Cryptographic Approaches to Privacy in Forensic DNA Databases

Philip Bohannon[1], Markus Jakobsson[1], and Sukamol Srikwan[2]

[1] Bell Laboratories, Murray Hill NJ 07974, USA,
{bohannon,markusj}@research.bell-labs.com
[2] Chulalongkorn University, Bangkok 10330, Thailand
sukamol@mail.sc.chula.ac.th

Abstract. Advances in DNA sequencing technology and human genetics are leading to the availability of inexpensive genetic tests, notably tests for individual predisposition to certain diseases. While such information is often valuable, its availability has raised serious concerns over the privacy of genetic information. These concerns are further heightened when genetic information is gathered into databases. We study access control for one class of such databases, forensic DNA databases, used to match unknown perpetrators against groups of potential suspects – usually convicted criminals. Our key observation is that for legitimate forensic queries, the sensitive information belonging to the target individual is already available to the querying agent in the form of a blood or tissue sample from a crime scene. We show how forensic DNA databases may be implemented so that only legitimate queries are feasible. In particular, a person with unlimited access to the database will be unable to extract information about any individual unless the necessary genetic information for that individual is already known. We develop a general solution framework, and show how to implement databases which handle certain cases of missing or incorrect DNA tests. Our framework and techniques are applicable to the general problem of encrypting information based on partially known or partially correct keys, and its security is based on standard cryptographic assumptions.

1 Introduction

It is becoming increasingly common that law enforcement agencies collect genetic data from individuals who have been convicted of felonies or certain misdemeanors [12,14,21]. This data is being organized into regional databases, and in the United States, the FBI Laboratory's **Co**mbined **D**NA **I**ndex **S**ystem (CODIS) [26] allows pooling of locally collected information to support nationwide searches. Upon collection of biological materials from the scene of a crime, such a database can be used to identify an individual from the database with almost complete certainty, a so-called "cold hit." This technique is particularly useful with violent and/or sexual assaults by an unknown perpetrator, and over 200 such "hits" have been made to date [13]. However, there is significant concern by civil rights and privacy groups that collection of DNA data into databases

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 373–390, 2000.

can lead to violations of privacy [8,20,30,32]. A particular concern is that the existence of the database may make it possible to discriminate *genetically* against individuals appearing in the database. For example, an insurance company with access to a database of genetic information might decline to insure an individual who is at greater risk for a certain disease [2,20,31]. While expansion of DNA databases is valuable from a law-enforcement perspective, the privacy concern would be aggravated if DNA samples were demanded from a wider population, e.g., everybody who applies for a visa or drivers license, all individuals at birth, etc.[1]

Our goal is to design forensic DNA databases which are infeasible to use, even by the owners of the database, for violating genetic privacy but remain relatively easy to use for legitimate law-enforcement purposes. For example, it should be infeasible for an attacker to determine information about a person's genetic makeup from information such as name or social security number. Similarly it should be infeasible for an attacker to create a list of persons who exhibit a certain genotype for a certain test. By contrast, the typical *legitimate* use of a forensic DNA database is to determine the identity of an unknown person who has left biological evidence at the scene of a crime, and this should remain an effective operation to perform. We formalize this problem below.

In fact, protection of private information in a database is a common cryptographic problem, and is normally solved by encryption methods and/or by sharing the information between some set of servers, such that a threshold number of them have to cooperate in order to release the information. As an example of the latter technique, the CODIS system identifies individuals in the centralized database only by the lab where their sample was processed and a lab-specified identifier of that sample. Thus, collusion with the labs would be required to associate a profile with a particular individual [37]. However, long-term trust in government agencies to act as guards of citizen's private information is inherently problematic, and the expansion of the current database has raised a number of concerns [13,29]. By contrast, we propose the use of encryption techniques to ensure that only legitimate queries are feasible. Thus, the database would remain secure even in a worst-case scenario in which it enters the public domain, along with user passwords, etc. In this approach, access to the database is controlled by treating the results of a set of DNA tests from an individual as the *key* used to decrypt information revealing the individual's identity. (DNA tests, discussed in the next section, can be thought of as revealing a few bits of information about an individuals genetic makeup.) What makes this approach particularly appealing for use with databases of forensic DNA data is that the information used as a key is exactly the information for which privacy is a concern. Consequently, the only agents who are allowed access to an individual's record are agents who already possess detailed genetic information about that

[1] For example, police in Britain have collected samples voluntarily from all male residents in a village to successfully reduce the suspect pool in a highly publicized crime [8]. A similar approach recently ended in the resolution of a German murder case [17].

individual. For example, the agent may possess a blood or tissue sample from the individual. If the agent in possession of such a sample is an attacker attempting to violate the individual's genetic privacy, searching the database is not helpful since the biological sample may be tested directly to much greater effect. It is in this sense that we claim that genetic information cannot be leaked from our database.

The problem space we consider is characterized by the number and cost of tests available, the accuracy of those tests, and the variation in tests used for forensic purposes in different locales. Since the science of genetic testing is progressing rapidly, we do not make fixed assumptions about these issues. Instead, we present a framework database implementation, and extend that framework with three different "plug-ins" to produce three concrete schemes. The first of these schemes, the *fixed-query solution*, is applicable when individual tests are exceptionally accurate, and the set of tests to be used for forensic searches is globally standardized. This scheme is simple, using a hash function modeled as a random oracle [3] to create an encryption key from an individual's test results. Our second scheme, *fixed-query with error correction*, addresses the issue of test errors. While in this domain false positives can be corrected with additional tests once a suspect is identified, false negatives have a very high cost, possibly allowing a violent criminal to remain uncaught. This scheme allows correction of a small number of testing errors by adapting recent *fuzzy commitments* techniques developed for use with biometric passwords [15]. Our third scheme, the *flexible-query solution*, addresses the situation where different, overlapping sets of tests may be available. Thus, the set of tests used to enter one individual may differ from the set used for another individual, neither of which may exactly match the set of tests available for analyzing a particular crime-scene sample. (However, in all cases, the set of tests used on a sample from a crime scene must significantly overlap the tests used on individuals in the database.) The use of differing sets of tests may arise simply due to continued evolution of testing technology, or due to differing standards in different countries. This solution is based on secret sharing [35], and also allows limited correction of errors if extra tests results for a crime-scene sample are available.

