_0, X_1, \ldots, X_t)$.

Proposition 24. *Every transition matrix P on a finite state space has a random mapping representation: we can find a function $f : \Omega \times \Lambda \to \Omega$, along with a Λ-valued random variable Z, satisfying*

$$P(f(i,Z) = j) = P_{i,j}.$$

Therefore, if $(Z_t)_{t=1}^{\infty}$ is a sequence of independent random variables, each having the same distribution as Z, and X_0 has distribution μ, then the sequence $(X_t)_{t=0}^{\infty}$ defined by

$$X_t = f(X_{t-1}, Z_t, \text{ for } t \geq 1$$

is a Markov chain with transition matrix P with starting distribution μ.

Definition 25. *A random time τ is called a* randomized stopping time *for the Markov chain (X_t) if it is a stopping time for the sequence (Z_t).*

Definition 26. *A* strong stationary time *for a Markov chain (X_t) with stationary distribution π is a randomized stopping time τ, possibly depending on the starting position i, such that*

$$P_i(\tau = t, X_\tau = j) = P_i(\tau = t)\pi(j).$$

Theorem 27. *[12] If τ is a strong stationary time, then*

$$\Delta(t) \leq \max_{i \in \Omega} P_i(\tau > t).$$

2.3 Matrix-vector multiplication

The naïve algorithm for matrix-vector multiplication over finite fileds has complexity $\Theta(n^2)$. However, there are many algorithms that achieve better complexity. The most well-known algorithm for matrix-vector multiplication over $GF(2)$ with complexity less than $O(n^2)$ is known as the Four Russians' Algorithm ([4], [1]). Its running time it $O(\frac{n^2}{\log(n)})$, but it requires auxiliary storage of $O(\frac{n^2}{\log(n)})$ bits.

The best known algorithm to perform matrix-vector multiplication over $GF(q)$ [21] has running time $O(\frac{n^2}{(\epsilon \log(n))^2})$ for any $\epsilon \in (0, 1)$. An important aspect of this algorithm is that it requires some amount of preprocessing. In our protocol, preprocessing is permitted, since the $n \times n$ matrix is the public key. The adversary can spend any polynomial amount of time in preprocessing this matrix, so that when he receives the challenge vector he can perform the operation as fast as he can. More formally, the theorem (expressed for finite fields) of Williams [21] states:

Theorem 28. *Let $GF(q)$ be a finite field of q elements. For all $\epsilon \in (0, 1)$, every $n \times n$ matrix A over $GF(q)$ can be preprocessed in $O(n^{2+\epsilon \log_2(q)})$ time such that every subse-*

quent matrix-vector multiplication can be performed in $O(\frac{n^2}{(\epsilon \log(n))^2})$ *steps on a pointer machine or a* $(\log(n))$*-word RAM, assuming operations in* $GF(q)$ *take constant time.*

Apart from the known algorithms for matrix-vector multiplication, there are some known lower bounds for the problem and conjectures about what the lower bounds should be. A lower bound of $\Omega(n^2)$ arithmetic operations is known [22]. However, one of the assumptions of that result is that the underlying field should be infinite.

More recently Henzinger et al. [10] addresses the problem of Online Boolean Matrix-Vector multiplication. In this problem, we want to compute a $n \times n$ matrix multiplication $A \cdot B$, but we are given one column-vector of B, v_i, at each round. After seeing each vector v_i, we have to output the product $A \cdot v_i$ before receiving the next vector. The paper shows that a conjecture that there is no truly subcubic ($O(n^{3-\epsilon})$) algorithm for the Online Boolean Matrix-Vector multiplication problem can be used to prove the underlying polynomial hardness appearing in many dynamic problems. It is easy to see that this problem is a generalization of the Matrix-Vector multiplication in which we are interested.

Chapter 3

Protocol Analysis

We begin this chapter with a description of the adversarial model we use. Then, we outline our candidate construction for the trapdoor matrices. In the following sections, we analyze the security of the protocol using theoretical arguments and experimental results.

3.1 Adversarial model

To argue the security of a protocol, it is necessary to first describe the adversarial model considered. In our case, we assume that the adversary has "comparable" power with the honest user. That means that the adversary is restricted to polynomial time algorithms.

As we have already mentioned, we focus on the scenario of the authentication of an honest party to an ATM using smart cards. So, we consider adversaries who know the authentication protocol and its public parameters, the honest party's public key and the type of smart card used. Then, they use all the known information to create a fake smart card and try to fool the ATM. Since our protocol is based on a very parallelizable problem (matrix-vector multiplication) and smart cards have very limited computational capabilities, we assume that the adversary can use the best available smart card, even if this is better than that of the honest user. However, he cannot connect it to external devices that can perform faster, maybe parallel, computation.

In a later section, we try to find practical parameters for our protocol. Of course, the adversarial power affects these parameters. A concrete example is the size of the public key, which is a $n \times n$ matrix. On one hand, we need n to be as small as possible, so that the trapdoor and challenge fit in the storage of the smart card. On the other hand, the time to perform the regular matrix-vector multiplication depends on n. So, n has to be large enough, so that the adversary needs noticeably more time, say tenfold, than the honest user in order to find the result of the computation.

