Secret Locking: Exploring New Approaches to Biometric Key Encapsulation

Seny Kamara
Johns Hopkins University
Department of Computer Science
3400 N Charles Street, Baltimore, MD 21218, USA
Email: seny@cs.jhu.edu

Breno de Medeiros
Florida State University
Department of Computer Science
Tallahassee, FL 32306-4530, USA
Email: breno@cs.fsu.edu

Susanne Wetzel
Stevens Institute of Technology
Department of Computer Science
Castle Point on Hudson, Hoboken, NJ 07030, USA
Email: swetzel@cs.stevens.edu

Keywords: Biometrics, secret sharing, secret locking, error-tolerance

Abstract: Biometrics play an increasingly important role in the context of access control techniques as they promise to overcome the problems of forgotten passwords or passwords that can be guessed easily.
In this paper we introduce and provide a formal definition of the notion of *secret locking* which generalizes a previously introduced concept for cryptographic key extraction from biometrics. We give details on an optimized implementation of the scheme which show that its performance allows the system for use in practice. In addition, we introduce an extended framework to analyze the security of the scheme.

1 Introduction

Biometrics play an increasingly important role in a broad range of security applications. In particular, biometrics have manifold applications in the context of access control techniques which to date are largely based on the use of passwords. Biometrics promise to overcome the problems of forgotten passwords or passwords that can be guessed easily.

Most biometric systems used in practice to date store *profiles* of users. A user profile typically consists of a collection of measurements of the user's physical characteristics (e.g., the user's iris patterns or fingerprints) obtained during an initial enrollment phase. Later, when a user presents herself for identification, the system performs measurements and matches those against the database of stored user profiles. If a "good" match is found, the user is identified. While these systems protect against an *online* attacker, they however, pose a considerable risk for *offline* attacks in which an attacker may obtain and exploit the knowledge of the stored profiles.

Recently, the alternative approach of *biometric key encapsulation* has been proposed: instead of replacing the use of passwords by means of biometrics, passwords are "hardened" by incorporating biometric features. No user profiles are stored in the system. Due to the inherent variability in biometric readings, the system, however, requires a *biometric feature extractor* in order to reliably recover the same (cryptographic) key from an imprecise input, i.e., to provide "error tolerance." In this context, a solution based on error-correcting codes and randomness extraction was developed (Juels and Wattenberg, 1999; Juels and Sudan, 2002; Dodis et al., 2004; Boyen, 2004). An alternative line of work based on secret sharing techniques was proposed in (Monrose et al., 2002; Monrose et al., 2001). While the former provides an information-theoretical optimal solution for error-tolerance, it at the same time requires a uniform level of error-tolerance for all users alike and as such poses significant challenges for use in practice. In contrast, the latter allows for an individual level of error-tolerance for each user.

In this paper we focus on extending the work in (Monrose et al., 2002). In particular, we introduce a formal definition of the notion of *secret locking* which generalizes the concept proposed previously. We furthermore provide an extended discussion on the determinant-based scheme. We give details on an optimized implementation of the scheme which show that its performance allows the system for use in practice. In addition, we introduce an extended framework to analyze the security of the scheme. In the original work, the security of the determinant-based construction was proved under an idealized attack model only. In this paper we consider arbitrary attacks. Finally, we discuss heuristic connections between the security of the scheme and well-known hard problems in computational mathematics and coding theory.

1.1 Motivation

Using biometrics in practice poses a number of challenges, in particular when used in applications to protect resource limited devices such as cell phones or PDAs. Ideally, these devices should obtain biometric measurements without requiring any additional dedicated hardware. Currently, most portable devices have built-in microphones, keyboards or writing pads. As such, systems using biometrics such as voice patterns, keystroke dynamics or stylus drawing patterns are more readily deployable than systems based on iris or retina scans. Furthermore, it should be difficult for an adversary to capture the user's biometric measurements, and in particular this counter-indicates fingerprint scans as a biometric in this regard, as fingerprint marks are quite easy to obtain.

Static vs. Non-static Biometrics. While *static* biometrics capture physiological characteristics of an individual (e.g., iris or retina patterns, and fingerprints), *non-static* biometrics (e.g., voice patterns, keystroke dynamics) relate to behavioral characteristics. In general, it is harder for an attacker to capture non-static than static biometrics, so they could prove useful for the type of application we consider. However, non-static biometrics have a high variability of robustness from user to user: Some users have more reliably reproducible feature readings than others. Consequently, less error-tolerance is required to support identification of users with more reliably reproducible feature readings (Doddington et al., 1998).