For such schemes to be employed in practice, sufficiently many tests must be run on each individual so that a key constructed from these tests can be resistant to exhaustive search. Modern and easily automated tests (see Sect. 2) seem to produce around 5.2-6 bits of entropy per test. Currently, the CODIS system specifies 13 tests [5], leading to up to 67-78 bits of entropy, which is more or less sufficient for our schemes. However, it is not clear if all tests are being used in all states, or if all tests produce 5 bits of entropy. Due to dramatic, on-going improvements in automatic testing technology (see, e.g. [34]) as well as the existence of thousands of potential locations on the human genome for new tests,[2] we assume in this paper that the requisite number of tests for our security parameters will be widely available in the reasonably near future.

[2] These locations have been identified as a by-product of the Human Genome Project [9,34].

The organization of the paper is as follows: an overview of forensic DNA databases follows in Sect. 2, followed by a discussion of related work in Sect. 3. We introduce the formal model of the problem space and our correctness requirements in Sect. 4. In Sect. 5, we present schemes to address several points in the problem space. Security is argued in Sect. 6. We present our conclusions and discuss various open problems and future work in Sect. 7.

2 Overview of Forensic DNA Techniques and Databases

In this section we introduce terminology and give a short overview of forensic DNA testing and data banking technology. More complete introductions are widely available (see, for example, [16,24]). A *DNA profile* is the combination of DNA test results from a particular individual. A *DNA test* reveals information about the DNA sequence present at a particular location, or *locus*, on the genome of that individual. The possible values present at a locus are referred to as *alleles*. For forensic tests, the alleles are usually expressed as the number of times a known DNA subsequence repeats at a given locus. Since human DNA consists of paired chromosomes, the term *genotype* is used to refer to the combination of alleles present at a particular locus on the two paired chromosomes. For privacy, an important attribute of a locus is the role played by the region of which the locus is a part in the biological function of an individual. In particular, regions of DNA which *code* for particular proteins are referred to as *genes*, and other regions are referred to as *non-coding regions*. In summary, a DNA test result reveals the genotype at a locus in either a coding or non-coding region, and a DNA profile is a set of such test results for an individual.

Human DNA profiles may be matched for a wide variety of reasons, including paternity tests and for identifying remains in wartime, as well as for forensic purposes. As described in the introduction, forensic DNA profiles are used primarily to match blood or tissue sample from a crime scene with that of a suspect from the database. (Other forensic uses include connecting unsolved crimes and identifying victims.) When such a match is used to prosecute a crime, statistical methods and databases of allele frequencies are used to obtain conservative estimates of the probability that the match is significant. A match is *significant* if it is due to the sample originating with the suspect rather than the random event that the profile for the suspect matches that of an unknown donor. For some sites, allele frequencies vary significantly with different ethnic populations, so the above estimates must take into account the reference population from which the forensic sample was derived (see [24]). In fact, the significance of a match on a set of tests is related to the security of an encryption scheme based on those tests – the latter issue is discussed further in Sect. 4.3.

Clearly, a locus with a wide variety of possible alleles is preferable to ensure that significant matches can be obtained with a minimum number of tests. When forensic DNA tests became available in the early 1980s, the technology used was relatively slow and expensive, and a significant amount of the blood or tissue sample left at a crime scene was consumed by each test [24]. While

a single test could distinguish a relatively large number of alleles (up to a few hundred), results were approximate, further complicating the calculations described above. With advances in molecular biology, automated procedures allow DNA to be extracted and "amplified" from a minute quantity of blood or other tissue (for example from saliva on a cigarette butt), allowing tests to be repeated at different labs to improve confidence. Furthermore, the results of these tests are discrete and estimation is not required. However, each test produces fewer alleles, requiring more tests to ensure significant matches. The evolution of testing technology has also had implications for the privacy risk of forensic data. While older tests were drawn exclusively from non-coding regions, modern tests are sometimes from genes [20,24]. Further, regions similar in structure to those used for testing have been determined to be correlated with certain diseases [18,36,38]. While loci currently used for forensic DNA testing are not known to be correlated with diseases, there is no guarantee that further study will not lead to the discovery of just such a correlation. Thus, any substantial forensic DNA database represents the risk that it may be possible in the future to use the database to correlate individuals with a genetic disease or some other biological trait. When designing the database, we therefore make the pessimistic assumption that *all* recordable information about an individual's DNA sequence should be kept private.