3.2 Candidate trapdoor construction

This section presents our construction for the trapdoored matrix A as a product of $L(n)$ random general elementary matrices. We argue that, given such a matrix A, it is difficult to factorize it as a product of $O(L(n))$ general elementary matrices. Then, the best strategy for any adversary would be to perform the fastest possible matrix-vector multiplication and return the result. If the complexity of matrix-vector multiplication is $f(n)$ and we pick $L(n) = o(f(n))$, then for large enough n the honest party has a noticeable timing advantage in computing the product Ax.

The procedure that a user follows to create a trapdoored matrix A is the following: For $k \in \{1, ..., L(n)\}$:

- Pick i_k uniformly at random from $\{1, ..., n\}$
- Pick j_k uniformly at random from $\{1, ..., n\} \setminus \{i_k\}$
- Pick a_k uniformly at random from $GF(q) \setminus \{0\}$
- Pick b_k uniformly at random from $GF(q)$

Then,

$$A = A^{(L)} \cdot A^{(L-1)} \cdot ...A^{(1)}$$

where $A^{(k)}$ is the general elementary matrix corresponding to the operation $A_{i_k} \leftarrow a_k \cdot A_{i_k} + b_k \cdot A_{j_k}$.

Thus, the trapdoor $T(A)$ consists of the sequence of $L(n)$ tuples:

$$(i_k, j_k, a_k, b_k), \text{ for } k = 1, 2, ..., L$$

We force $i_k \neq j_k$, because we want the matrix A to be invertible.

The program that the honest party uses in order to take advantage of the knowledge of the trapdoor begins by copying $y = x$, that is $y_i = x_i$ for $1 \leq i \leq n$. Then, the k-th step, for $1 \leq k \leq L$, has the form:

$$y_{i_k} = a_k \cdot y_{i_k} + b_k \cdot y_{j_k}.$$

The final value of $y = (y_1, y_2, ..., y_n)$ is the output.

The complexity for the honest user in $\Theta(L(n))$.

3.3 Protocol Security

The previous section describes how to create a trapdoored matrix using $L(n)$ random general elementary row operations. In this section, we estimate how big $L(n)$ should be so that the resulting matrix is indistinguishable from a random invertible matrix to any polynomial time adversary.

3.3.1 Markov Chain and Stationary distribution

The process of producing a trapdoored matrix yields a sequence of random variables with state space $\Omega = GL(n, q)$, the set of all invertible $n \times n$ matrices over $GF(q)$, which satisfies the Markov property. So, this process defines a finite Markov chain, M.

Claim 29. *The unique stationary distribution of M is the uniform on $GL(n, q)$.*

This claim is indispensable in showing the security of the protocol. If we allow $L(n)$, the number of tuples in the trapdoor $T(A)$, to be large enough, then we end up with an approximately uniformly random matrix in $GL(n, q)$. So, for large enough

$L(n)$, no adversary can tell whether a matrix was produced by the previous process or not.

We cannot hope to set $L(n)$ equal to τ_{mix} because we know a lower bound on τ_{mix} of $\Omega(n^2/\log_q(n))$. To see that, we create a graph with vertex set Ω and edges between states A and B for which $P_{A,B} + P_{A,B} > 0$. We know that the diameter, *diam*, of this graph, for fixed q, is $\Theta(n^2/\log_q(n))$ [3]. Then, from the general lower bound on mixing time $\tau_{mix} > \frac{1}{2}diam$ [12], we get that $\tau_{mix} = \Omega(n^2/\log_q(n)))$.

Proposition 30. *The transition matrix of M, P, is symmetric.*

Proof. We show that $P_{A,B} = P_{B,A}$, for all $A, B \in GL(n,q)$.

- If $P_{A,B} = 0$, then $P_{B,A} = 0$. By contradiction: If $P_{B,A} \neq 0$, then there is a general elementary matrix C such that $B = C \cdot A$. But then, $A = C^{-1} \cdot B$ and from Proposition 9, C^{-1} is a general elementary matrix. So, $P_{A,B} \neq 0$, which is a contradiction.

- If $P_{A,B} \neq 0$, then there is a unique general elementary matrix C such that $B = C \cdot A$. Again from Proposition 9, C^{-1} is a general elementary matrix. The uniqueness comes from the fact that A is invertible, so $C = B \cdot A^{-1}$.

  Since at each step of the Markov chain we pick a random general elementary matrix to multiple the current state and there is only one such matrix that moves state A to state B and conversely, $P_{A,B} = P_{B,A}$.

□

Proof. (Claim) Since P is a transition matrix, the sum of each row is 1 and because of the symmetry also the sum of each column is 1. So, P is doubly stochastic and from Theorem 16 its stationary distribution is the uniform over $GL(n,q)$. Also, from Proposition 8 there exists a t such that $P^t_{A,B} > 0$, for all $A, B \in GL(n,q)$ and $P_{A,A} \neq 0$. Therefore, M is irreducible and aperiodic and from Theorem 22 the uniform distribution over $GL(n,q)$ is its unique stationary distribution. □

Ideally, we would like to show that after $c \cdot n\ln(n)$, or even $c \cdot n\ln(n)^2$, steps, for a not too large constant c, the resulting matrix is indistinguishable from random for any polynomial time algorithm. Then, even for relatively small n's, $c \cdot n\ln(n)$ or $c \cdot n\ln(n)^2$ is substantially smaller than the number of operations needed to perform matrix-vector multiplication even using the fastest known algorithm. So, the protocol could be practical.

