© 2007 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Pre-print of article that appeared at CVPR 2007.

The published article can be accessed from:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4270135

Secure revocable fingerprint biotokens

Terrance. E. Boult[1,2, ‡], Walter J. Scheirer[1, ‡] and R. Woodworth[2, ‡],
[1]VAST Lab University of Colorado at Colorado Springs and [2]Securics, Inc
Colorado Springs, CO. lastname@vast.uccs.edu or lastname@securics.com

Abstract

This paper reviews the biometric dilemma, the pending threat that may limit the long-term value of biometrics in security applications. Unlike passwords, if a biometric database is ever compromised or improperly shared, the underlying biometric data cannot be changed. The concept of revocable or cancelable biometric-based identity tokens (biotokens), if properly implemented, can provide significant enhancements in both privacy and security and address the biometric dilemma. The key to effective revocable biotokens is the need to support the highly accurate approximate matching needed in any biometric system as well as protecting privacy/security of the underlying data. We briefly review prior work and show why it is insufficient in both accuracy and security.

This paper adapts a recently introduced approach that separates each datum into two fields, one of which is encoded and one which is left to support the approximate matching. Previously applied to faces, this paper uses this approach to enhance an existing fingerprint system. Unlike previous work in privacy-enhanced biometrics, our approach improves the accuracy of the underlying system! The security analysis of these biotokens includes addressing the critical issue of protection of small fields. The resulting algorithm is tested on three different fingerprint verification challenge datasets and shows an average decrease in the Equal Error Rate of over 30% -- providing improved security and improved privacy.

1. The Biometric Dilemma

The key properties of biometrics, those unique traits that do not change significantly over a lifetime, are also their Achilles heel. **The biometric dilemma is that while biometrics can initially improve security, as biometric databases become widespread, compromises can/will ultimately undermine biometrics' value and usefulness for security.**

At least 40 million "financial records" were compromised or illegally sold in 2005, and over 50 million more financial/identity records lost or stolen in 2006. A database with millions of permanent "non-revocable" biometric records will become more significant cyber-target. With the current trend, it is a question of when, not if, a major biometric database will be compromised. Of course, they don't have to be hacked, just shared or sold. In 2001, Colorado tried to sell their face and fingerprint DB, and still offers it free of charge to any government agency that wants access [Krouse-01].

While many companies say their biometric templates cannot be used to recreate the data, [Ross-05] has shown recovery of fingerprints from templates. Furthermore, reconstructing the original fingerprint data is not required, just generation of any of the infinitely many prints that match the stored template. With techniques that allow generation of "gummy fingers" [Matsumoto-et-al-02], the potential loss is not "academic" and not just an issue of privacy. While vendors claim new "liveness" detection prevents spoofing, the authors have tested many of the new sensors, from optical to capacitive to thermal, and spoofed every one we have tested.

A compromised biometric cannot be "replaced" and that permanent loss feeds the perception of invasion from any use of biometrics – if decades later the government or a corporation wants to play Big Brother, you cannot take back the information they gathered or lost. With the ease with which today's sensors can be spoofed and with the instructions readily available on the Internet, this is a security time bomb!

While many people like to think of biometrics as "unique", operationally they are not. E.g., the best fingerprint systems tested by the US government have only 98% true acceptance rates, when set to reject 99.99% of false matches. Even FBI examiners have made high-profile misidentifications with fingerprints. At 99.99%, finding a false match in a database of millions is likely, leading to what we call the *doppelganger threat*, where compromised databases with millions of users will allow an intrudder to find a few "close enough" matches they can directly impersonate.

No one serious about security would use a password, a card, or a certificate that cannot be changed or revoked. Why expect less of biometric solutions?

A critical issue in the biometric area is the development of a technology that allies the privacy concerns while supporting the security goals. A partial solution is to never store the original biometrics, but some cancelable token generated from it. This concept was

‡ This work supported in part under NSF STTR "Improving Privacy and Security in Biometrics," Award Number OII-0611283

introduced in [Ratha-et-al-01], and called Cancelable biometrics. However, since the underlying biometric data is not actually canceled or revocable, this oxymoron can cause confusion, We introduce the term *biotoken*, to refer to "the revocable identity token produced by applying a revocable transform to biometric data, such that identity matching is done in the encoded/revocable form".

To address the biometric dilemma the biotokens must support a large variety of tokens per individual so that each database is independent and non-linkable and allow effective revocation and reissue. They must have sufficiently high accuracy and must provide for cryptographically strong protection of the underlying data. Even with all of that, they must explicitly have some approach to address the doppelganger threat.

A few approaches for cancelable or revocable biotokens have been discussed in the literature, with a review and classification of them presented in [Ratha- et.al.-07]. They divide the field into four catagories: Biometric salting, Fuzzy schemes, Biometric Key generation and non-invertible forms. Note that our approach does not fit within any of these catagories.