As mentioned in the introduction, several countries are creating forensic genetic databanks [6,28], and almost all states across the USA are creating DNA databanks using the FBI's CODIS system [26]. CODIS stores information in two indexes: 1) The Convicted Offender Index, which contains DNA profile of individual convicted of crimes, and 2) the Forensic Index, containing DNA profiles developed from crime scene specimens. Each index stores a specimen identifier, the laboratory identifier, names of laboratory personnel in charge of the DNA profile, and the DNA profiles. Information such as case-related information and social security numbers are not included in CODIS. When CODIS identifies a potential match, the laboratories responsible for the matching profiles contact each other to validate or refute the match. After a match has been confirmed by qualified DNA analysts, laboratories may exchange additional information including the identity and location of the identified suspect. More than forty states have passed legislation that requires convicted offenders to provide samples for DNA database. These states have collected approximately 450,000 DNA samples, and analyzed over 150,000. However, a review of DNA data-banking laws observed that little attention is paid to issues of genetic privacy in most of the state legislation [21].

3 Related Work

We consider how to safeguard a database against abuse. Here the somewhat imprecise term "abuse" corresponds to reading private information about individuals in the database when some "key" information has not already been gathered, in this case as evidence, by the querying agent. Our concept allows

the holder of the database to obtain the answer to the query after inputing some record-specific and identifying pieces of data. It can be used for any type of database where knowledge of some data (of sufficient size) is required for a lookup to be possible. In our setting, this "key information" is the results of some fixed forensic tests.

Two recent papers that are related in a technical sense are those of Monrose, Reiter and Wetzel [23] who used similar methods for strengthening login sessions by measuring typing speeds; and Ellison, Hall, Milbert and Schneier [10], who suggested a method for "fuzzy" passphrase based logins, where only a certain portion of the passphrase needs to be correct. Whereas the former employs techniques believed to be resistant against lattice-based attacks (see e.g., [7,25,33]), the latter has been found to be susceptible to such attacks, as recently pointed out by Bleichenbacher [4]. Note that in both cases, the technology is in some sense used to relate attributes of an individual (typing habits and personal information, respectively) to the identity of that individual.

Our solution is in principle related to that of Monrose, Reiter and Wetzel, and one of our proposed plug-ins uses a similar structure to embed secret information. It differs on a few important points, and to understand these, we will briefly review how the MRW scheme for password hardening works. The scheme is based on a password, a secret sharing scheme such as polynomial interpolation, and a two-dimensional matrix with two columns and some number of rows, where each entry may contain a point useful for interpolation or a random number. Each row corresponds to some measurement of how the user typed a letter of the password. For example, one column might corresponding to the letter being typed "quickly" after the previous letter, while the other column corresponds to it being typed "slowly" (as compared to some average speed). The idea is that individuals routinely type *certain* letters in their password either quickly or slowly. When an individual routinely types a letter slowly, a share of the secret is entered into the first column, and similarly for the second column if the user routinely types that letter quickly. In either case, the other entry in the row contains a random number. If an individual does not have a routine behavior for a measurement corresponding to a particular row in the table, shares of the secret can be entered in both columns of that row. Once the table is filled, each entry in the table is further encrypted with the user's password.

This table is used to harden the user's password as follows. While the user enters his password, the system measures the time between and duration of keystrokes. If a measurement falls below a certain threshold for the ith keystroke, value $(i, 1)$ from the matrix is selected; otherwise, value $(i, 2)$ is selected. Each such selected value (the number of which equals twice the length of the password, minus one) is decrypted, using the entered password as the decryption key. Then the values collected are used to determine the secret of the secret sharing scheme. For example, when using polynomial interpolation for secret sharing, the values are interpreted as points which can be thought of as lying on a polynomial. These points are interpolated to determine where the polynomial intersects the y-axis, and the resulting value is treated as a key. Using this key, a file (containing

keystroke statistics) is decrypted. If the result is a file of the correct format (which can be obtained by introducing some redundancy), the login session is said to succeed. The measurements taken of the user's typing speeds are then added to the statistics in the file. A new table and polynomial are created based on the modified statistics, and the statistics file is encrypted with the key from the new polynomial and again stored in the user profile.

The MRW technique shares with ours the use of personal information (typing patterns vs. genetic profiles) to allow access to some resource or data, the mapping of features of personal information to points on a polynomial, and the use of polynomial interpolation as a secret sharing scheme in the decryption process. However, the situations differ in two principal ways: 1) All of the personal information used in the MRW scheme is available at each use since the user must type every key of the password. In one of our proposed solutions, substantial portions may be missing, and the scheme works as long as a sufficient amount is known. 2) The bulk of the entropy in the MRW scheme resides in the password, and the personal information used is related to a particular password. Thus, the key can easily be reset by choosing a new password. In our case, the key is genetic information which cannot be reset, and thus must remain secure for the lifespan of individuals in the database.

A third recent paper of relevance is that of Juels and Wattenberg [15], in which they put forth a method for hash functions whose inputs can contain a certain number of errors without changing the output. However, if the errors exceed a certain threshold, the resulting output is unpredictable, as for standard hash functions. We show how to employ this method as a building block in one of our proposed schemes, and discuss it further in that context.

4 Model and Problem Definition

In this section we present a trust model for a forensic DNA database, and define the problem of implementing such a database in a secure manner. Our definition hinges on two parameters: (1) the number of tests required for a lookup and (2) the number of such tests that may be in error. We also give an overview of our correctness criteria, and discuss the issue of how many DNA tests would actually be required for such a scheme to be secure.

4.1 Trust Model

In order to perform the tests required to enter an individual into a forensic DNA database, it is necessary to obtain a blood or tissue sample from the individual, and to submit that sample to a government or commercial lab for genetic testing. Later, when a lookup is to be performed, similar testing has to be performed on the blood or tissue sample for which a lookup is to be made. In our model, we have to trust that the labs performing these tests are honest, and that almost all of the test results are correct. Additionally, we trust the labs not to run inappropriate tests on the samples, not to reveal information that would compromise

the privacy of the individuals from whom the samples were taken, and to discard samples and test results after their test results have been reported. It is possible to conceal the identity of an individual from a lab (that does not violate the above assumptions) simply by using an alias identifier instead of a personal identifier to label the samples. This is already commonly done in genetic testing so that control samples may be used to verify the accuracy and integrity of the lab processes [24].

