Secure Smartcard-Based Fingerprint Authentication *

T. Charles Clancy†
Computer Science
University of Maryland, College Park
tcc@umd.edu

Negar Kiyavash, Dennis J. Lin
Electrical and Computer Engineering
University of Illinois, Urbana-Champaign
{kiyavash, djlin}@uiuc.edu

ABSTRACT

In this paper, the fundamental insecurities hampering a scalable, wide-spread deployment of biometric authentication are examined, and a cryptosystem capable of using ngerprint data as its key is presented. For our application, we focus on situations where a private key stored on a smartcard is used for authentication in a networked environment, and we assume an attacker can launch o -line attacks against a stolen card.

Juels and Sudan's *fuzzy vault* is used as a starting point for building and analyzing a secure authentication scheme using ngerprints and smartcards called a *fingerprint vault*. Fingerprint minutiae coordinates m_i are encoded as elements in a nite eld F and the secret key is encoded in a polynomial $f(x)$ over $F[x]$. The polynomial is evaluated at the minutiae locations, and the pairs $(m_i, f(m_i))$ are stored along with random (c_i, d_i) cha points such that $d_i \neq f(c_i)$. Given a matching ngerprint, a valid user can seperate out enough true points from the cha points to reconstruct $f(x)$, and hence the original secret key.

The parameters of the vault are selected such that the attacker's vault unlocking complexity is maximized, subject to zero unlocking complexity with a matching ngerprint and a reasonable amount of error. For a feature location measurement variance of 9 pixels, the optimal vault is 2^{69} times more di cult to unlock for an attacker compared to a user posessing a matching ngerprint, along with approximately a 30% chance of unlocking failure.

*Supported by The Boeing Company under Illinois Technology Challenge Grant 03-115. Any opinions, ndings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily re ect the views of The Boeing Company or the State of Illinois.

†This work was completed while T. Clancy was with the Electrical and Computer Engineering department at the University of Illinois, Urbana-Champaign.

WBMA'03, November 8, 2003, Berkeley, California, USA.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption

Keywords

biometrics, ngerprint, smartcard, authentication

1. INTRODUCTION

In our increasingly electronic environment, everything is becoming a network, from the computer terminals we sit at, to the supermarket checkout, and even the locks on our doors. In these abstract networks, all forms of authorization and access control require networks to have a secure method of authenticating users.

Smartcards o er a new paradigm for authentication. Now, users' private keys are stored on smartcards. These users can prove their identity by using the card to provide a correctly signed message to an authentication server. Relying on the security of public-key authentication, the new task is to protect the private key on the smartcard itself. We believe that biometric authentication, in particular ngerprints, is a practical method of providing this protection.

There are two main approaches for using biometric information. The rst is ngerprint matching, where the smartcard stores a template of the user's ngerprint and requires the user to present a matching template before it will sign messages on the user's behalf. The second method is ngerprint *mapping*, where a ngerprint is used to obscure the private key, without storing a template. The private key can only be recovered and consequently used to sign an authentication message if a valid ngerprint is provided.

From a fundamental security standpoint, the key distinction between the two techniques is whether a physical attack on the smartcard could yield any useful information. If the private key and biometric template are both stored unencrypted on the smartcard, they would both be susceptible to a physical attack. The goal here is to provide guaranteed security, rather than the illusion of security.

This paper focuses on using a modi ed version of Juels and Sudan's fuzzy vault [**?**] termed the *fingerprint vault* to encrypt the private key on the smartcard using ngerprint information, and derives theoretic bounds on its security. By analyzing the techniques used to perform the authentication and the entropy of the biometric data used, the complexity of various attacks can be quanti ed. In all cases, there is a trade-o between reproducibility and security.

The overall goals of this work are as follows:

- de ne the ngerprint vault scheme and its associated algorithms;

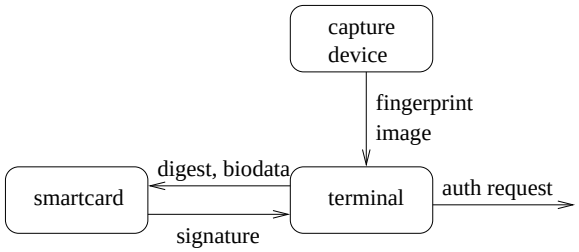

Figure 1: Authentication Diagram

- present probabilistic and systematic bounds on some of the vault parameters;
- de ne a class of unlocking techniques that can be used for both attacks and by legitimate users; and
- nd the optimal fuzzy vault parameter choice to maximize attackers' complexity and minimize users' complexity, while ensuring reasonable reproducibility

The remaining sections are organized as follows. Section two discusses background information and motivates the research. Section three outlines prior similar research e orts. Section four presents the ngerprint vault and analyzes its applicability from a probabilistic sense. Section ve analyzes the security of the ngerprint vault from a complexity-theoretic sense. Section six describes empirical results stemming from sample ngerprint data. Section seven concludes. Proofs are provided in the appendix of the extended version found on the authors' website.

2. BACKGROUND AND MOTIVATION

An airport is a motivating example where a ngerprint and smartcard solution is ideal. Passengers each have smartcards storing their personal information, including a private key. In order to access the information on their smartcard at airline terminals, they must provide the smartcard with valid ngerprint data.