However, this task seems to be really difficult. So, we start by giving evidence that some specific adversaries cannot succeed. Of course, it is not obvious that even if the adversary knows that a matrix is trapdoored, he can use that information to create an attack on the protocol. But, for the rest of the chapter we try to argue that an adversary cannot even distinguish a trapdoored matrix from a random invertible matrix.

3.3.2 One-column case

We analyze the mixing time of the Markov chain, M', that is defined as $(X_t \cdot v_i)_t$, where X_t is the t-th state of M and v_i is the i-th basis vector. Similarly, to the Definition 6, we define an elementary operation in the one-column case as the resulting column when we substitute coordinate x_i by $a \cdot x_i + b \cdot x_j$, $a \neq 0$. This Markov chain is a projection of the original one to a single column. So, its analysis gives us some idea about the analysis of the original chain M.

The state space of M' is all the non-zero n-vectors with elements in $GF(q)$. As before the transition matrix is symmetric and the Markov chain is finite and irreducible, so its unique stationary distribution is the uniform over its state space.

It is not difficult to see that τ_{mix} of M' is $\Omega(n\ln(n))$. In each step, we pick an i, which shows the element that gets updated. If an index in $\{1, ..., n\}$ is not chosen in any of the $L(n)$ steps as i, then the corresponding element would be 0. This is a coupon collector's problem [6], in which we have n "coupons" and each one can be collected equally likely with replacement. Therefore, the number of steps needed is $\Omega(n \cdot \ln(n))$.

Assuming no cancellations

In the Markov chain M', it is possible that the new value of a coordinate is 0 when the previous value was non-zero. We start by analyzing what happens if there are no such cancellations. This is practically the case when the underlying finite field has large order q.

Basically, we define the auxiliary Markov Chain M" with state space $\Omega' = \{0,1\}^n \setminus \{0^n\}$. In this chain, 0 corresponds to a zero coordinate of the original chain and 1 corresponds to an updated coordinate. This chain is an absorbing Markov chain and the vector 1^n is the only absorbing state (Def. 17).

From Proposition 18 we know that the stationary distribution is

$$\pi(x) = \begin{cases} 1 & \text{if } x = 1^n \\ 0 & \text{otherwise} \end{cases}$$

Let us define the random variables T_i for all $i : 1 \leq i \leq n-1$ as the number of steps needed to visit for the first time a state with $i+1$ 1's, given that the starting state has i 1's. These random variables follow a geometric distribution with $p_i = \frac{n-i}{n} \cdot \frac{i}{n-1}$.

Then, from Theorem 27

$$\Delta(t) < P(\sum_{i=1}^{n-1} T_i > t)$$

because $T = \sum_{i=1}^{n-1} T_i$ is a strong stationary time.

In the book of Motwani and Raghavan [15], it appears as an exercise to find the tail inequality of the sum of independent geometric distributions with different parameters:

Theorem 31. *Let $X_1, X_2, \dots, X_n$ be independent geometrically distributed random variables with parameters p_i, for $1 \leq i \leq n$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E[X] = \sum_{i=1}^{n} 1/p_i$, $p = \min_i p_i$, and $\delta > 0$,*

$$P(X > (1+\delta)\mu) < \left(e^p(1-p/2)^{(1+\delta)}\right)^{\mu}.$$

The proof given here is almost identical to the proof of Theorem 4.1 [15].

Proof. For any positive t

$$P(X > (1+\delta)\mu) = P(e^{tX} > e^{t(1+\delta)\mu})$$

Applying the Markov inequality to the right-hand side, we have

$$P(X > (1+\delta)\mu) < \frac{E[e^{tX}]}{e^{t(1+\delta)\mu}}$$

We observe that

$$E[e^{tX}] = E[e^{t\sum_{i=1}^{n} X_i}] = E[\prod_{i=1}^{n} e^{tX_i}]$$

Since the X_i are independent, the random variables e^{tX_i} are also independent. So,

$$E[\prod_{i=1}^{n} e^{tX_i}] = \prod_{i=1}^{n} E[e^{tX_i}]$$

Then, we compute $E[e^{tX_i}]$ for $t < \ln\left(\frac{1}{1-p_i}\right)$

$$\begin{aligned} E[e^{tX_i}] &= \sum_{k=1}^{\infty} P(X_i = k)e^{t\cdot k} = \sum_{k=1}^{\infty}(1-p_i)^{k-1}p_i e^{t\cdot k} \\ &= p_i e^t \sum_{k=0}^{\infty}((1-p_i)e^t)^k = \frac{p_i e^t}{1-(1-p_i)e^t} \\ &= \left(1 - \frac{1-e^{-t}}{p_i}\right)^{-1} \end{aligned}$$