Biometric Key Encapsulation requires the exact reconstruction of the underlying key, and some form of error-tolerance must therefore be employed in order to accommodate the variability in biometric readings. In order for a system to accommodate different levels of error-tolerance allowed to identify particular users, ideally it should allow for variable error-tolerance. Alternatively, the system-wide level could be adjusted to the worst case, i.e., the least robust user. In (Juels and Wattenberg, 1999; Juels and Sudan, 2002; Dodis et al., 2004; Boyen, 2004) error-tolerance is achieved by means of error-correcting codes and randomness extraction. In practice, this solution either requires uniformity, with the same error-correcting code employed for all users, or the codes need to be defined on a user-by-user basis. While the former solution suffers from the problem that the security of the system is reduced to the level of the least robust user, the latter reveals to an attacker the code used (and therefore the level of error-tolerance supported) upon inspection.

In contrast, the system introduced by Monrose et al. allows for non-uniformity of robustness of a user's biometric characteristics. In particular, the system hides the amount of error-tolerance required by a specific user. In other words, if the attacker has access to the key encapsulation value, his effort to decide how much error-tolerance the particular user required should be roughly equal to the effort of breaking the key encapsulation of that user.

2 Related Work

There are numerous approaches described in literature to use biometrics for authentication purposes or to extract cryptographic secrets from biometrics. There are various systems using biometric information during user login process (e.g., (Joyce and Gupta, 1990)). These schemes are characterized by the fact that a model is stored in the system (e.g., of user keystroke behavior). Upon login, the biometric measurements (e.g., user keystroke behavior upon password entry) are then compared to this model. Since these models can leak additional information, the major drawback of these systems is that they do not provide increased security against offline attackers.

In (Soutar and Tomko, 1996), a technique is proposed for the generation of a repeatable cryptographic key from a fingerprint using optical computing and image processing techniques. In (Ellison et al., 2000), cryptographic keys are generated based on users' answers to a set of questions; subsequently, this system was shown to be insecure (Bleichenbacher and Nguyen, 2000). Davida, Frankel, and Matt (Davida et al., 1998) propose a scheme which makes use of error-correction and one-way hash functions. The former allows the system to tolerate a limited number of errors in the biometric reading. This approach was generalized and improved in (Juels and Wattenberg, 1999) by modifying the use of error-correcting codes.

In (Monrose et al., 2002; Monrose et al., 2001), a new approach is proposed, focusing on using keystroke features and voice characteristics to *harden* the passwords themselves. The work improves on previous schemes in that it is the first to offer better security against a stronger attacker. Furthermore, this approach allows a user to reconstruct the key even if she is inconsistent on a majority of her features. The techniques introduced by (Ellison et al., 2000; Davida et al., 1998; Juels and Wattenberg, 1999) respectively, do not permit that.

Recently, a new theoretical model for extracting biometric secrets has been developed (Juels and Sudan, 2002; Dodis et al., 2004; Boyen, 2004), extending the work in (Juels and Wattenberg, 1999). The model is based on the use of population-wide metrics combined with (optimal) error-correction strategies. While the model is provably secure and allows

for optimal constructions under certain assumptions, it has not been empirically validated that these constructions are applicable to biometrics of interest in practice.

3 Secret Locking and Secret Sharing Schemes

In the traditional setting, a *secret sharing scheme* consists of a dealer, a set of participants $P = \{P_1, \ldots, P_n\}$, an access structure $\Gamma \subseteq 2^P$ as well as algorithms Share and Recover. In order to share a secret s amongst the participants, the dealer uses the algorithm Share to compute each share s_i to send to user P_i. In order to reconstruct the shared secret using the algorithm Recover, only those shares are needed which correspond to authorized subsets of participants —i.e., shares corresponding to sets in the access structure Γ. The most well-known secret sharing schemes are threshold schemes. While these schemes have a simple access structure (which contains all user sets of cardinality larger than a threshold t, i.e., $S \in \Gamma \iff |S| > t$), for use with biometrics, we are interested in secret sharing schemes with different properties.

The concept of a compartmented access structure was introduced in (Simmons, 1990), and has received attention from a number of researchers (Brickell, 1989; Ghodosi et al., 1998). In a compartmented secret sharing scheme, each user P_i is assigned a level $\ell(P_i)$. The same level may be assigned to different users. In order to reconstruct the secret, one share from each level is needed. More formally, the access structure of the compartmented secret sharing scheme is $\Gamma = \{A \in 2^P : A \cap P^i \neq \emptyset\}$, where $P^i = \{P_j \in P : \ell(P_j) = i\}$.