First, we present a simpli ed, generic protocol for use with smartcard-based biometric authentication. It is similar to those found in many popular public-key authentication schemes, such as SESAME [**?**, **?**] and SSH [**?**].

In general, a server should believe a user is who they say they are if they can provide a signed message containing their identity and a nonce or challenge (i.e. random number) selected by the server. If the public key associated with the speci ed identity correctly veri es the signature, only the valid user could have sent it. If a certi cate is not provided by the user, the server will need a database of public keys.

From the client standpoint, signing ability is required to authenticate. Figure **??** demonstrates how this can be done with biometric data. First, a ngerprint image is captured from a scanner. This data is sent to a terminal which translates it into some smaller numeric representation, or template. Both the message digest we wish to sign and the biometric template are sent to the smartcard. Provided the biometric data is valid, the smartcard will use its internally stored private key to generate and return a signature for the message digest.

Our analysis shall focus on the methods by which smartcards use biometric data to sign a message. This assumes the worst case scenario: a smartcard has been stolen, and an attacker with complete physical access is attempting to retrieve the private key.

Given physical access, there are two main classes of physical attacks against smartcards: noninvasive or side-channel attacks, and invasive attacks.

Three of the most popular noninvasive attacks are power analysis [**?**], timing analysis [**?**], and electromagnetic (EM) analysis [**?**]. All of these attacks can be used to determine sensitive information on a smartcard; however, noninvasive attacks can generally be thwarted by clever algorithms, data obfuscation, and shielding techniques.

Invasive attacks [**?**, **?**] generally involve dissolving the chip packaging and reverse engineering the processor itself. Given complete physical access, it is impossible to prevent an attacker from retrieving data stored in memory. The only way to protect against such an attack is to encrypt the contents of memory using a key not stored on the card itself. Then, an attacker may retrieve the data from the card, but it will be of no use.

Protecting the smartcard now requires e orts on two fronts. First, the smartcard processor itself must be designed such that it is immune to the various on-line side-channel attacks. Secondly, without a matching ngerprint, an attacker should not be able to obtain any sensitive information from the card. In particular, users' private keys must be stored in an encrypted format. This second front is the focus of the work presented here.

3. PAST WORK

In the past few years, there have been several research e orts aimed at addressing the intersections between cryptography and biometrics. Here we address biometric cryptosystems in general, delve more deeply in to the fuzzy vault, and then brie y examine various polynomial reconstruction techniques.

3.1 Biometric Cryptosystems

In 1998, Davida, et al. [**?**], were among the rst to suggest o -line biometric authentication. It moved biometric data from a central server into a signed form on a portable storage device, such as a smartcard. Their system was essentially a PKI-like environment that did local ngerprint matching. Its main aw is that it required some local authentication authority to have a key capable of decrypting the template stored on the storage device. While they address the key management issues, the basic premise is still that of local ngerprint matching, and is therefore inherently insecure.

The next year there were three innovative, yet similar methods that did not perform biometric matching. The rst is the fuzzy commitment scheme [**?**]. Here, a secret (presumably a private key used for later authentication) is encoded using a standard error correcting code such as Hamming or Reed-Solomon, and then XOR-ed it with a biometric template. To retrieve the secret, a slightly di erent biometric template can again be XOR-ed, and the result put through an error correcting decoder. Some small number of bit errors introduced in the key can be corrected through the decoding process. The major aw of this system is that biometric data is often subject to reordering and erasures, which cannot be handled using this simple scheme.

In [**?**], a technique was proposed using the phase information of a Fourier transform of the ngerprint image. The ngerprint information and a randomly chosen key are mixed

together to make it impossible to recover one without the other. In order to tolerate errors, the system used a l- ter that minimizes the output variance corresponding to the input images. To provide further redundancy, an encoding process stores each bit multiple times. The work does not address how much these steps reduce the entropy of the original image; thus, it is not clear that there exists a set of parameters which will allow the system to reliably recognize legitimate users while providing a reasonable amount of security.

A third paper [?] has a similar theoretical foundation to this work, but aims toward a completely di erent application. Here, Monrose, et al., attempt to add entropy to users' passwords on a computer system by incorporating data from the way in which they type their password. Since the biometric being used here so so radically di erent from ngerprints, their results are not applicable to this work.

Recently, Juels and Sudan [?] proposed the *fuzzy vault*, a new architecture with applications similar to Juels and Wattenberg's *fuzzy commitment scheme*, but is more compatible with partial and reordered data. The fuzzy vault is used here as a starting point for the biometric scheme presented in this paper.

3.2 Fuzzy Vault

Here, we describe the original fuzzy vault, with some slight notational di erences. As with any cryptosystem, there is some message m that needs to be encrypted, or in this case *locked*. Some symmetric fuzzy key can be used to accomplish this task, and then used again later to decrypt, or *unlock* the original message. Here, our message m is rst encoded as the coe cients of some degree k polynomial in x over a nite eld $\mathbb{F}_q$.

This polynomial $f(x)$ is now the secret to protect. The locking set $\mathcal{L}$ is a set of t values $l_i \in \mathbb{F}_q$ making up the fuzzy encryption key, where $t > k$. The locked vault contains all the pairs $(l_i, f(l_i))$ and some large number of cha points (α_j, β_j), where $f(\alpha_j) \neq \beta_j$. After adding the cha points, the total number of items in the vault is r.