From the above observations we have that for $t < \ln\left(\frac{1}{1-p/2}\right)$

$$\begin{aligned}
P(X > (1+\delta)\mu) &< \frac{\prod_{i=1}^{n}\left(1 - \frac{1-e^{-t}}{p_i}\right)^{-1}}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n}\left(\exp(-\ln(1 - \frac{1-e^{-t}}{p_i}))\right)}{e^{t(1+\delta)\mu}} \\
&= \frac{\exp(-\sum_{i=1}^{n}\ln(1 - \frac{1-e^{-t}}{p_i}))}{e^{t(1+\delta)\mu}} < \frac{\exp(2(1-e^{-t})\sum_{i=1}^{n}\frac{1}{p_i})}{e^{t(1+\delta)\mu}} \\
&< \frac{e^{2(1-e^{-t})\mu}}{e^{t(1+\delta)\mu}}
\end{aligned}$$

In the above we have used the fact that for $x \in (0, 1/2)$, $\ln(1-x) > -2x$. So,

$$-\ln(1 - \frac{1-e^{-t}}{p_i}) < 2\left(\frac{1-e^{-t}}{p_i}\right)$$

Now, we substitute t by $\ln\left(\frac{1}{1-p/2}\right)$ which yields

$$P(X > (1+\delta)\mu) < \left(e^{p}(1-p/2)^{(1+\delta)}\right)^{\mu}$$

□

In our case,

$$\begin{aligned}
\mu &= \sum_{i=1}^{n-1} 1/p_i = \sum_{i=1}^{n-1} \frac{n\cdot(n-1)}{i\cdot(n-i)} \\
&= \sum_{i=1}^{n-1}\left(\frac{n-1}{i} + \frac{n-1}{n-i}\right) = 2(n-1)\sum_{i=1}^{n-1}\frac{1}{i} \\
&= \Theta(n\ln(n))
\end{aligned}$$

because

$$\sum_{i=1}^{n-1}\frac{1}{n-i} = \sum_{i=1}^{n-1}\frac{1}{i}$$

Applying Theorem 31 for $\delta = 1$ in our case gives:

$$P(T > 2\cdot n\ln(n)) < \left(\frac{e}{4}\right)^{\Theta(n\ln(n))}$$

because

$$e^p(1-p/2)^2 = e^{1/n}(1-1/2n)^2 < e(1-1/2)^2 < e/4$$

Therefore, from theorem 27, $\tau_{mix} = O(n\ln(n))$.

Actual chain

The previous method gives us an understanding of what the mixing time should be when $q \to \infty$, since as q gets large the probability of cancellation goes to zero. However, when q is small, the random variables T_i's do not follow a geometric distribution. It is possible that the starting state has i non-zero elements, but now the Markov Chain can visit states with less than i non-zeros, before reaching a state with $i+1$ non-zero elements. This is because in this case the probability of cancellation is greater than zero.

So, we run some simulations with different n's and q's in order to estimate the expected number of steps needed until all elements of the column are updated.

The pseudocode for the test and the results of the simulations for different q's follow.

Algorithm 1 One-column case

```
1: $B \leftarrow \{1,2,\dots,n\}$
2: $count \leftarrow 1$
3: $x \leftarrow$ random basis vector
4: $top$:
5: **if** $B = \emptyset$ **then return** $count$
6: pick a random general row operation $(i,j,a,b)$
7: $x_i \leftarrow a \cdot x_i + b \cdot x_j$
8: $count++$
9: **if** $x_j \neq 0$ **then** $B \leftarrow B \setminus \{i\}$
10: **goto** $top$.
```

We conjecture the following:

Conjecture 1. *After $t = O(n\ln(n))$ steps, $||P'^t - \pi|| < 2^{-n}$, where P' is the transition table and π' is the stationary distribution of M'.*

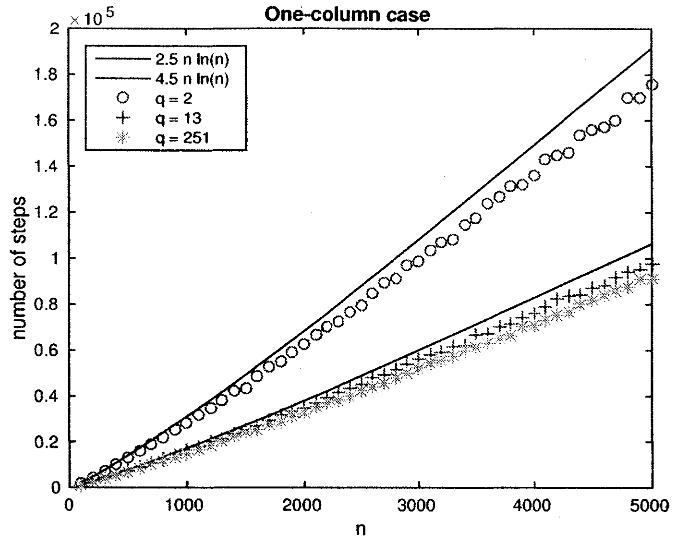

Figure 3-1: Results of the simulation for the one-column case. The x-axis corresponds to different values of n, while the y-axis shows the value of the variable *count*, namely the number of steps needed before the Algorithm 1 terminates.