Compartmented access structures can be used to achieve error-tolerance in biometric key encapsulation: Let $\phi = (\phi_i)_{i=1,\ldots,m}$ be the set of discretized measurements[1] of biometric features (for instance timing intervals between different keystrokes). Each ϕ_i assumes a value in the same finite set D. For each user U, let $R_i(U) \subset D$ be the range of values that are likely[2] to be observed by measuring ϕ_i on user U. In order to encapsulate a key K for user U, where the key is a random value from a finite field $\mathbb{F}_q$, proceed as follows. First, define a virtual participant set $P = \{P_{i,j}\}_{\{i=1,\ldots,m;j\in D\}}$, and assign to $P_{i,j}$ a level $\ell(P_{i,j}) = i$. Next, use the Share algorithm for the compartmented access structure to compute initial shares $\hat{s}_i^j$. Finally, perturb this initial set of shares to obtain shares s_i^j which match the initial shares $\hat{s}_i^j$ whenever $j \in R_i(U)$, and are set to a newly chosen random share value otherwise. When the legitimate user U presents herself for authentication, it is sufficient to measure each value $\phi_i(U) \in D$ of the biometric feature ϕ_i on user U, then select the share $s_i^{\phi_i(U)}$ from level i and apply the Recover algorithm. By construction, the outcome is likely to be the encapsulated secret K. On the other hand, the same is not likely to be the case if a different user U' tries to impersonate U, as the feature values for U' are not likely to align (i.e., fall in the likely range at each level) with those of U.

The above idea can be readily applied with any efficient compartmented secret sharing scheme, such as that in (Ghodosi et al., 1998), if the target is simply user authentication. However, as we seek mechanisms to achieve secure key encapsulation, the scheme must moreover have the property that an attacker who has access to the set of all shares cannot determine which shares to pick at each level. It can be readily seen that the scheme in (Ghodosi et al., 1998) is not secure in this sense, and therefore is not sufficient for our purposes.

We can abstract the previously introduced concepts as follows: Let D be a finite set, and consider the product set D^m. We call an element $(\phi_i)_{i=1,\ldots,m} \in D^m$ a *sequence of feature values.* Consider some universe $\mathcal{U}$, and for each element U of the universe, and for each feature value ϕ_i we associate the *likely range*, a subset $R_i(U) \subset D$. Let $\rho_i = R_i(U)/D$ be the relative size of the likely range $R_i(U)$. Let $\tau_i(U)$ be defined as $-\log(\rho_i)$, which equals the logarithm of the expected number of random trials before a value for $\phi_i(U)$ is chosen within U's likely range $R_i(U)$, among all values in D. Finally, let $\tau_U = \sum_i \tau_i(U)$. The value τ_U is the logarithm of the expected number of random trials before one produces a sequence of likely features (for U) by simply choosing random sequences in D^m. Clearly, τ_U is a natural parameter of the difficulty of guessing likely sequences for U.

Definition 1 *A secure secret locking scheme is a set of algorithms* Share *and* Recover *with the following properties:*

1. *Given a compartmented participant set $P = \{P_{i,j}\}_{\{i=1,\ldots,m;j\in D\}}$, where D is a finite set, and given a secret K in $\mathbb{F}_q$,* Share *produces a collection s_i^j of shares (which are values in a set $\mathcal{S}$) which implement the access structure $\Gamma = \{A \in 2^P : A \cap P^i \neq \emptyset, i = 1, \ldots, m\}$, where $P^i = \{P_{i,j}\}_{j\in D}$. In other words,* Share *and* Recover *implement a*

[1] Biometric measurements are continuous values. Measurements are discretized by breaking the range of the measurement into equal probability ranges.

[2] One needs repeated measurements of each biometric feature in order to arrive at the range of likely values. Particularly with non-static biometrics this range may vary over time. Refer to (Monrose et al., 2002) for details of a practical implementation of such a scheme.

compartmented secret sharing scheme with levels $i = 1, \ldots, m$.

2. *Assume that a set of shares* s_i^j *originally produced by* Share *has been perturbed by substituting for* s_i^j *a random element of* $\mathcal{S}$ *whenever* j *is not a likely value for* $\phi_i(U)$. *Then, each probabilistic algorithm* $\mathcal{A}$*, that receives as input the share set (partially randomized as above), and that terminates in polynomially many steps in* τ_U *has negligible probability of success in recovering the original shared secret.*

Binary Features: In the following we describe some general constructions of the secret locking concept introduced in (Monrose et al., 2002). For simplicity of argument, we assume that all features assume binary values, i.e., $D = \{0, 1\}$, even though all schemes described can be generalized to any finite D. In the binary case, the range of values $R_i(U)$ for a feature i and element U is one of three possibilities, namely $\{0\}$, $\{1\}$, or $\{0, 1\}$. In the latter we call the feature *non-distinguishing* for U, while in the former two cases the feature is *distinguishing*.