In order to crack this system, an attacker must be able to separate the cha points from the legitimate points in the vault. The di culty of this operation is a function of the number of cha points, among other things. A legitimate user should be able to unlock the vault if they can narrow the search space. In general, to successfully interpolate the polynomial they have an unlocking set $\mathcal{U}$ of t elements such that $\mathcal{L} \bigcap \mathcal{U}$ contains at least $k+1$ elements

To summarize the vault parameters:

- $f(x)$ is a degree k polynomial in $\mathbb{F}_q[x]$
- $t \geq k$ points in $\mathcal{L}$ interpolate through $f(x)$
- $r \geq t$ is total number of points in the vault

This vault shall be referred to as $\mathcal{V}(\mathbb{F}_q, r, t, k)$.

Spurious polynomials of degree k interpolated by t points may show up in the randomly selected cha points. In [?], the authors present a lemma describing the security of their scheme based on the number of these polynomials that exist in a vault with general parameters.

LEMMA 1. *For every $\mu > 0$, with probability $1 \quad \mu$, a vault of size t contains at least $\frac{\mu}{3} q^k \ ^t (r/t)^t$ polynomials $f'(x)$ of degree less than k such that the vault contains exactly t points of the form $(x, f'(x))$.*

3.3 Polynomial Interpolation

In order to actually reconstruct the secret locked within the fuzzy vault, the points in the unlocking set must be used to interpolate a polynomial. The unlocking set will contain both real points and cha points.

The simplest mechanism for recovering the polynomial is a brute-force search, where various $k+1$ element subsets of the unlocking set are used to interpolate a degree k polynomial, using Newtonian Interpolation [?].

A second method is to use a Reed-Solomon decoder [?], as suggested by Juels and Sudan. While RS codes are traditionally used to correct errors in messages transmitted over noisy channels, they are essentially a generalization of the polynomial reconstruction problem. Using a (t, k) code, t points can be fed into the decoder, and the degree k polynomial will be returned.

There are two main RS decoding algorithms: the Berlekamp-Massey algorithm [?], and the Guruswami-Sudan algorithm [?]. Berlekamp-Massey requires $\frac{k+t}{2}$ real points in the unlocking set while Guruswami-Sudan only requires $\sqrt{kt}$. Unfortunately, this extra error correcting capability requires signi cantly more computation. Interestingly enough, the two algorithms provide nearly identical unlocking complexities for a wide range of vault parameters, so we shall focus on the Berlekamp-Massey method.

Another eld called *noisy polynomial interpolation* has had some recent advances, notably by Arora and Khot [?] and Bleichenbacher and Nguyen [?]. However, the results of this work are not applicable to the fuzzy vault. In [?], they examine the problem of nding all polynomials that interpolate through the points $(x_i, [y_i \quad \delta, y_i + \delta])$. The *noise* in our points has been removed by the existence of the fuzzy vault, so this is not useful to us. In [?], they look at the problem of interpolating the points $(x_i, y_{i,1}), (x_i, y_{i,2}), ...$, but since we do not have cha points overlapping with real points, their new algorithm is also not applicable.

4. FINGERPRINT VAULT

In this section, we describe our modi ed *fingerprint vault*, and the algorithms used to lock and unlock data. Additionally, theoretic bounds on our ability to successfully unlock the vault are determined.

4.1 Generalized Fuzzy Vault

First, the fuzzy vault speci es that the size of the locking set, unlocking set, and RS codewords are all the same size. Here, we loosen this restriction.

- t is the number of points in $\mathcal{L}$
- τ is the number of points in $\mathcal{U}$
- n is the RS codeword size

In [?], the authors consider the case where $\mathcal{L}$ and $\mathcal{U}$ are taken from discrete sets, however for biometric purposes these values are not discrete (or are taken from a su ciently high-resolution discrete set). Hence, for all the elements in $\mathcal{U}$, we need to nd the closest elements in vault, and then try unlocking with those values. Consequently, a quantization problem is introduced. How closely can we pack cha points and still maintain a reasonable probability of quantization error?

The key being used to lock our fuzzy vault is pixel coordinate locations, (x_i, y_i), of features on a ngerprint image.

Consequently, and ideal eld for the vault is $\mathbb{F}_{p^2}$, such that p is prime.

Lemma **??** probabilistically gives us the number of spurious polynomials of a particular degree in our vault. The presence of many such polynomials is the key to proof of security in [**?**]. Unfortunately, for reasonable vault parameters (see Section **??**), there exists a δ with $k \quad \delta \quad t$ such that the expected number of spurious degree k polynomials interpolated by more than δ points is less than one. The consequence is that the brute-force search for a degree k polynomial interpolating δ points will yield a unique result, namely, the vault secret.

The precise value of δ such that the results of an unlocking attempt can be veri ed will be required later:

COROLLARY 1. *A value of δ satisfying the above requirement is*

$$\delta \geq \left\lceil \frac{\log \frac{1}{3} p^{2k}}{\log \frac{kp^2}{r}} \right\rceil \tag{1}$$

However, an attacker will still have to search through the space of possible polynomials. We will show in Section **??** that this complexity can still be formidable. First, however, we will de ne our algorithms.