This conjecture implies that the mixing time of the Markov chain M' is $O(n \ln(n))$. As show in the figure, the constant c seems to depend on q: smaller q's (e.g. $q = 2$) delay the mixing of the chain, because of the bigger probability of cancellation. Nevertheless, even for medium q's ($q = 13$), the constant c seems to be small enough for our purposes.

3.3.3 One-row case

Similarly to the one-column case, we define M" as the projection of M to one row. We run simulations for the one-row case and count the expected number of steps needed for a single row of the matrix to become uniformly random over $GF(q)^n \setminus \{0^n\}$.

The pseudocode for this test is shown in 2.

The results of the above simulation for different q's are show in Figure 3-2:

Again, we conjecture that:

Conjecture 2. *After* $t = O(n \ln(n))$ *steps,* $||P''^t - \pi|| < 2^{-n}$*, where* P'' *is the transition table and* π'' *is the stationary distribution of M".*

Algorithm 2 One-row case

```
1: $B \leftarrow \{1, 2, \ldots, n\}$
2: $count \leftarrow 1$
3: $A \leftarrow I_n$
4: $row \leftarrow$ a random element in $\{1, 2, \ldots, n\}$
5: $top$:
6: **if** $B = \emptyset$ **then return** $count$
7: pick a random general row operation $(i, j, a, b)$
8: $A_i \leftarrow a \cdot A_i + b \cdot A_j$
9: $count++$
10: **if** $i = row$ **then**
11:     $B \leftarrow B \setminus \{k\}, \forall k$ such that $A(j, k) \neq 0$
12: **goto** $top$.
```

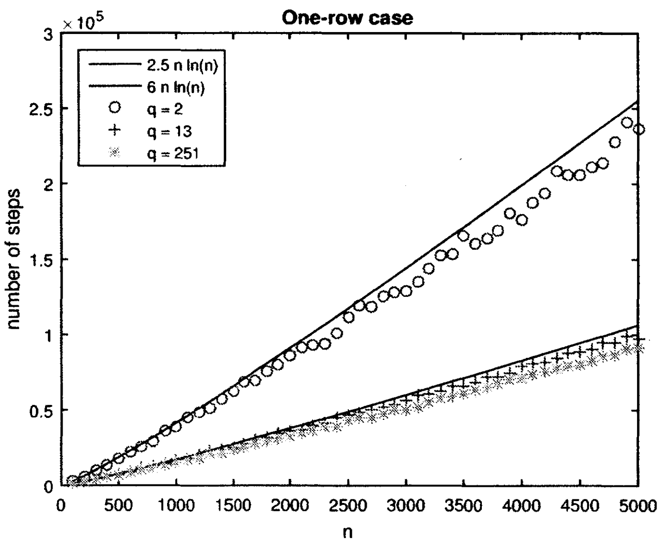

Figure 3-2: Results of the simulation for the one-row case. The x-axis corresponds to different values of n, while the y-axis shows the value of the variable *count*, namely the number of steps needed before the Algorithm 2 terminates.

Therefore, the mixing time of M" is of the form $c \cdot n \ln(n)$. However, as show in the figure, in this case c seems to be larger than in the one-column projection when q is small. This is again connected to the higher probability of cancellation.

3.3.4 Expected number of elements equal to $x \in GF(q)$

Another test an adversary can try is to count the number of elements equal to $x \in GF(q)$ in the public matrix A and compare it with the expected number of such x's in a random invertible matrix. If the two quantities differ significantly, then the adversary knows that the matrix is trapdoored.

Let us define by $N_{rand}^{(x)}$ and $N_{trap}^{(x)}$ the random variable that shows the number of elements equal to x in a random invertible matrix and trapdoored matrix respectively.

Computing $E[N_{rand}^{(x)}]$

If A is a random invertible matrix over $GF(q)$, then by linearity of expectation:

$$E[N_{rand}^{(x)}] = E[\sum_{1 \leq i,j \leq n} \mathbb{1}_{A_{i,j}=x}] = \sum_{1 \leq i,j \leq n} E[\mathbb{1}_{A_{i,j}=x}]$$

Each element of A has the same probability of being x. We denote this probability by p_x. Then, $E[\mathbb{1}_{A_{i,j}=x}] = p_x$.

Since A is an invertible matrix, p_x, is the same for every $x \neq 0$. So, if we first compute p_0, then we can find p_x from the equation:

$$p_x = \frac{1 - p_0}{q - 1}, \forall x \in GF(q) \setminus \{0\}$$

It is well known that the number of invertible matrices with elements in $GF(q)$ is equal to $(q^n - 1) \cdot (q^n - q) \cdot \ldots \cdot (q^n - q^{n-1})$. Similarly, we can count the number of matrices in $GL(n, q)$ with the element in position (i, j) equal to 0.