Biometric salting, mixing randomness, has been tried by many groups but has not produced effective systems in terms of accuracy and has the issue that the "random pad" must still be protected since recovery of the original data given the pad is generally easy. All the current papers in this area also require pre-aligned data, an unrealistic assumption.

Fuzzy schemes, the best of which we consider to be [Tuyls-et-al-05], are making progress but still signficantly decrease the effectivness of the underlying algoithms. The existing work also presumes the fingerprint data has been aligned, e.g., using core-delta alignment, which is problematic. Even given aligned data, the best reported results only increased the Equal Error Rate of the underlying algorithm by factor of 2 while they embedded a 40bit key.

Non-invertible forms were first suggested in [Ratha-et-al-01]. Non-invertibility alone does not provide significant protection. Since fingerprint systems are tolerant of moderate error levels, even if the ambiguities can never be resolved, the protection may still not be sufficient. In their more recent work [Ratha-et-al-07], define sophisticated transforms. They present performance data on their matcher, using an IBM internal DB of 181 pairs. The revocable transforms reduced the system accuracy, it is hard to interpret the performance numbers of from an internal DB. Furthermore, these transforms, which are formally non-invertible, have very limited ambiguity. In their best performing "surface folding" transform, only about 8% of the data changes its local topology, hence we can conclude only a small fraction of the data is logically non-invertible. Given the transform, one could invert the non-folded regions, and then take each point in the folded region and send it to the few (2 or 3) potential locations. Since a fingerprint matcher would likely match a print with 8% extra data, we would consider that to effectively compromised and hence not cryptographically secure.

A good review of biometric key generation is given in [Uludag et. al. 04]. The idea is a mixture of quantization and encryption of the biometric data to produce a unique key. However, the process of encryption will transform the input with only one bit difference, i.e., nearly identical biometrics, to very different numbers. Given that any biometric has a range of expected variations for the same individual, either the biometric key will often not match the individual or the data must be degraded so that all variations for an individual map to the same data. However, this would significantly degrade the miss detection rate. The results in (Uludag et. al. 04) show a loss of two orders of magnitude in accuracy.

In [Boult 06] an approach somewhere between the non-invertible and key-generation approach was proposed for face-based biometrics. That approach combines the ideas of transformation of data, robust distance measures and encryption of biometric data. After some scaling, it separates data into two parts, the fractional part, retained for local distance computation, and the integer part which is then encrypted. When applied to faces, the approach significantly improved the performance of the PCA, LDA and EMGB algorithms. It is the only algorithms of which we are aware that actually improved performance while providing privacy protection and security enhancements.

2. Cryptographically Secure Biotoken Overview

The computation of our Cryptographically Secure Biotoken, which we call a Biotope™, uses a feature space transform applied to an existing minutiae-based fingerprint system. The approach supports both transforms that are public-key cryptographically invertible and/or using cryptographic one-way functions such as MD5. In either case, even if both the parameters and the transformed data are compromised, there is no practical way to recover the original data, thus, removing the risks if centralized databases are compromised. (Obviously, given access to the private key plus the transform parameters and data would allow inversion, but that key is not used in the verification process and need not be online at all.)

In short, the fundamental advances of the approach are provided by a transformation that provides a robust, distance-based computation for supporting confidence in verification while supporting revocability and verification without identification, while at the same time, permit have thousands of simultaneous instances in use without the ability for anyone to combine the stored data to reconstruct

the original biometric.

Before we introduce the transform, we discuss a helper function. As we split the encoded data, we will be using a modulus-like operation. Such an operation can take nearby elements and separate them by significant distances. Thus, we developed what we call a reflected modulus, or cmod, such that nearby elements are mapped to nearby elements after applying cmod. We implement this with a folding technique to map items near each other after mapping, e.g, if we want a window of size E, we let $x = d \% (E*2)$,

$rmod(d,E) = x$ if x< E, and
$rmod(d,E) = (E*2)-x$ otherwise.

It is easy to show that if x and y are such that $|x-y|<t$ then $|rmod(x,z)-rmod(y,z)| < t$. Since cmod does not increase distances between points, as a traditional modulus can do, it is better suited to many of the transforms needed in public key biotokens.

We first review the core ideas in [Boult-06], then discuss its adaption to fingerprints. A key insight into the approach is that a robust distance measure is, by definition, not strongly impacted by outliers. Logically, outside a window, non-matching data has constant, or zero, impact. Many fingerprint systems use a robust distance in matching minutiae, e.g., in [Ratha-et-al-07] they use a matcher which ignores any match outside a fixed size box.