3.1 Secret Locking Constructions

For each construction, it is sufficient to provide the algorithms Share and Recover, as the security property is not constructive. Instead it must be verified for each construction. We first describe an implementation of secret locking introduced in (Monrose et al., 2002) which is based on the well-known Shamir secret sharing scheme:

Shamir Secret Sharing (SSS) is based on polynomial interpolation. In general, for a random polynomial $f(x)$ over $\mathbb{Z}_p$ of degree $d - 1$ and a secret $K = f(0) \in \mathbb{Z}_p$ to be shared, a share will be determined as a point on the polynomial, i.e., as the tuple $(x, f(x))$. Using Lagrange interpolation, the knowledge of at least d distinct shares will allow the reconstruction of the secret K (Shamir, 1979).

In order to construct a secret locking scheme based on SSS, it is sufficient to choose $f(x)$ as a polynomial of degree $m - 1$ with $f(0) = K$. The $2m$ shares $\{s_i^0, s_i^1\}_{1 \le i \le m}$ of secret K are determined as $s_i^0 = f(2i)$ and $s_i^1 = f(2i + 1)$. Consequently, any m shares will allow for the reconstruction of the secret K, and clearly one share per row will do. However, this scheme is not compartmented, but simply a threshold scheme. Furthermore, it does not provide security in the sense of our definition if the percentage of distinguishing features is small (i.e., less than 60% of the total number of features). This is due to the fact that it is then possible to treat the system as a Reed-Solomon list decoding problem, which can be solved by means of a polynomial time algorithms (Guruswami and Sudan, 1998).

A truly compartmented construction based on unimodular matrix constructions is also presented in (Monrose et al., 2002), and is the focus of our attention for the remaining part of the paper.

Determinant-based Secret Locking Construction. The determinant-based scheme introduced in (Monrose et al., 2002) encapsulates a secret by means of a set of vectors in a vector space. In general, for a secret $K \in \mathbb{Z}_p$ to be shared, the shares are determined as vectors in $\mathbb{Z}_p^m$. The secret can be reconstructed by arranging m of the shares in an $m \times m$-dimensional matrix and computing its determinant.

In order to construct the set of shares, initially m vectors s_i^0 in $\mathbb{Z}_p^m$ are chosen with the property that $\det(s_1^0, \ldots, s_m^0) \bmod p = K$, the secret to be encapsulated. The second set of shares is then determined by means of a unimodular transformation matrix $\Upsilon = (\upsilon_1, \ldots, \upsilon_m)$ where $\upsilon_i \in \mathbb{Z}_p^m$ $(1 \le i \le m)$. The unimodular matrix can be efficiently generated by permuting the rows of a random, triangular unimodular matrix: $\Upsilon = \Pi \cdot \Upsilon' \cdot \Pi^{-1}$, where $\Pi = (\underline{\pi}_1, \ldots, \underline{\pi}_m)$ is any permutation matrix and $\Upsilon' = (\underline{\upsilon}'_1, \ldots, \underline{\upsilon}'_m)$ is an upper-triangular matrix that has 1 for each diagonal element and random elements of $\mathbb{Z}_q$ above the diagonal. Eventually, the second set of shares is computed as $s_i^1 = \Upsilon s_i^0$ for $1 \le i \le m$.

It can be easily seen from the way the shares are constructed, that this scheme indeed implements a compartmented access structure. In fact, if one share is picked from each one of the m levels (feature), the secret K can be reconstructed —due to the unimodular relation between the two shares at the same level. However, if the two shares from the same level are used, then the reconstructed secret is random, as the unimodular relationship between the two sets of shares is not preserved. In the following sections, we discuss the security characteristics of this scheme, and provide details on an optimized implementation of the scheme with good performance profile.

3.2 Security

In this section we explore some of the underlying hard problems that are related to the security of the determinant-based sharing scheme described above. Note that while the construction and its analysis are presented only for the case of binary features, similar arguments can be presented for the general case.

First, consider the case when all features are distinguishing, and thus only one sequence of feature values reveals the secret. By construction, all other shares are random and cannot be combined with the true shares to obtain any partial information about the

secret. Moreover, without further information (such as cipher-text encrypted under the encapsulated key) the attacker cannot distinguish when the correct secret is reconstructed. The probability of success is therefore 2^{-m}, where m is the total number of features.