Proposition 32. *There are $(q^{n-1} - 1) \cdot (q^n - q) \cdot \ldots \cdot (q^n - q^{n-1})$ matrices in $GL(n, q)$ with element in position (i, j) equal to 0.*

Proof. Without lost of generality, we can assume that $i = 1$. Let $A \in GL(n, q)$ with $A_{i,j} = 0$, then there are $(q^{n-1} - 1)$ choices for the $n - 1$ elements of the first row, since we exclude the all-zero row. Then, there are $q^n - q$ choices for the second row, because the only limitation is that the second row should not belong to the subspace produces by the first row. Similarly, we can count the choices for all subsequent rows. □

Therefore,

$$p_0 = \frac{(q^{n-1}-1)\cdot(q^n-q)\cdot\ldots\cdot(q^n-q^{n-1})}{(q^n-1)\cdot(q^n-q)\cdot\ldots\cdot(q^n-q^{n-1})} = \frac{q^{n-1}-1}{q^n-1}$$

and

$$p_x = \frac{1-p_0}{q-1} = \frac{1-\frac{q^{n-1}-1}{q^n-1}}{q-1} = \frac{q^{n-1}}{q^n-1}, \forall x \in GF(q)\setminus\{0\}$$

Computing $E[N^{(x)}_{trap}]$

Obviously, this expected value depends on the number of steps we have used to create the trapdoor. If this number is large enough, then by the analysis of the previous sections, $E[N^{(x)}_{trap}] \approx E[N^{(x)}_{rand}]$.

From the other presented test, we know that we need at least $\Omega(n\ln(n))$ steps. Therefore, we try to compute the value of $E[N^{(x)}_{trap}]$ when we have used $\Theta(n\ln(n))$ steps to create the trapdoor.

Again, by linearity of expectation

$$E[N^{(x)}_{trap}] = E[\sum_{1\le i,j\le n} \mathbb{1}_{A_{i,j}=x}] = \sum_{1\le i,j\le n} E[\mathbb{1}_{A_{i,j}=x}] = \sum_{1\le i,j\le n} q^{(x)}_{i,j}$$

where A is a trapdoor matrix with trapdoor size $L = \Theta(n\ln(n))$ and $q^{(x)}_{i,j}$ is the probability that $A_{i,j} = x$.

Also,

$$\begin{aligned}
|q^{(x)}_{i,j} - p_x| &= |\sum_{\substack{v\in GF(q)^n\setminus\{0^n\}\\ \text{s.t.} v(i)=x}} (P'^t(v_j,v) - \pi'(v))| \\
&\le \sum_{\substack{v\in GF(q)^n\setminus\{0^n\}\\ \text{s.t.} v(i)=x}} |P'^t(v_j,v) - \pi'(v)| \\
&\le \sum_{v\in GF(q)^n\setminus\{0^n\}} |P'^t(v_j,v) - \pi'(v)| \\
&= 2||P'^t(v_j,\cdot) - \pi'||
\end{aligned}$$

where v_j is the j-th basis vector and π' is the stationary distribution of M' (the uniform over $GF(q)^n \setminus \{0^n\}$) and P' its transition matrix.

Therefore, from Conjecture 1:

$$|q_{i,j}^{(x)} - p_x| \leq 2^{1-n}$$

and

$$|E[N_{rand}^{(x)}] - E[N_{trap}^{(x)}]| < \frac{n^2}{2^{n-1}}$$

3.4 Other Trapdoor Constructions

The above candidate construction is not the only one that can be used in the protocol. It is possible that picking the rows i and j in each step randomly delays the mixing of the Markov chain. There are examples of Markov chain where we see this behavior. For instance, in the case of random walks on upper triangular matrices, the mixing time is $O(n^2 \log(n))$ when the generator set is $\{A : A$ elementary matrix corresponding to row operation $A_i \leftarrow A_i + b \cdot A_j$, where $i < j$ and $b \in GF(q)\}$ [17], but it is $O(n^2 \log(q))$ when the generator set is $\{A : A$ elementary matrix corresponding to row operation $A_i \leftarrow A_i + b \cdot A_{i-1}$, where $b \in GF(q)\}$ [18].

So, another possible construction could be the following:

For $k \in \{1, \ldots, L(n)\}$:

- Pick j_k uniformly at random from $\{1, \ldots, n\} \setminus \{(k-1) \bmod n, k \bmod n\}$
- Pick a_k uniformly at random from $GF(q) \setminus \{0\}$
- Pick b_k uniformly at random from $GF(q)$
- Pick c_k uniformly at random from $GF(q)$

Then,

$$A = A^{(L)} \cdot A^{(L-1)} \cdot \ldots A^{(1)}$$

where $A^{(k)}$ is the corresponding to the operation $A_{k \bmod n} \leftarrow a_k \cdot A_{(k \bmod n)} + b_k \cdot A_{(k-1 \bmod n)} + c_k \cdot A_j$. Each $A^{(k)}$ can be written as a product of two general elementary matrices.

Thus, the trapdoor $T(A)$ consists of the sequence of $L(n)$ tuples:

$$(j_k, a_k, b_k, c_k), \text{ for } k = 1, 2, ..., L$$

The program, that the honest party uses, takes advantage of the knowledge of the trapdoor and is similar to the one presented previously.

We notice that in the case of very large q, $2n$ steps are enough for the mixing of the one-column case. However, similar experiments to the random case show that this might not be true for small q's. Therefore, since we have no additional theoretical analysis for this construction, we focus our attention to the previous one.