4.2 Feature Extraction

Feature extraction has been the focus of much research in past years, and this paper does not address it in detail. For the tests performed in this paper, the VeriFinger toolkit from Neurotechnologija Ltd. [**?**] was used to extract ngerprint features. The feature extraction process is visually represented in Figure **??**.

In **??**(a), one can see an image scan of a ngerprint. This is received directly from the ngerprint capture device. Various edge detection algorithms are then used to convert that information into **??**(b). This gure features a cleaned-up version of the original scan. From there, features can be readily identi ed, as in **??**(c). Each identi ed are in **??**(c) represents a ngerprint *minutiae*, which is a location where a ngerprint ridge either splits or ends.

Here, we shall consider the feature extraction and alignment as a black box, yielding normalized (x, y) pixel coordinates of ngerprint minutiae. The outputs from the black box for several scans of the same nger are generally close to one another, unless the ngerprint image was severely clipped. The intra-scan variance imposes limitations on our system.

4.3 Feature Noise

Each person's ngerprint consists of a xed set of minutiae locations $m_i = (x_i, y_i) \in \mathcal{M}$. However, due to systematic errors in image capture, processing, and alignment, noise is added to each point, such that our nal points are

$$m_i' = (x_i + n_{xi}, y_i + n_{yi})$$

Additionally, the processing noise may discard features or add additional features that are not present on the actual ngerprint.

The next step in the analysis is to derive a model for the noise introduced by the capture device and extraction algorithms. From this information, we can derive matching error probabilities. To simplify analysis, an additive Gaussian noise model will be assumed.

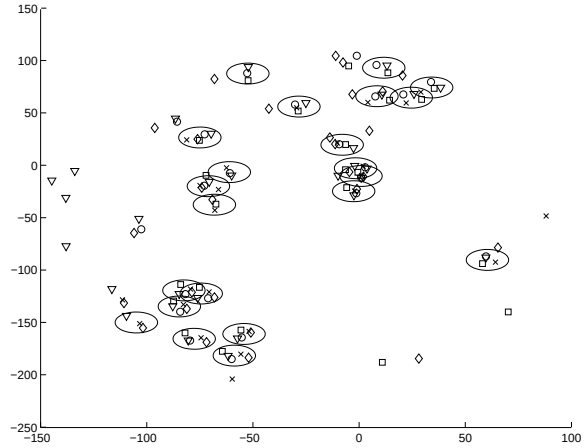

Figure 3: Plot of minutiae from 5 scans of the same person, with reliable regions marked

To estimate the distribution on the minutiae locations, statistical data is needed. To accomplish this, features were extracted from N sample ngerprint images of the same person, and then aligned. The data used here was associated using the bounded nearest neighbor averaging technique described in the next section. The result is a set of expected values $(\bar{x}_i, \bar{y}_i)$ for each detected minutiae, the number $n \quad N$ of samples having a minutiae in that neighborhood, and the variance and covariance of those n minutiae, $(\ {}^{2}_{x,i}, \ {}^{2}_{y,i}, \rho_i)$. Figure **??** shows these regions where features reliably appear.

These values will be used later to compute a bound on the possible density of cha points for a given quantization error probability.

4.4 Locking Set

The locking set $\mathcal{L}$ is computed in a method similar to the method for nding minutiae variances. A user's nger is scanned and processed N times, resulting in N sets of minutiae, $b_1, ..., b_N$. These are correlated using the following algorithm with distance threshold T and multiplicity threshold S:

1. let A be the set of average points with multiplicity
2. for each minutiae set b_i
3. for each minutiae $m_j \in b_i$
4. nd element in $n_k \in A$ such that $|n_k \quad m_j| < T$
5. select closest n_k that has not already been used
6. if no matches, add m_j to A with multiplicity one
7. else add m_j to average and increase multiplicity
8. $\mathcal{A} = \{a \in A : \texttt{multiplicity}(a) > S\}$

Features appearing in S or fewer scans are discarded as noise. These points generally occur in the edge regions of the image, where feature extraction is less reliable. This locking set can then be used to create the ngerprint vault.

4.5 Chaff Points

The number and location of cha points is limited by the variance in the ngerprint capturing and feature extraction algorithm. Cha points cannot be placed too close to real points, or they will cause quantization problems. Given

Figure 2: Feature Extraction Process: (a) original image, (b) after edge detection, (c) including feature points

some acceptable distance d is found, cha points can be placed anywhere as long as they are at least distance d from any real points. Additionally, there is no reason to place cha points next to each other at any distance less than d, because an attacker can immediately ignore them as unlikely candidates, as they are so close together.

LEMMA 2. *Given elements of $\mathbb{F}_{p^2}$ have pairwise Euclidean distance no less than d, the total number of elements r with packing density ρ is less than $\frac{4\rho p^2}{d^2\pi}$.*

The optimal packing technique for circles is using a hexagonal lattice, and has a packing density of $\rho = \frac{\pi}{2\sqrt{3}} \approx 0.91$ [**?**]. Unfortunately, this density could never be achieved, as we require the locations of cha points to look random. If they all existed on a lattice, any discontinuities in the lattice pattern would be the real points.