In the presence of non-distinguishing features the setting is different. For instance, consider the case where the first feature is non-distinguishing. Let ϕ^0 and ϕ^1 be two feature sequences that differ only in the first feature, with $\phi_1^0 = 0$ and $\phi_1^1 = 1$. Suppose further that both feature sequences are valid for U, i.e., lead to reconstruction of the correct secret. That means that the following matrices have the same determinant: $K = \det\left(s_1^0\, s_2^{\phi_2}\, s_3^{\phi_3}\, \cdots\, s_m^{\phi_m}\right) = \det\left(s_1^1\, s_2^{\phi_2}\, s_3^{\phi_3}\, \cdots\, s_m^{\phi_m}\right)$, where $\phi_i = \phi_i^0 = \phi_i^1$, for $i > 1$. It is well-known that the determinant is a multi-linear function of the matrix columns, which implies: $\det\left(s_1^0 - s_1^1\, s_2^{\phi_2}\, s_3^{\phi_3}\, \cdots\, s_m^{\phi_m}\right) = 0 \mod p$. We conclude that if the first feature is non-distinguishing one finds a non-trivial algebraic relation on the sets of shares. The method is not constructive, however, because it requires previous knowledge of a valid sequence of values for all the other features. In order to search for such relations systematically, one represents the choice for the value of feature i as a function of a boolean variable:

$$\phi_i(x_i) = \texttt{if}\; x_i\; \texttt{then}\; s_i^1\; \texttt{else}\; s_i^0.$$

The determinant computation may then be fully expanded as a boolean circuit, and the equation which expresses the determinant being equal to $0 \mod p$ reduced to a single boolean formula. Any satisfying assignment to that formula corresponds to a sequence of feature values which may be a valid sequence for U, and conversely all valid sequences for U give rise to satisfying assignments.

Since SAT approximation algorithms can generally only handle relatively small boolean formulas (in the thousands of variables), the complexity of this approach can be estimated by studying a relaxation of the problem. In order to "linearize" the boolean formula, we allow feature choices in the whole field $\mathbb{F}_q$, by putting $\phi_i(x_i) = (1 - x_i)s_i^0 + x_i s_i^1$. Note that $\phi_i(0) = s_i^0$ and $\phi_i(1) = s_i^1$ correspond to legitimate shares, while for other values in $x_i \in \mathbb{F}_q$ there is no natural interpretation to the meaning of $\phi_i(x_i)$. Linearization enables the use of the rich machinery of computational algebra to attack the corresponding "relaxed" problem of finding zeros of the multilinear polynomial $\Delta(x_2, \ldots, x_m)$ which represents the determinant $\det(s_1^0 - s_1^1, (1 - x_2)s_2^0 + x_2 s_2^1, \cdots, (1 - x_m)s_m^0 + x_m s_m^1)$.

The complexity of this zero-finding problem was assessed by means of experiments using the symbolic computation package MAPLE. In particular, the experiments determined the number of non-zero coefficients of $\Delta(x_2, \ldots, x_m)$, for $5 \leq m \leq 10$. It was assumed that only $\lfloor 0.4m \rfloor$ of the features were distinguishing —a conservative approach, since the fewer distinguishing features there are, the more symmetric the polynomial should be, and the greater the chances are that some of its coefficients evaluate to 0. The results or the experiments are shown in Fig. 1. These results support the security of the scheme, as the number of non-zero coefficients exhibits an exponential increase. As a consequence, this renders any algebraic attempts to attack the problem ineffective, and in fact, even the best approximation algorithms known to date to simply counting zeros (as opposed to finding them) on multilinear polynomials have linear cost with the number of non-zero coefficients (Karpinski and Lhotzky, 1991).

$m = 5$	6
$(2, 100, 25, 30.6)$	$(2, 60, 49, 60.25)$
7	8
$(2, 80, 97, 120.6)$	$(3, 60, 225, 252.3)$
9	10
$(3, 40, 449, 502.6)$	$(4, 10, 961, 1017.7)$

Figure 1: m = # of features. The quadruplet under $m = 6$ indicates that the # of distinguishing features was 2, and Δ had a minimum of 49 and an average of 60.25 non-zero coefficients over 60 random trials.