In addition, we have a database producer whose task it is to administer the collection of blood or tissue samples, send these to the above mentioned labs, collect the results, and create database entries. This database producer may consist of a multitude of independent entities, each responsible, for example, for a certain geographical region. We trust each such entity not to enter incorrect data, and not to violate the privacy of the individuals in the database.

4.2 Problem Definition

In the following, T represents the trusted authority described above, and h is a hash function that can be described as a random oracle [3]. The goal of an attacker in the following discussion is to obtain some plaintext information from some record in the database. (We do not require that the attacker succeed in decrypting an entire entry, but merely that some non-trivial information is obtained about some plaintext record.)

A problem instance is characterized by three parameters, k, c and s, and a set of n different tests $\tau_1, \ldots \tau_n$. The parameter k refers to the number of test results (out of n) which must be provided as part of a query, c refers to how many of these test results may be in error, and s is a security parameter such that 2^{-s} is the maximum success probability of an adversary. The test results τ_j are typically small integers; for simplicity we denote the range of possible outcomes of test τ_j as $[1, \ldots, \omega_j]$. Let M_i be a plaintext message describing the individual with index i. For example, M_i might be a social security number, a reference to an entry in some database of convicted offenders, or a lab name and lab-assigned individual identifier as in the CODIS system. A record in the database is comprised of an encrypted version of M_i, possibly along with certain additional information specific to a particular problem solution.

A query is a set of test results from an unknown individual u. While the test results may be for any k-subset of the n tests in the system, we refer to them as $\tau_{u1}, \ldots, \tau_{uk}$ for denotational simplicity. A query result is a plaintext M_u if a match is made, and a string stating "no match" otherwise. A scheme is said to be *secure* if it meets the following requirements:

- **Correct.** When at least k test results are presented as part of a query, if all the computation is performed as stated in the protocol description, the query result is M_i for an individual i whose test results match the query values on all but at most c tests, but for a negligible probability.
- **Complete.** All the required computation can be performed in time polynomial in the length of the security parameters.

– **Hiding.** It is impossible to extract data from the database without prior knowledge of test results of the individual for whom the lookup is to be performed.

4.3 Parameter Choices

In this section, we discuss the selection of k. We do not assume that the tests are identically distributed, although we do assume that they are independent (the case where tests are correlated can be handled with obvious extensions). In all cases, the minimum entropy present in some $k - c$ tests supplied as part of a query, and not used for error correction or indexing, equals the security parameter s.

We have computed the test-result entropy present in frequency distributions given in [11] for two loci, `d21s11` and `HumFGA`. The entropy was computed using the formula $E = -\sum_{i=1}^{\omega_j} f(i) \cdot \log_2 f(i)$ where $f(i)$ is the frequency with which the test takes on allele value i. Since some alleles are more common in certain sub-populations, frequency distributions for tests are computed independently for the different ethnic groups in [11]. Assuming that an attacker may know the ethnic group of an individual, we conservatively take the *minimum* of the computed sup-population entropy values as the entropy of the test. For these two tests, and assuming independence of the two alleles, the minimum entropy for any of the given ethnic groups is 5.3 and 5.24 bits respectively. If we assume this is representative for sites used in modern test protocols, and that a security parameter of 80 bits of entropy is desirable, then approximately 80/5.24, or a minimum of 15 tests, would be required.

5 Solutions

We begin by presenting a general solution framework, and then show how the scheme can be specialized for three different cases. The solution corresponding to the first case is the simplest and easiest to analyze. However, it assumes that individual tests produce accurate results and that the set of tests is fixed and unchanging. The second solution is almost as efficient, and allows for some errors to be corrected using standard error correcting methods. Finally, the third solution, which bears resemblance to the MRW solution, allows for the query set of tests to differ from the set of tests originally run on the individual, as long as the intersection of these sets contains k or more of the n tests. This solution also allows for limited error correction when more than k tests are available.

5.1 Solution Framework

Our database is produced by the trusted authority T, and then sent to individual control offices, such as local or regional police headquarters. In the following, E_{K_i} is a symmetric encryption function with key, K_i, chosen uniformly and at random

from a large keyspace $\mathcal{K}$. The cipher is assumed to have the property that given (only) one ciphertext and a potential plaintext, it is not possible to determine if these correspond to each other without knowledge of the key. This property relates directly to the unicity distance of the cipher (see [22] for an overview).

Each record in the database consists of a form of M_i encrypted with E_{K_i} (discussed further below), and a "lock" L_i. K_i may be randomly generated or may be computed from from the tests, τ_{ij} during initial record creation. Later, when attempting to determine if a query set matches this record, a "candidate" $\tilde{K}_i$ is computed, making use of the test results in the query set and the lock, L_i.

Creating a New Record.

1. **Identification.** For each new individual, indexed by number i, the trusted authority T obtains a blood or tissue sample, and performs the n tests on the sample, determining the outcome τ_{ij}, $1 \leq j \leq n$, of each such test.
2. **Secret key and lock generation.** T generates a "lock" L_i, and a key $K_i \in \mathcal{K}$. Here, K_i can (only) be computed given the lock L_i and the test values τ_{ij}, which act to "unlock" L_i. The manner in which these parameters are generated is scheme-specific, and will be detailed in Sect. 5.3.
3. **Data encryption.** Let M_i be a plaintext message describing the individual with index i. Let $\mu_i = M_i | red(M_i)$ for some fixed redundancy function red. T computes the ciphertext $c_i = E_{K_i}(\mu_i)$ of μ_i. T further computes L_i in a scheme-specific manner.
4. **Record creation.** T appends the record $R_i = (c_i, L_i)$ to the public database, erases all intermediary values, and discards the remainder of the blood or tissue sample.