Chapter 4

Implementation in practice

In this chapter, we consider the practicality of the protocol. We give the basic characteristics of a smart card and based on them we specify the values that the protocol parameters can get.

4.1 Characteristics of smart cards

In order to judge the implementability of our protocol, we provide some of the technical characteristic of a specific smart card [16].

The P60D080 and P60D144 devices are members of the new SmartMX2 Family produced by NXP. Their basic features:

- EEPROM: choice of 80 KB or 144 KB
- ROM: 384KB
- RAM: 8.125 KB (8320 B)
  - 5632 B CXRAM (including 256 B IRAM) usable for CPU
  - 2688 B FXRAM usable for Fame2 or CPU
- SmartMX2 CPU: orthogonal instruction set offering 32-/24-/16-/8-bit instructions optimized for secured and low power smart card applications

- ISO/IEC 7816 contact interface (UART) and ISO/IEC 14443A Contactless Interface Unit (CIU)
  - ISO/IEC 7816 contact interface (UART) offering hardware support for ISO/IEC 7816 $T = 0$ and $T = 1$ protocol stack implementation
  - ISO/IEC 14443A Contactless Interface Unit (CIU) supporting data rates of 106 kbit/s, 212 kbit/s, 424 kbit/s, 848 kbit/s and offering hardware support for ISO/IEC 14443 $T = CL$ protocol stack implementation

4.2 Suggestions on the protocol parameters

We suggest that we need at least one order of magnitude difference on the time the prover, with her trapdoor, needs to spend to perform the computation and the time any other adversary, without the trapdoor, would need. Simultaneously, we need to be sure that n is not too large to be practical. The use of this protocol in smart cards sets also constrains on the parameters. For example, smart cards have limited space, so the trapdoor must be designed so that it offers a significant advantage to the prover, even in the case that she can use limited space to perform the computation.

We have implemented the naïve algorithm for matrix-vector multiplication and the algorithm that uses the trapdoor information in Matlab in order to estimate the size of the parameters of the protocol. We have tested both the constructions of the previous chapter, but since the results are very similar, we present only those of the first construction. In this construction, the trapdoor is a sequence of random elementary matrices.

The parameters of our protocol are:

- q, the size of the finite field, $GF(q)$, over which all operations are done. In the implementation, we try three different values of q, a small one (2), a medium (13) and a sufficiently large (251). Smarts cards have limited space, so we limit q so that each element is at most one byte.
- n, which gives the size of the public key and the challenge. The public key is

a $n \times n$ matrix over $GF(q)$ and the challenge is an n-vector over $GF(q)$. Also, the size of the trapdoor depends on n.

- $L(n)$, the size of the trapdoor. We consider two cases either $L = O(n \ln(n))$ or $L = O(n \ln(n)^2)$. From the previous tests, we know that we need $L(n)$ at least $\Omega(n \ln(n))$. But, these are just specific tests that provide some guidance for the size of the trapdoor. So, our choices shown at the next figures are $L = 2.5n \ln(n)$ for $q \in \{13, 251\}$, $L = 6n \ln(n)$ for $q = 2$ and $L = n \ln(n)^2$ for $q \in \{2, 13, 251\}$.

The Table 4.1 shows the ratio of average time of matrix-vector multiplication over different $GF(q)$ ($q \in \{2, 13, 251\}$) using the naïve algorithm over the average time of finding the correct output using the trapdoor over the same finite fields for different n's. We repeat the process for the different $L(n)$'s mentioned above.

From the figures, we can see that for all the different q's and $L(n)$'s, $n = 1000$ *is enough to get a 10-time advantage by using the trapdoor compared to using the basic algorithm.*

It is very probable that the algorithm that we used for the normal matrix-vector multiplication can be further optimized or that the adversary can try to impersonate an honest user with a fake smart card that is more powerful than the authentic card that is issued by the trusted party (e.g. by the bank). However, we believe that an advantage of 18-times, which corresponds for example to $q = 13$, $n = 1600$ and $L(n) = n \ln(n)^2 \approx 87090$, should be enough so that an adversary cannot succeed. We note that this suggestion is based on the assumption that no polynomial adversary can distinguish a random invertible matrix from a trapdoored one with trapdoor size $L(n) = n \ln(n)^2$. If this assumption does not hold for this $L(n)$, then we should adjust the parameters accordingly.

For these parameters, the storage that is needed for the trapdoor is approximately $170KB$ and of the challenge is $800B$. These numbers seem to fit in a smart card, according to the specifications we provided above. In addition, by using pseudorandomness to generate the trapdoor information, the required storage might become even smaller. In this case, the trapdoor is constructed using the output of a pseudo-