Points can be randomly packed by repeatedly selecting random eld elements and putting a cha point there if it is at least distance d from all other points. This yields a packing density of $\rho \approx 0.45$, and is guaranteed to be random. Another technique, *random close packing* [**?**], could yield densities closer to $\rho \approx 0.75$, but these techniques have not been su ciently studied in the two-dimensional case, and their randomness has never been quanti ed.

For this results to be useful, a minimum distance d between points needs to be computed in terms of the vault parameters. Here, we make a simplifying assumptions which will slightly loosen our bound, but yield simpler results: the noise distribution is spherically Gaussian, or the two axes are independent and identically distributed.

LEMMA 3. *The probability of successfully decoding a single point at distance at least d from all others using the maximum likelihood rule is*

$$P_s \quad 1 \quad \exp\left(\frac{d^2}{8\ ^2}\right) \tag{2}$$

THEOREM 1. *The probability of error for decoding points at least δ out of t points in the fuzzy vault $\mathcal{V}(\mathbb{F}_{2^p}, r, t, k)$ is bounded below by*

$$P_e \geq \sum_{i=\delta}^{t} \binom{t}{i} \exp\left(\ \frac{\rho p^2}{2r\pi\ ^2}\right)^i \left(1 \quad \exp\left(\ \frac{\rho p^2}{2r\pi\ ^2}\right)\right)^{t\ \ i} \tag{3}$$

for a given point variance ^{2}.

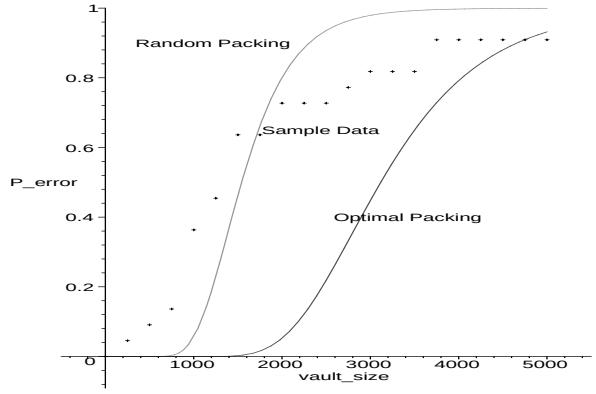

Figure 4: Probability of error as a function of r, using $^2 = 9$ and $p = 251$, with vault parameters $t = 40$ and $\delta = 12$, plotted with error probabilities using random packing in sample data sets.

Figure **??** shows the error probability as a function of the vault size for the optimal and randomized packing methods. Also included is error probabilities from actual ngerprint data using the random packing method. For each person, ve ngerprint scans were available. The rst four were used to create a vault, and the fth was used to try and unlock it. The plotted probabilities represent the fraction of data sets that had enough true points in the unlocking set to successfully unlock the ngerprint vault. The experimental results are quite close to the theoretical, which is impressive given our simpli ed noise model.

Note that this is the probability of being able to successfully decode, and does not deal with the complexity of actually performing that decoding.

4.6 Unlocking Set

The unlocking set $\mathcal{U}$ starts o initially as some set U of minutiae locations from a single scan of the user's nger. To select the elements of the unlocking set, for each point in U the user nds the closest point in the vault. If the ngerprint capture process did not introduce noise features, we should have $\mathcal{U} \subseteq \mathcal{L}$. However, there is some probability that a minutiae will by chance be closer to a cha point than a true point. Also, there is a chance that U will contain

spurious minutiae which was not included in $\mathcal{L}$.

This is where Reed-Solomon codes comes into play. For any n points, we can determine the polynomial, or Reed-Solomon codeword in this case, if at least $\frac{\delta+n}{2}$ of those points are correct. If n is reasonably close to r, then very few attempts will be required to compute the polynomial.

5. UNLOCKING COMPLEXITY

Vault unlocking can be viewed in two contexts. The rst is the complexity of a valid user unlocking a vault with a matching ngerprint image. One goal is to minimize this complexity. The second context is the complexity of an attacker without ngerprint information trying to crack the vault. We wish to maximize this complexity while the attacker wishes to minimize it.

There are two obvious techniques for unlocking the vault. The rst is the brute-force method, or $\mathbf{bf}(r,t,k)$, where r is the total number of points, t is the number of real points, and k is the degree of the polynomial. For an attacker, r and t are the same as the ones in the vault parameter, however for a valid user, r is the size of their unlocking set and t is the number of non-cha points in that set.

THEOREM 2. *The complexity of the* $\mathbf{bf}(r,t,k)$ *problem using a suitable* δ *to ensure a unique result is* $\mathcal{C}_{bf} = \binom{r}{\delta}\binom{t}{\delta}^{-1}$.

EXAMPLE 1. *Consider a vault over* $\mathbf{F}_{251^2}$ *with* $r = 1000$ *total points and* $t = 40$ *real points over a degree* $k = 8$ *polynomial. Under Corollary* **??**, $\delta = 12$. *Using Theorem* **??**, *the complexity of an attacker breaking the vault is approximately* 2^{58} *polynomial interpolations.*

EXAMPLE 2. *Consider the same vault, only a valid user intersects their unlocking set with the vault to obtain* $r = 30$ *points,* $t = 22$ *of which are real points. With such a small vault, obviously* $\delta = k+1 = 9$. *Using Theorem* **??**, *the complexity of a valid user unlocking the vault is approximately* 2^7 *polynomial interpolations.*

The second example illustrates that a brute-force decoding algorithm is less than ideal a valid user. Another method of unlocking the vault is through the use of a Reed-Solomon decoder. In the $\mathbf{rs}(r,t,n,\delta)$ problem, r, t, and δ have the same meanings as before, and n is the size of the Reed-Solomon codewords involved.