We note that as new database entries are created, these records can simply be appended to the existing database.

Searching the Database.

1. **Identification.** Say that an agent, A, wants to look up a given individual in the database. A obtains a blood or tissue sample, presumably left at the scene of a crime, from an unknown person u. A has k of the n determined tests performed on the sample, say tests 1 through k for denotational simplicity, giving results $\tau_{u1}, \ldots, \tau_{uk}$. These results comprise the query set for the database.
2. **Data decryption and output.** For each record $R_i = (c_i, L_i)$ of the database, the agent computes $\tilde{K}_i$ in a scheme-specific manner using L_i and $\tau_{u1}, \ldots, \tau_{uk}$, and attempts to decrypt the corresponding ciphertext, c_i, using $\tilde{K}_i$. The result is $\tilde{\mu}_i = D_{\tilde{K}_i}(c_i)$. If this has the correct form (i.e., is of the form $M_i | red(M_i)$) then A outputs $(i, \tilde{K}_i, M_i)$. With overwhelming probability, $i = u$ when a match is made. If no record results in a transcript of the correct form, then A outputs "no match" and halts.

5.2 Indexing

As discussed so far, the scheme requires that each record in the database be scanned during a lookup. While a match against a specific record will be very fast compared, say, to a fingerprint match, it remains worthwhile to consider how to index the database for quicker lookups. One possibility of indexing would be to separate the test results into two portions. The first portion would be set aside for encryption and matching, as previously explained, whereas the second portion would be used to generate an index of the records to be created. Given test results, we would select the database entries whose indices are closest to that generated from the results. The second portion could simply be left as plaintext, as it does not reveal any information about the owner, as long as the identity of the record owner remains unknown.

We note that this technique allows a tradeoff between the cost for performing experiments and the computational expense, as we can narrow down the search by determining partial information about the index. However, by keeping the result of additional tests in plaintext, it becomes important that it not be possible to match records in the database with individuals. For example, an attacker might attempt to correlate new records appearing in the database with known convictions in a given area. This possibility can be mitigated by batching updates, or by padding small updates with false records, etc.

5.3 Three Plug-ins

We consider three ways to construct the building block referred to as "the lock" in the above described scheme.

Fixed-query Solution. In our first implementation, we assume that the tests $\tau_1, \ldots, \tau_n$ are fixed and *universal*, meaning that the same set of tests will always be available whenever entries are created or samples tested from crime-scenes. We assume further that almost no errors are introduced when performing the tests. In this solution, we do not need any lock L_i, but plainly set the key K_i (resp. $\tilde{K}_i$) as an appropriately long portion of $h(\tau_{i1}, \ldots, \tau_{in})$ for the n tests performed. Assuming that the hash function can be modelled as a random oracle, it is sufficient that our tests have a sufficient entropy (say 80 bits) for the resulting scheme to be secure. We note that this solution generalizes directly to a small number of distinct *sets* of standard tests, corresponding, for example, to standard sets of loci used in different countries. This can be achieved plainly by computing several such keys, and creating several records correspondingly.

Fixed-Query Solution with Error Correction. This technique incorporates the use of *fuzzy commitments* from a recent result by Juels and Wattenberg [15] to allow some portion of a key to be in error and yet be successfully used to decrypt a codeword. Recall that error correction codes correct messages[3] to the

[3] In this context, "message" refers to the key which may be in error, not to the plaintext object being encrypted/decrypted.

nearest *codeword*. The set of messages which will be corrected to a codeword is by definition the *neighborhood* of that codeword. Without reference to a particular error-correcting code, we outline how the technique from [15] may be used to encrypt some object μ_i, based on some "user key," U, such that any $\tilde{U}$ can be used to decrypt μ_i as long the difference between $\tilde{U}$ and U is small – within the power of the error correcting scheme to correct. In general, U can be thought of as biometric information such as a fingerprint – in our setting, U would be the $\tau_{i1}, \ldots, \tau_{in}$.

The first step of encrypting μ_i based on U in this scheme is to select a codeword, σ_i, at random from a set of codewords of the same length in bits as U in the chosen error-correcting code. The next step is to compute the *bitwise difference*, δ_i, between U and σ_i. An appropriate portion of $h(\sigma_i)$ is then used as K_i, and μ_i is encrypted using E_{K_i} to form c_i. The value, δ_i is stored as plaintext with c_i.

In order to later decrypt c_i with a candidate key, $\tilde{U}$, δ_i is subtracted in a bitwise manner from $\tilde{U}$. Error correction is applied to the resulting $\tilde{U} - \delta_i$ value to generate $\tilde{\sigma}_i$, the closest codeword under the error correction code. If the difference between U and $\tilde{U}$ is small enough, this correction will succeed giving $\tilde{\sigma}_i = \sigma_i$. The candidate key, $\tilde{K}_i$, is set as an appropriate portion of $h(\tilde{\sigma}_i)$, and an attempt is made to decrypt c_i with $\tilde{K}_i$. We can use this construction in our scheme, setting $L_i = \delta_i$.