n	L(n) = 6 n ln(n)	L(n) = 2.5 n ln(n)		L(n) = n ln(n))^2		
	q = 2	q = 13	q = 251	q = 2	q = 13	q = 251
100	1.3300	3.1673	3.7086	1.7109	1.6922	2.1278
200	2.6625	6.1402	8.0743	2.9884	2.8715	3.8218
300	3.4758	9.2257	9.8954	3.7545	4.0250	4.5732
400	4.3861	13.0689	11.9358	4.3800	5.4569	5.6866
500	6.5349	12.9418	16.2930	6.4234	6.0505	7.3003
600	7.4609	19.0222	21.7165	7.1326	7.9309	8.7560
700	7.5339	21.7697	24.1286	7.1862	8.8563	9.8807
800	8.7573	27.2890	29.3588	8.1600	10.6391	11.4942
900	9.5655	28.0601	26.4763	8.6299	10.9658	11.2429
1000	11.8088	32.5718	27.9198	10.4294	12.2709	11.6950
1100	13.5450	35.0642	37.0816	11.8015	12.9359	13.6667
1200	13.9060	38.3508	38.6044	12.0000	13.7471	14.0317
1300	15.6534	41.7347	40.2852	13.3420	14.8411	15.1297
1400	14.9496	43.9892	44.8841	12.7402	15.6790	15.8871
1500	17.5381	47.4801	49.7784	14.7710	16.5414	17.4910
1600	18.7442	55.7037	52.2541	15.5242	18.9429	18.1416
1700	18.6975	51.5867	56.3132	15.6737	18.0399	19.6400
1800	20.1533	52.8366	65.2348	16.6585	18.0420	21.7798
1900	21.4612	56.2787	64.8387	17.5495	19.1803	22.3495
2000	24.3148	61.9339	66.4707	19.3869	20.8125	22.6190

Table 4.1: Ratio of the timing results using the naïve algorithm and the trapdoor. The difference shown for $q = 2$ and $q \in \{13, 251\}$ for $L = cn\ln(n)$ is because of the different constants c used.

random generator. The honest user stores only a small random seed and every time she needs the trapdoor, she runs the pseudorandom generator with this stored seed and reconstructs the trapdoor.

Chapter 5

Conclusion

Authentication is an important well-studied problem. In this thesis, we investigate the idea of public-key authentication via smart cards. Smart cards are often used today and they offer a useful tool in authentication. However, their limited capabilities in storage and computing power make some of the usual authentication protocols impractical.

In the past, public-key authentication was based on computing an answer to a challenge that no one without the secret key should be able to efficiently compute. In our case, the scenario is different. Everyone knowing the public key can compute the correct answer, but the secret key provides a faster way to complete this task.

We give specific constructions where the public key is an $n \times n$ matrix over $GF(q)$ and the challenge is to quickly compute a matrix-vector multiplication with a random n-vector. After describing the construction and giving evidence that theoretically this construction could be secure, we give an elementary implementation and we try to give specific suggestions for the parameters of the protocol. These first experimental results seem optimistic, since we can find many tuples of parameters that both fit in the memory of a smart card and provide the necessary level of security.

Further directions of this research include the theoretical proof that the proposed construction is secure, at least asymptotically. The proof of the conjectures we have pointed in the text would be a first step in this direction. If these conjectures are proved true, the theoretical support for this method becomes strong. Also, more experiments on tests that an adversary could perform would be useful in order to acquire more confidence in the security of the protocol or perhaps to better adjust the protocol's parameters (e.g. the size of the trapdoor). The proposed scheme seems invulnerable to any attacks we can think of, and may be suitable for practical use, although further study is warranted before any large-scale deployment should be attempted.

Bibliography

[1] Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.

[2] David Aldous and James Allen Fill. *Reversible Markov Chains and Random Walks on Graphs.* 2002. http://www.stat.berkeley.edu/ aldous/RWG/book.html.

[3] Daniel AndrÃĺn, Lars HellstrÃűm, and Klas MarkstrÃűm. On the complexity of matrix reduction over finite fields. *Advances in Applied Mathematics*, 39(4):428 – 452, 2007.

[4] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradžev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics—Doklady*, 11(5):1209–1210, 1970.

[5] Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '93, pages 344–359. Springer-Verlag New York, Inc., 1994.

[6] Paul Erdös and Alfréd Rényi. On a classical problem of probability theory. *MTA Mat. Kut. Int. Közl*, 6A, 1961.

[7] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[8] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology*, volume 263 of *CRYPTO '86*, pages 186–194. Springer, 1986.

[9] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[10] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 21–30, New York, NY, USA, 2015. ACM.

[11] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proceedings on Advances in Cryptology*, volume 1109 of *CRYPTO '96*, pages 104–113. Springer Berlin Heidelberg, 1996.

[12] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[13] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[14] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[15] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[16] NXP. smartmx2_p60, 2010. http://www.nxp.com/products/identification_and_security/smart_card_ics/smartmx2_p60/P60D080PX30.html.

[17] Igor Pak. Two random walks on upper triangular matrices. *Journal of Theoretical Probability*, 13(4):1083–1100, 2000.

[18] Yuval Peres and Allan Sly. Mixing of the upper triangular matrix walk. *Probability Theory and Related Fields*, 156(3-4):581–591, 2013.

[19] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.

[20] Claus P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings on Advances in Cryptology*, CRYPTO '89, pages 239–252. Springer-Verlag New York, Inc., 1989.

[21] Ryan Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *SODA*, pages 1–11. ACM Press, 2007.

[22] Shmuel Winograd. On the number of multiplications necessary to compute certain functions. 58(5):1840–1842, 1967.