THEOREM 3. *The complexity of the* $\mathbf{rs}(r,t,n,\delta)$ *problem over* $\mathbb{F}_{p^2}$ *is*

$$\mathcal{C}_{rs} = \binom{r}{n}\left(\sum_{i=\max(\frac{n+\delta}{2},n\quad r+t)}^{\min(n,t)} \binom{r\quad t}{n\quad i}\binom{t}{i}\right)^{-1} \tag{4}$$

such that n *satisfies* $\delta \quad n \quad \min(r, 2t \quad \delta)$ *and* $n|(p^2 \quad 1)$.

COROLLARY 2. *The complexity of* $\mathbf{bf}(r,t,\delta) = \mathbf{rs}(r,t,\delta,\delta)$, *or taking* $n = \delta$ *reduces Reed-Solomon unlocking into a brute-force unlocking.*

COROLLARY 3. *For* $r >> t$, $\mathbf{bf}(r,t,\delta) \quad \mathbf{rs}(r,t,n,\delta)$, *for all* $n \geq \delta$, *or unless an attacker can eliminate a significant number of chaff points of a locked vault, he or she can do no better than a brute-force attack.*

Let's examine the previous two examples in the context of Reed-Solomon decoding. Figure **??** illustrates the complexity of a full-scale attack, an attack where partial ngerprint information is known, and a legitimate unlocking of the vault. We can see that depending on the relationship of r and t, the optimal method for unlocking the vault can change. By using a Reed-Solomon decoder, a valid user can now unlock the vault in 1 or 2 tries, while an attacker can still do no better than a brute-force attack.

If an attacker is able to eliminate many of the cha points, presumably through side knowledge of some of the ngerprint characteristics, nding the optimal attack now becomes more interesting. The minimum complexity is no longer one of the bounding cases.

In this paper, we assume that we can choose cha points in such a way as to confuse the attacker and force him to consider all points. For the most part we can disregard attacks where an attacker can eliminate certain cha points based on the probable minutiae con gurations. Research [**?**, **?**, **?**] indicates that the probability of two people having the same ngerprint is approximately 1×10^{-80}. Assuming ngerprints are equiprobable, this corresponds to $\log_2(1 \times 10^{-80}) \approx 265$ bits of entropy. Thus, for a complexity-theoretic attack, the minutiae entropy is su - ciently large, thus trivializing attacks which take probable minutiae con gurations into account. This also indicates we could never achieve more than 265-bit security, regardless of vault parameters.

6. EMPIRICAL RESULTS

Throughout the paper, the term "reasonable vault parameters" has been used repeatedly. Here, we use actual ngerprint data to determine what "reasonable" is. Each of the vault parameters, p, r, t, and k has limitations placed on it by the behavior of actual ngerprint data.

The locking and unlocking algorithms were implemented in MATLAB, and sample ngerprint data was used to test the error probabilities. Four scans of the same individual were used to create a vault, and a fth used to try and unlock it. The number of true points in the unlocking sets for these real vaults was used to validate the statistical models.

The eld, $\mathbb{F}_q$, de nes the underlying mechanics of our entire system. Throughout, we have been using $\mathbb{F}_{p^2}$, for prime p. In general, we wish to represent a feature pixel location. For the examples presented so far, $p = 251$ was used. This way, minutiae locations can be stored in 16-bit numbers, and $251^2 \quad 1 = 63000 = 2^3 \quad 3^2 \quad 5^3 \quad 7$, a very smooth number, yielding many choices for the Reed-Solomon codeword size.

Increasing the ngerprint image resolution and consequently the eld size has little e ect on the resulting security. As the resolution increases, so does the minutiae variance, . These two parameters cancel one another out, making the underlying eld selection based more on convenience than security.

The number of real points, t, is the size of the locking set. The algorithm described earlier takes several scans of the same person and locates minutiae that appear in two or more of the scans. This algorithm was implemented in MATLAB and used to create various locking sets. For 5 scans of each person and $S = 1$, we obtain locking sets which ranged from 25 to 60 points, with mean 38 and standard deviation 11.

The degree of the polynomial the vault protects is bounded below by the amount of data we wish to encode in it. Each