Flexible-Query Solution. This technique adapts secret sharing techniques [35] to allow a record to be decrypted with any number of tests missing, as long as sufficient tests are presented for the security of the scheme. The idea is that the lock, L_i allows test results to be mapped to points on a polynomial of degree k, allowing interpolation of K_i if at least k points are presented.

In this solution, p and q are large primes such that $q|p-1$, and g is a generator of a subgroup G_p of size q. Consider first the generation of the key K_i and the lock L_i on record creation:

T generates a random k-degree polynomial $\rho_i(x)$ with coefficients in Z_q. T further computes a set of n points, $s_i(j) = \rho_i(h(j, \tau_{ij})) \bmod q$, $1 \leq j \leq n$, and a second set of n points, $P_i(j) = g^{s_i(j)} \bmod p$. The lock, L_i is set to $P_i(1), \ldots, P_i(n)$. K_i is computed as an appropriately long portion of $h(g^{\rho_i(0)} \bmod p)$.

In order to perform a look-up, $\tilde{P}_i(0)$ is extrapolated using the measurements τ_{uj} and the points $P_i(j)$ on the polynomial as follows: Let $x_{uj} = h(j, \tau_{uj})$ be the x coordinate of such a point, and $\mathcal{X}$ the set of such coordinates, where this set is of size k. The interpolation is computed as $\tilde{P}_i(0) = \prod_{j=1}^{k} P_i(j)^{\lambda_{ij}} \bmod p$, where λ_{ij} is the Lagrange weight for the coordinate x_{ij} in $\mathcal{X}$. Finally, $\tilde{K}_i$ is computed as the appropriate portion of $h(\tilde{P}_i(0))$.

Discussion. Figure 1 illustrates a portion of the construction of K_i and L_i. The idea is to use the secret of a secret sharing scheme [35] as the key K_i. By mapping test results to shares of the secret, we allow decryption of the record

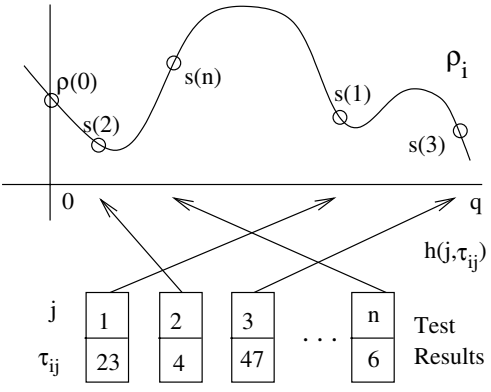

Fig. 1. Relationship between ρ_i, τ_{ij} and $s_i(j)$

if and only if sufficient tests are present. In this case, the secret sharing scheme is based on interpolation of a discrete polynomial, ρ_i. As shown in the figure, h is used to map (test name, test result) pairs into points on the polynomial. One possibility would be to use these $s_i(j)$ values as the lock L_i. If this were done however, lattice based attacks [33] might be used to determine ρ_i based on the limited set of points which might be generated from the test results. By using h to perturb the location of the points on the x-axis, such attacks are made more difficult. However, the final step of our lock construction, which uses $P_i(j) = g^{s_i(j)}$ in the lock rather than $s_i(j)$ itself, provides much greater protection by destroying the homomorphic properties of the system – for example, it is not possible to compute a representation of the product of two s_i values when they are represented as exponents of g, if the Diffie-Hellman assumption holds.

It is clear that this scheme allows flexibility in which tests are used, since k can be less than n, so long as any k tests contain at least s bits of entropy (recall that s is the security parameter of the problem instance). This is in contrast to the previous two solutions which require $k = n$.

Error Correction. In order to allow for error correction in this scheme, we can try different subsets of k test results for a match. We note that this solution is worst-case exponential in the number of errors present, but can be used for some low number of incorrect tests, particularly if indexing is employed (as this narrows down the search space substantially), and is much faster than trying every possible *value* of an incorrect test.

Remark Concerning Entropy. We note that in the last solution, we want the different tests results to have similar entropy, as an attacker only needs to successfully guess k of these, and the success rate of such an attack relates only to the entropy of the tests being guessed. If the tests have substantially different entropy, we can split test results with large amounts of entropy into several log-

ical tests, causing such logical tests to have similar entropy. (Clearly it may not be possible to utilize all the entropy present in all tests if doing so.)

6 Security Analysis

It is clear that our scheme is *correct* (i.e., that the correct output is obtained) and *complete* (i.e., that all computation can be performed.) Moreover, our scheme is *hiding*, i.e., it is impossible to extract data from the database without first performing a sufficient number of tests on tissue or blood. More formally,

Claim: Assume that K_i is set as an s-bit number chosen uniformly at random. Let $\mathcal{A}$ be an adversary attempting to decrypt a ciphertext c encrypted with the encryption method E, using this unknown key K_i, which is only used for the creation of c. Then, our scheme is hiding against $\mathcal{A}$, or $\mathcal{A}$ can be used as a black box algorithm to break the assumed semantic security of E.

This follows straightforwardly from the definitions of the related concepts. The security of our scheme follows from the above, given that the hash function h can be modeled as a random oracle, and the record lock L_i does not leak any information that can be used to compute the key K_i. (The lock is the only auxiliary data related to the key K_i, under the assumption that there is no correlation between the test results used for the lock creation and those used for indexing.)