4 Implementation

In order to implement the scheme in practice, it is not sufficient to have error-tolerance purely from the secret sharing construction, as features ϕ_i will occasionally assume a value outside the likely range $R_i(U)$ even if evaluated on the legitimate user U. We call such errors "noisy errors." Unlike the natural variation of measurements within likely ranges, the variability introduced by noisy errors is not tolerated well by the secret locking construction. In practice, we can accommodate a few of these errors by simply executing an exhaustive search on a Hamming ball of small radius e centered on the measured input sequence $\phi = (\phi_i(U))_{i=1,\ldots,m}$. We show that with appropriate optimizations, this method is practical for small values of e, for instance $e \leq 3$, which seems more than sufficient to guarantee a reasonable false negative rate with keyboard typing patterns (Monrose et al., 2002).

The first optimization we made was to change the mechanism for reconstructing the secret from the selected matrix entries. Instead of insisting on sharing the determinant —which would require working with matrices over large finite fields $\mathbb{F}_q$, with $\log q \geq 80$

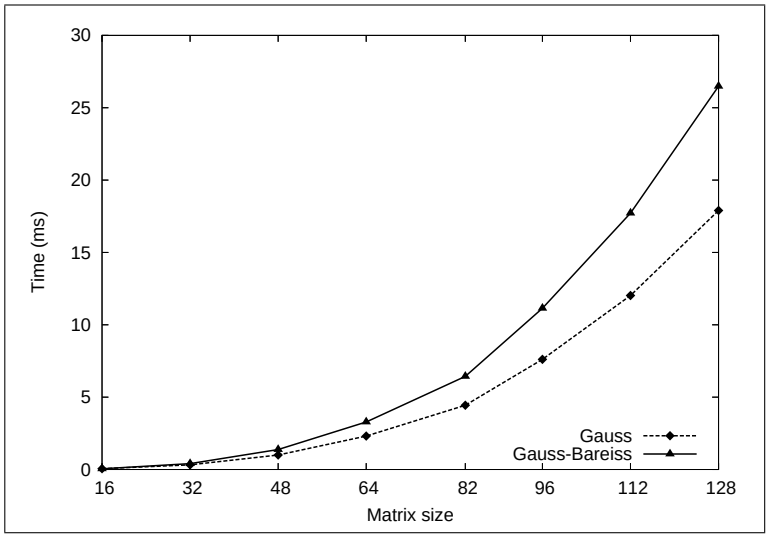

Figure 2: Time to compute determinants as a function of the matrix size. The times are averaged over 10000 runs.

—we instead use a hash function such as SHA-1 to process the concatenation of all matrix entries from the distinguishing features. Recall from Section 3.2 that once a feature sequence is found with the correct values for all distinguishing features, the non-distinguishing positions can be detected by showing that the determinant remains unchanged if that feature value is flipped. Using this modified recovery algorithm we can allow the dimension of the base field to be made much smaller, without affecting the entropy of the keyspace. In our experiments we used $\mathbb{F}_q = \mathbb{Z}_{8191}$, which allows each matrix entry to fit in a 16-bit buffer. This also enables implementation of all modular and matrix operations using native 32-bit integer operations and optimized `C` code.

A second optimization is to save the (common value of the) determinant of a correct set of features to enable fast elimination of incorrect guesses during the exhaustive test of possible candidates in the Hamming ball. Note that saving the determinant does not present a security risk, since (1) the secret is no longer the determinant, but the entirely independent hash value of the concatenated matrix entries corresponding to distinguishing positions, and (2) identifying the determinant value reduces the degree of freedom in the possible choices by adding a single polynomial relation between these choices. Note that this extra information is comparable to that available to an adversary that is able to correctly guess the position of a non-distinguishing feature—when it may write a similar relation with determinant to equal zero (see Section 3.2).

Experimental Setup. All the experiments were conducted on a 64-bit dual 2 GHz PowerPC G5 running MacOS Server 10.3.5, with 3 GB main memory, and 4 KB virtual pages. Our implementation is in `C` and compiled with `gcc 3.3` using the `-O3`, `-ffast-math`, `-malign-natural` and `-fprefetch-loop-arrays` optimization flags. (For more details on `gcc` optimizations see (The GNU Project, 2005).) We note that the Apple G5 provides native support for 32-bit applications and that all our code was compiled for a 32-bit architecture. All arithmetic is performed in $\mathbb{Z}_p^*$, where p is prime and equals $8191 = 2^{13} - 1$. Since we are working on a 32-bit architecture and $8191 < 2^{16}$, all elements in $\mathbb{Z}_{8191}^*$ can be stored in `shorts`. This means that multiplication in $\mathbb{Z}_{8191}^*$ will not overflow the size of a regular `int` and that we can implement the scheme without multi-precision arithmetic.