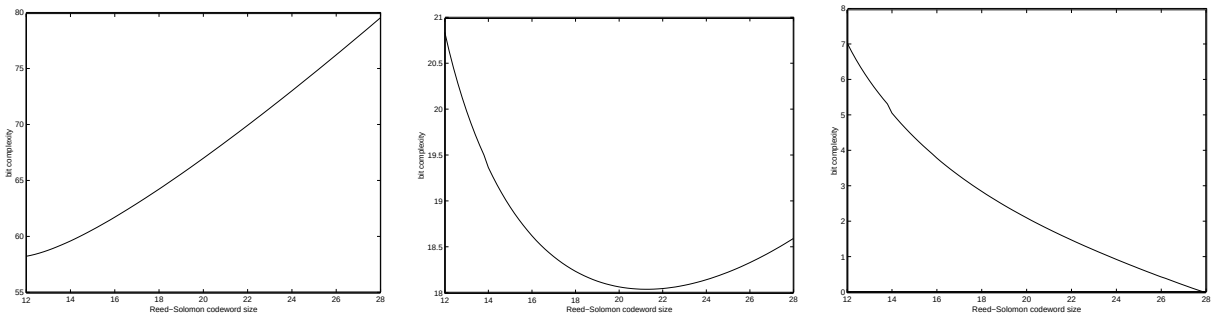

Figure 5: Log of complexity for Reed-Solomon decoding as a function of codeword size; (a) complexity of full attack, rs$(1000, 40, n, 12)$; (b) complexity of partial information attack, rs$(120, 40, n, 12)$; (c) complexity of legitimate unlocking, rs$(30, 22, n, 12)$.

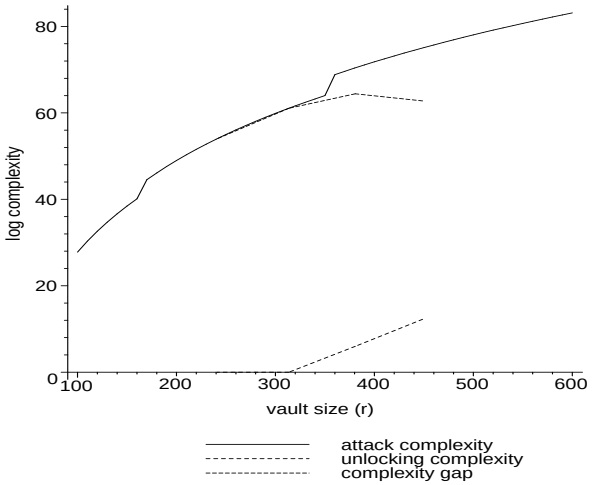

Figure 6: Vault performance as a function of vault size, with $k = 14$, $\tau = 20$, and $t = 38$ over $\mathbb{F}_{251^2}$

coe cient is an element of $\mathbb{F}_{251^2}$, and can therefore hold 15.9 bits of information. As a result, a 128-bit key can be encoded using 9 coe cients, or in a degree 8 polynomial. Consequently, we shall consider $k \geq 8$.

The total number of points r depends on the number of cha points added to the vault, and is a function of the desired error probability. Figure **??** gave the probabilities for a particular set of input values. Here, we shall examine this trade-o in more detail.

First, examine how the vault performs as a function of its size. Figure **??** shows both the complexity of a normal user, and the complexity of an attacker for a vault with 38 real points over a degree 14 polynomial. We can see that as the total number of points increases, so do both complexities. In order to keep user complexity to a minimum, we shall select the largest value of r such that the user has zero complexity.

The other key parameter that can be varied to alter our vault performance is k, the degree of our polynomial. Figure **??**(a) shows the attack complexities as a function of k. This complexity was computed by rst nding the maximum number of points r such that the user has zero complexity, and from there computing δ, the minimum number of points interpolating our polynomial in order to guarantee success. Using r and δ, the di culty of a brute-force attack can be computed.

Figure **??**(b) is essentially a reality check on our selection of τ, the size of our unlocking set. For a given τ and k, real ngerprint data was again used to compute the probability of successfully unlocking the vault. It can be seen that given $\tau = 20$, approximately 20 to 30 percent error occurs. From a user's perspective, this means that every couple times they access their smartcard, a second ngerprint scan will be required in order to successfully unlock the vault. This seems reasonable given that we expect that the false positive rate to be in nitesimally small. The corresponding curve in **??**(a) indicates that the maximum complexity is 2^{69} for $k = 14$.

Consequently, over $\mathbb{F}_{251^2}$ we have determined the optimal vault to be:

- polynomial: $k = 14$, $\delta = 17$
- cha points: $r = 313$, $d = 10.7$
- attack complexity: 2^{69}

7. CONCLUSION

In this paper, we have considered the practical implications of using ngerprint information to secure a smartcard. Because ngerprints are often inconsistent, we must resort of a fuzzy scheme for storing the secret key. We show that with real-life parameters, it is impossible to ensure the security envisioned by Juels and Sudan. However, we de ne a modi ed scheme called the *fingerprint vault*, provide associated algorithms and a mechanism for nding optimal vault parameters. Parameters are provided which makes retrieving the secret 2^{69} times more di cult for the attacker than a legitimate user.

There are a couple ways by which security may be improved. An obvious way is to use multiple ngerprints to store a longer private key which could be hashed down to the appropriate length. Another way is to improve the detection and extraction algorithms so as to lower , allowing us to pack in more cha points.

8. REFERENCES

[1] Agrawal, D., Archambeault, B., Rao, J., and Rohtagi, P. The em-side channel(s). Workshop on Cryptographic Hardware and Embedded Systems, CHES 2002.