It is clear that our solution, if employing the first plug-in, is secure, given that we use no lock in this plug-in. In the second solution, it is clear that bits proportional to the the distance between codewords in the error correcting code may be leaked, since the lock, while not revealing the codeword to which the user key is mapped, may reveal the user key's relative position to the nearest codeword. The precise amount of leakage, and thus a precise analysis of the security of the scheme, is dependent upon the particular error correction code chosen, and the selection of an appropriate code for genetic tests is a topic of ongoing research. Further discussion of the scheme's security may be found in [15]. Whereas we know of no proof of security for our third construction, we note that this solution avoids the properties employed by lattice-based attacks [33], as it avoids linear relationships between the items used for interpolation.

6.1 Attacks with Biological Knowledge

One might argue that a potential attacker with access to other databases of genetic information would be able to perform a successful search in the database we are designing. After all, if a database with complete genetic information about all human beings (of which there are only about 2^{33}, an insignificant number in comparison to the security parameters of the scheme) is available to the attacker, this data can be used to limit search of the key space to only existing combinations, allowing decryption of all of the information present in our database. Such an attack is analogous to dictionary attacks on user passwords.

In general, we consider the possession of such a database by an attacker to be in itself a tremendous risk to the genetic privacy of the individuals whose information appears therein. It is unlikely that the additional ability to decrypt a forensic database would add significantly to that risk. However, one might argue that an *anonymized* genetic database would not constitute such a risk to individual privacy. If such an anonymized database included the test loci used in a forensic database organized as we propose, the forensic database would serve exactly to reveal the identities associated with records in the database. In order to mitigate this risk, loci used for forensic purposes should simply be left out of any such large-scale anonymized database.

Remark - the Icelandic Database. In fact, just such a database is being constructed in Iceland for the purpose of medical genetic research [27], and is the subject of significant controversy on exactly the issue of genetic privacy (see, for example, [19]). We argue that this database presents a far greater risk to individual genetic privacy than is posed by publication of any present or conceived forensic DNA database, simply because, as already mentioned, the data stored in forensic databases is not known to relate to any individual attribute. By contrast, the data to be gathered in the Icelandic database is exactly related to genetic disease, so that the database will be useful for medical research. Regarding the possibility of using a forensic database to destroy the anonymous property of a database such as the Icelandic one, it was pointed out in [1] that many other avenues exist for determining the identity of individuals in such a database, using features such as the number and gender of relatives.

7 Conclusions and Future Work

In this paper we introduce a cryptographic approach to protecting the privacy of information in forensic DNA databases. Our solution allows the recovery of an individual's identity and/or criminal record if – but only if – the genotypes present at a certain number of loci on the suspect's DNA are determined. Based on differing assumptions about standardization of forensic genetic tests and the need for error correction, we have proposed three solutions to this problem within a common framework. In our first, fixed-query solution, tests are completely standardized and no error correction is required. Our second solution extends the first to add error correction using fuzzy commitments from [15]. Our third solution employs polynomial interpolation to allow decryption based on a threshold number of tests, allowing arbitrary tests to be missing. In each case, a "lock" is stored with each record which, when combined with test results for the individual, allows the individual's identity to be decrypted. In each case, if the requisite loci are not identified, it is not feasible to identify or decrypt the record in question. Given the assumption that deriving the threshold number of values can only be accomplished by testing a sample of biological material from the individual, it is not feasible to derive any genetic information from the database which is not already available in the sample.

Based on preliminary computations about the entropy present in modern tests, current profiles may not contain sufficient entropy for achieving sufficient security, implementing error correction, or adding indexing. However, with the rapid introduction of new sites and loci, it is reasonable that a profile with over 100 bits of entropy would be easily available, or may in fact already be available at the time of publishing.

While we argue that these techniques are practical from a computational perspective, and will soon be practical from a genetic testing perspective, certain limitations remain that are the topic of ongoing research. First, it may not be practical to perform all possible tests against each individual. In itself, this is not a problem, unless the amount of biological material available in a sample is so small as to strictly limit the number of tests which may be performed on the sample. In this case, it would be useful to know which test to run next given the output of previous tests. For example, given the result of a "universal" test given to all individuals, one would like to run subsequent tests which would help distinguish between those records in the database which matched the sample on this test. Unfortunately, it is not obvious how to do this without revealing secret information. Second, there are certain searches which may be performed on an unencrypted database which may not be performed on our database. One such search is for a close relative of an individual from which a sample is obtained. In the case of a parent or child, one would expect about half the tests to match, which may provide useful leads in a case. Another such search takes place when a blood sample contains a mixture of blood from two individuals, where the identity of neither individual is known to the investigators. In this case, exact values of test results are not known, since each test returns results for both individuals.

We note that it may be possible to reuse the information of the offset described in the second plug-in for purposes of indexing. We have not examined how to practically do this.

Finally, we note that the general form of the problem definition and our solutions apply to any problem domain where a *secure* but *imprecise* key is required. These problems include the use of biometric data (fingerprints, retinal scans, etc.) [15] and the generation of keys which can be effectively recreated by human beings [10].

Acknowledgements. We thank Daniel Bleichenbacher and Phong Nguyen for helpful discussions on lattice based attacks. We thank Ari Juels for helpful general discussions, and Amin Shokrollahi for explanations of error correcting codes.