	$m = 16$	32	48
$e = 2$	0.009	0.220	1.371
3	0.041	1.688	13.554
4	0.174	10.559	184.924

Figure 3: Time (in seconds) to recover the key K from a feature sequence ϕ' such that $dist(\phi, \phi') = e$, where e = # of errors corrected and m = # of features. 80% of ϕ's features are distinguishing.

Computing Determinants. Given feature sequence ϕ', we begin by generating all sequences within a hamming distance e of ϕ'. We call this set $\beta(\phi', e) = \{\phi^* \in \{0,1\}^m : dist(\phi^*, \phi') \leq e\}$. For each $\phi^* \in$

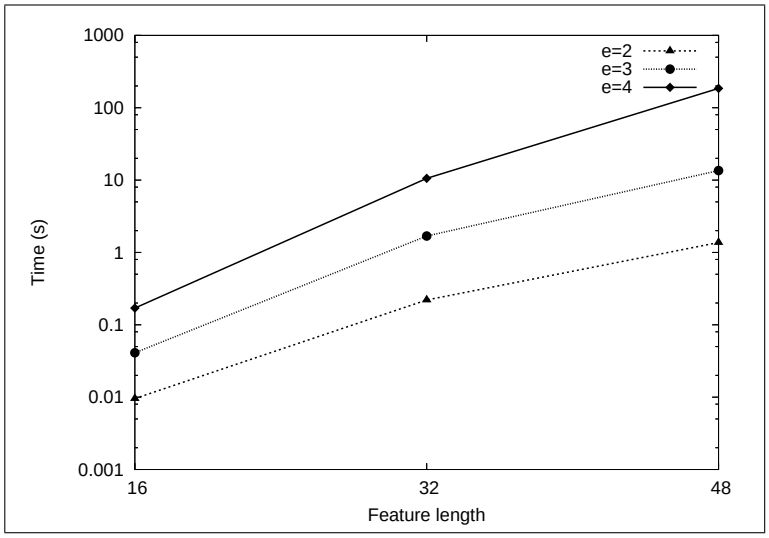

Figure 4: Time to recover the key K from a feature sequence ϕ' such that $dist(\phi, \phi') = e$, where e is the # of errors corrected. 80% of ϕ's features are distinguishing.

$\beta(\phi', e)$, we then compute $\delta' = \det(s_1^{\phi_1^*} \cdots s_m^{\phi_m^*})$ and check whether $\delta' = \delta$, the latter being the stored determinant value of (any) correct choice of feature values. Since $|\beta(\phi', e)| = \sum_{i=0}^{e} \binom{m}{i}$, we have to perform a large number of determinant computations, and therefore it is important to optimize the running time of determinant evaluation. We first benchmarked the performance of various determinant algorithms and implementations, in particular Gaussian elimination and Gauss-Bareiss (both described in (Cohen, 1993)) and compared their performance (Fig. 2), concluding that plain Gaussian elimination performs better in this task. We found that a large part of the time spent was in the computation of modular inverses. Consequently, we precomputed all inverses in $\mathbb{Z}_{8191}^*$ and replaced our use of the extended Euclidean algorithm by simple table lookups. Since each element in $\mathbb{Z}_{8191}^*$ can be stored in two bytes, the entire table can fit in approximately 16 KB.

Reusing Computations. Apart from optimizing individual determinant computations we re-used intermediate elimination results to speed up Gaussian elimination when several determinants are computed in succession. Consider M_1 and M_2, two $m \times m$ matrices. During Gaussian elimination, elements in column i only affect elements in columns $j > i$. If the leftmost column where M_1 and M_2 differ is i, then the operations we perform on columns 0 through $i-1$ when computing $\det(M_1)$ and $\det(M_2)$ will be the same, and we avoid repetition by storing the intermediate results. We take maximum advantage of this optimization by choosing an appropriate ordering when generating all the feature sequences within the Hamming ball. Figures 3 and 4 summarize the timings for recovering the correct key K. That is, these timings include determining the distinguishing features for those $\phi^* \in \beta(\phi', e)$ with $\delta = \det(s_1^{\phi_1^*} \cdots s_m^{\phi_m^*})$ and computing K as the respective hash value.

In the case of keyboard biometrics, the number of features is approximately 15 and one noisy error must be corrected with 12 distinguishing features (numbers from (Monrose et al., 2002)), which our implementation can compute in a fraction of a second. A measure with twice as much entropy, say 30 features and two noisy errors, would also take less than half a second. These results were obtained in a powerful machine by today's standards, however these times are sufficiently small that we feel confident the scheme can be practically implemented in most current 32-bit architectures.