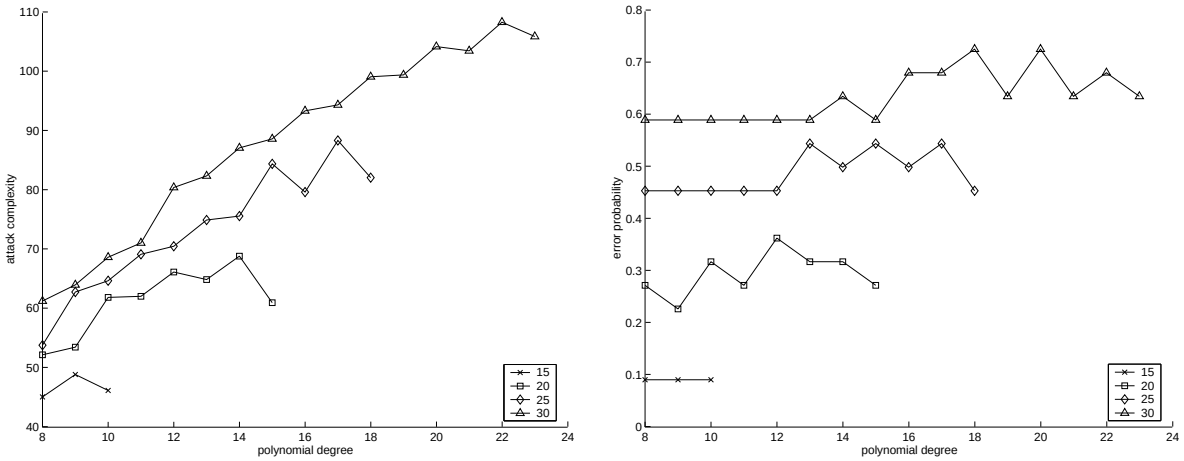

Figure 7: Vault performance as a function of k and τ: (a) attack complexities as a function of k for various τ; (b) decoding failure as a function of k for various τ

[2] Arora, S., and Khot, S. Fitting algebraic curves to noisy data. ACM Symposium on Theory of Computing, STOC 2002.

[3] Blahut, R. *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[4] Blahut, R. *Modem Theory: An Introduction to Telecommunications*. Cambridge University Press, preprint.

[5] Bleichenbacher, D., and Nguyen, P. Q. Noisy polynomial interpolation and noisy chinese remaindering. Advances in Cryptology, EUROCRYPT 2000.

[6] Davida, G., Frankel, Y., and Matt, B. On enabling secure applications through o -line biometric identi cation. IEEE Symposium on Privacy and Security, 1998.

[7] Guruswami, V., and Sudan, M. Improved decoding of reed-solomon and algebraic-geometric codes. Symposium on Foundations of Computer Science, FOCS 1998.

[8] Hildebrand, F. B. *Introduction to Numerical Analysis*. McGraw-Hill, 1956.

[9] Jaeger, H., and Nagel, S. Physics of granular states. *Science 255*, 1524 (1992).

[10] Juels, A., and Sudan, M. A fuzzy vault scheme. ACM Conference on Computer and Communications Security, CCS 2002.

[11] Juels, A., and Wattenberg, M. A fuzzy commitment scheme. ACM Conference on Computer and Communications Security, CCS 1999.

[12] Kocher, P. Timing attacks on implementations of di e-helmman, rsa, dss, and other systems. Advances in Cryptology, CRYPTO 1996.

[13] Kocher, P., Jaffe, J., and Jun, B. Di erential power analysis. Advances in Cryptology, CRYPTO 1999.

[14] Kuhn, M., and Anderson, R. Tamper resistance: A cautionary note. Workshop on Electronic Commerce, USENIX 1996.

[15] Kummerling, O., and Kuhn, M. Design principles for tamper-resistant smartcard processors. Workshop on Smartcard Technology, USENIX 1999.

[16] Looi, M., Ashley, P., Seet, L. T., Au, R., and Vandenwauver, M. Enhancing sesamev4 with smart cards. International Conference on Smartcard Research and Applications, CARDIS 1998.

[17] Massey, J. L. Shift register synthesis and bch decoding. *IEEE Transactions on Information Theory 15*, 1 (1969), 122–127.

[18] Monrose, F., Reiter, M., and Wetzel, S. Password hardening based on keystroke dynamics. ACM Conference on Computer and Communications Security, CCS 1999.

[19] Nichols, R. K., Ed. *ICSA Guide to Cryptography*. McGraw-Hill, 1999, ch. Biometric Encryption.

[20] Osterberg, J., Parthasarathy, T., Raghavan, T., and Sclove, S. Development of a mathematical formula for the calculation of ngerprint probabilities based on individual characteristics. *Journal of the American Statistical Association 72* (1977), 772–778.

[21] Pankanti, S., Prabhakar, S., and Jain, A. On the individuality of ngerprints. *IEEE Transactions on PAMI 24* (2002), 1010–1025.

[22] Sclove, S. The occurance of ngerprint characteristics as a two-dimensional process. *Journal of the American Statistical Association 74* (1979), 588–595.

[23] Steinhaus, H. *Mathematical Snapshots*, 3 ed. Dover, 1992.

[24] Vandenwauver, M., Govaerts, R., and Vandewalle, J. Overview of authentication protocols: Kerberos and sesame. IEEE Carnahan Conference on Security Technology 1997, pp. 108–113.

[25] Verifinger. Neurotechnologija ltd. http://www.neurotechnologija.com.

[26] Ylonen, T. Ssh secure login connections over the internet. Security Symposium, USENIX 1996, pp. 37–42.