References

1. R. Anderson.: The DeCODE proposal for an Icelandic health database. `http://www.cl.cam.ac.uk/řja14/iceland/iceland.html`, 1998. 387
2. G.J. Annas.: Privacy rules for DNA databanks: Protecting coded 'future diaries'. *JAMA*, 270(19):2346–2350, 1993. 374

3. M. Bellare, P. Rogaway.: Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communication Security*, pages 62–73. ACM Press, 1993. 375, 380
4. D. Bleichenbacher.: Private communication. 378
5. B. Budowle, T.R. Moretti.: Genotype profiles for six population groups at the 13 CODIS short tandem repeat core loci and other PCR-based loci. *Forensic Science Communication*, 1(2), July 1999. 375
6. D. Butler.: UK to set up DNA database of criminals. *Nature*, 370:588–589, 1994. 377
7. M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.-P. Schnorr, J. Stern.: Improved low-density subset sum algorithms. *Journal of Computational Complexity*, 2:111–128, 1992. 378
8. A. de Gorgey.: The advent of DNA databanks: Implications for information privacy. *American Journal of Law and Medicine*, 16:381–398, 1990. 374
9. C. Dib, S. Faure, C. Fizames, D. Samson, N. Drouot, A. Vignal, P. Millasseau, S. Marc, J. Hazan, E. Seboun, M. Lathrop, G. Gyapay, J. Morissette, J. Weissenbach.: A comprehensive genetic map of the human genome based on 5,264 microsatellites. *Nature*, 380:152–154, 1996. 375
10. C. Ellison, C. Hall, R. Milbert, B.Schneier.: Protecting secret keys with personal entropy. *Future Generation Computer Systems*, to appear, 1999. 378, 388
11. R. Fourney.: Allele frequency distribution tables. http://www.cstl.nist.gov/div831/strbase/freq_tab.htm. 381
12. K. Fox.: Criminal justice. In *Mapping Public Policy for Genetic Technologies*. National Conference of State Legislators, 1998. 373
13. C. Goldberg.: DNA databanks giving police a powerful weapon, and critics. New York Times, Thursday 19, 1998. 373, 374
14. B.E. Henry, G.S. Rogers, C. Mauterer, D.K. Dodd, J.W. Hicks.: Technical evaluations of databanking methods. In *Proceedings of the Eighth International Symposium on Human Identification*, 1997. 373
15. A. Juels, M. Wattenberg.: A fuzzy commitment scheme. In *6th ACM Conference on Computer and Communication Security*, (to appear), 1999. 375, 379, 383, 384, 386, 387, 388
16. L.T. Kirby.: *DNA Fingerprinting: An Introduction*. Oxford University Press, 1992. 376
17. R. Köttger.: Probe nummer 3889 führte zum Mörder. Die Welt, August 27, 1999. 374
18. T.G. Krontiris.: Minisatellites and human disease. *Science*, 269:1682–1683, 1985. 377
19. Mannvernd.: The Mannvernd web site. http://www.mannvernd.is. 387
20. J.E. McEwen.: DNA data banks. In M. Rothstein, editor, *Genetic Secrets: Protecting Privacy and Confidentiality in the Genetic Era*, pages 231–254, 1997. 374, 377
21. J.E. McEwen, P.R. Reilly.: A review of state legislation on DNA forensic data banking. *American Journal of Human Genetics*, 54:941–958, 1994. 373, 377
22. A.J. Menezes, P.C. van Oorschoot, S.A. Vanstone.: *Handbook of Applied Cryptography*. CRC Press, 1997. 382
23. F. Monrose, M. Reiter, S. Wetzel.: Password hardening based on keystroke dynamics. In *6th ACM Conference on Computer and Commnication Security*, (to appear), 1999. 378
24. National Research Council.: *The Evaluation of Forensic DNA Evidence*. National Academy Press, 1996. 376, 377, 380

25. P. Nguyen, J. Stern.: The hardness of the hidden subset sum problem and its cryptographic implications. *Crypto'99, LNCS 1666*, pages 31–46, 1999. 378
26. S.J. Niezgoda Jr., B. Brown.: The FBI laboratory's COmbined DNA Index System program. In *Proceedings of the Sixth International Symposium on Human identification*, 1995. 373, 377
27. Working Group of the Ministry of Health and Social Security.: Bill on a health sector database. http://brunnur.stjr.is/interpro/htr/htr.nsf/pages/gagnagr-ensk, 1998. 387
28. E. Peerenboom.: Central criminal DNA database created in Germany. *Nat Biotechnol*, 16(6):510–511, 1998. 377
29. R. Perez-Pena, J. Blair.: Albany plan widely expands sampling of criminals' DNA. New York Times, Saturday, August 7, 1999. 374
30. P.R. Reilly.: DNA banking. *American Journal of Human Genetics*, 51:1169–1170, 1992. 374
31. P.R. Reilly.: Fear of genetic discrimination drives legislative interest. *Human Genome News*, 8:3–4, 1997. 374
32. B. Scheck.: DNA data banking: A cautionary tale. *American Journal of Human Genetics*, 54:931–933, 1994. 374
33. C.-P. Schnorr, H.H. Hörner.: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. *Eurocrypt'95, LNCS 921*, pages 1–12, 1995. 378, 385, 386
34. J.W. Schumm.: New approaches to DNA fingerprint analysis. *Promega Notes Magazine*, 58:12–18, 1996. 375
35. A. Shamir.: How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 375, 384
36. G.R. Sutherland, R.I. Richards.: Single tandem DNA repeats and human genetic disease. In *Proceedings of the National Academy of Science USA 92*, pages 3636–3641, 1995. 377
37. Technical Working Group on DNA Analysis Methods (TWGDAM).: The combined DNA index system (CODIS): A theoretical model. In L.T. Kirby, editor, *DNA Fingerprinting: An Introduction*. Oxford University Press, 1992. 374
38. K. Wrogeman, V. Biancalana, D. Devys, G. Imbert, Y. Trottier, J-L. Mandel.: Microsatellites and disease: A new paradigm. In *DNA Fingerprinting: State of the Science*, pages 141–152, Basel, Switzerland, 1993. Birkhäuser Verlag. 377