5 Conclusions and Future Work

While the security analysis in this paper does not constitute a complete proof in the standard model, the outlined heuristic connections between the security of the scheme and well-known hard problems in computational mathematics show the difficulty of the underlying problem. The remaining open questions will be addressed by future research. In addition, future work includes testing of the implementation for use in the context of other non-static biometrics (e.g., voice patterns).

6 Acknowledgements

The authors would like to thank Fabian Monrose and Mike Reiter for helpful discussions and suggestions throughout this work. The first author is supported by a Bell Labs Graduate Research Fellowship.

REFERENCES

Bleichenbacher, D. and Nguyen, P. (2000). Noisy polynomial interpolation and noisy chinese remaindering. In *Advances in Cryptology—Proc. of EUROCRYPT '2000*, volume 1807 of *LNCS*, pages 53–69. Springer-Verlag.

Boyen, X. (2004). Reusable cryptographic fuzzy extractors. In *Proc. of the* 11^{th} *ACM Conf. on Comp. and Comm. Secur.* ACM Press.

Brickell, E. F. (1989). Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 9:105–113.

Cohen, H. (1993). *A Course in Computational Algebraic Number Theory*, volume 183 of *Grad. Texts in Mathematics*. Springer-Verlag.

Davida, G. I., Frankel, Y., and Matt, B. J. (1998). On enabling secure applications through off-line biometric identification. In *Proc. of the 1998 IEEE Symp. on Secur. and Privacy*, pages 148–157.

Doddington, G., Liggett, W., Martin, A., Przybocki, M., and Reynolds, D. (1998). Sheep, goats, lambs and wolves. a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. In *Proc. of the* 5^{th} *International Conference on Spoken Language Processing*.

Dodis, Y., Reyzin, L., and Smith, A. (2004). Fuzzy extractors and cryptography, or how to use your fingerprints. In *Proc. of Adv. in Cryptology–Eurocrypt'04*.

Ellison, C., Hall, C., Milbert, R., and Schneier, B. (2000). Protecting secret keys with personal entropy. *Future Generation Computer Systems*, 16:311–318.

Ghodosi, H., Pieprzyk, J., and Safavi-Naini, R. (1998). Secret sharing in multilevel and compartmented groups. In *Proc. of the 3rd Australasian Conf. on Info. Secur. and Privacy (ACISP'98)*, volume 1438 of *LNCS*, pages 367–378. Springer-Verlag.

Guruswami, V. and Sudan, M. (1998). Improved decoding of reed-solomon and algebraic-geometric codes. In *Proc. of the* 39^{th} *IEEE Symp. on Found. of Comp. Sci.*, pages 28–37.

Joyce, R. and Gupta, G. (1990). Identity authorization based on keystroke latencies. *Comms. of the ACM*, 33(2):168–176.

Juels, A. and Sudan, M. (2002). A fuzzy vault scheme. In *Proc. of the 2002 IEEE Internl. Symp. on Inform. Theory*, pages 480–ff.

Juels, A. and Wattenberg, M. (1999). A fuzzy commitment scheme. In *Proc. of the* 6^{th} *ACM Conf. on Comp. and Comm. Secur.*, pages 28–36.

Karpinski, M. and Lhotzky, B. (1991). An (ϵ, δ)-approximation algorithm of the number of zeros of a multi-linear polynomial over gf[q]. Technical Report 1991-8569, Uni. Bonn, Inst. für Informatik, Abteilung V.

Monrose, F., Reiter, M. K., Li, Q., and Wetzel, S. (2001). Cryptographic key generation from voice (extend. abst.). In *Proc. of the 2001 IEEE Symp. on Secur. and Privacy*.

Monrose, F., Reiter, M. K., and Wetzel, S. (2002). Password hardening based on keystroke dynamics. *Internl. J. of Info. Secur.*, 1(2):69–83.

Shamir, A. (1979). How to share a secret. *Comms. of the ACM*, 22(11):612–613.

Simmons, G. (1990). How to (really) share a secret. In Goldwasser, S., editor, *Adv. in Cryptology–Proc. of CRYPTO'88*, volume 403 of *LNCS*, pages p390–448. Springer-Verlag.

Soutar, C. and Tomko, G. J. (1996). Secure private key generation using a fingerprint. In *Cardtech/Securetech Conf. Proc.*, volume 1, pages 245–252.

The GNU Project (1988–2005). The GNU compiler collection. `http://gcc.gnu.org`.