In [Boult-06] we used this observation to define a transform that scales the data then separated them into the fraction (r) and integer (g). The integer, g, is considered stable and must match exactly so then when these fields are encrypted they will still match. That paper presents a theorem that if the scaling is correct, then the robust distance measure on the raw data and the induced distance measure after encoding can only improve the matching performance.

While floating point and fraction/integers may be appropriate for face-based representations, for fingerprints the data is inherently small integers. We still transform each datum via $v'=(v-t)*s$, with scale (s) and translation (t). However, we then separate each datum into 2 parts, one, q (quotient), that must match exactly, basically defining the "window" for the robust computation, and the second, r (reflected modulus), which is not encoded and which supports the local distance computation. Given a parameter E, which depends on the expected range of variations in v, we define residual $r = rmod(v',E)$, and quotient $q = int(v'/E)$. Then we can apply one-way or cryptographic transform of q to produce w, which must match exactly. As the data is separated in to r and q, the result leaves an unencrypted value, r, within the "window" in which local distance can be computed, and then encrypts the larger (and hence very stable) part of the position information, thus effectively hiding the original positional data.

To ensure that the biometric data is protected even if the "transformation" parameters are compromised, we need to ensure that the q values are cryptographically secured. For large data items, e.g., doubles, encryption of q may be effective. For small data items, as we have in fingerprints, additional work must be done to protect the data. For a single small field there is little that can be done. As we will see later, for a collection of fields, there is a mix of both public-key and hashing that can protect many small fields and improve the overall performace while making reissue straightforward.

We can also add a user passcode. This passcoded biotoken inherently provides two-factor security mixed such that only a single biotoken is stored. The inclusion of the passcode provides strong revocation, and makes the resulting biotoken "verificaition only", providing increase privacy protection and the best protection from a doppleganger threat. Applications that require "duplication detection" during enrollment, e.g., passports, can use regular biotokens during enrollment and verification-only biotokens during operation. The enrollment testing DB, which is going to be infrequently used, can be more tightly controlled and keep the keys needed for generation of the revocable tokens in a different server. Then the verification only biotokens can then be used for the day-to-day operation. We call this approach a *operationally-verification* system, which is still considerable better for both privacy and security than a traditional verification system or even a revocable biotoken-based verification. We consider this "verifivation only" biotokens the only real protection against a doppleganger threat since the use of a revocable technology does not stop someone from searching a DB to find a victim that is a natural match. If two people's biometrics approximately match after a revocable transform they almost assuredly likely match in raw form. So a *operationally-verification*, technology is the only real defense against a doppleganger threat, and tbe best defence against the biometric dilemma.

Before we detail how we have implemented our initial finger-based biotokens, we discuss the Bozorth matcher on which it is based.

3. Background on the Bozorth Matcher

The description of the original matcher is based on the source code and on [Watson-et-al-04]. The natural form of the Bozorth matcher takes as input a minutiae file with x,y,θ,q, where x,y is the location, θ the angle and q the quality, with such files produced by mindct from the NIST toolset. The matching algorithm is comprised of three major steps:

1. Construct intra-fingerprint minutia pair comparison tables for the probe fingerprint and one table for each gallery fingerprint to be matched.

2. Construct an inter-fingerprint pair-pair compatibility table, wherein the system compares a probe print's minutia pair comparison table to a gallery print's minutia pair comparison table and constructs a new pair-pair compatibility table.

3. Scan the inter-fingerprint pair-pair table
   a. Traverse and link table entries into a web/forest of clusters that have consistent orientation and consistent endpoints when linked per cluster.
   b. Combine compatible clusters and accumulate a match score.

To construct an intra-fingerprint minutia pair table, the system takes each pair of minutiae that are sufficiently close and generates a pair table entry for them. Each pair table entry stores seven pieces of information $\{\boldsymbol{d_{kj}}, \beta_1, \beta_2, \boldsymbol{k}, \boldsymbol{j}, \theta_{kj}\}$. These are, respectively, the distances between the pair, the angles of each minutia with respect to the line connecting them, the overall orientation of the line connecting them and the indexes of the point in the pair.

To construct an inter-fingerprint pair-pair capability table, we must determine which pair-lines match between the probe and gallery. The distance between minutiae pairs is independent of rotation and translation of the original print and thus can be matched directly between a probe pair and a gallery pair. The differences in pair distances are within a relative threshold. In particular the Bozorth algorithm considers a pair to match if $(d_p-d_g)^2 < (.1*(d_p+d_g))^2$, where d_p (d_g) is the distance between the probe (gallery) pair being considered. (Do you see the robust distance measure there?) Logically, rotation is more complex, since the rotation angle needed to bring one pair of minutiae into alignment will not be the same as the angle needed for another pair of pairs. Note that each pair table entry stores the angles between the end point minutia's orientation and the intervening line between both minutiae. Therefore, the endpoint angles remain relatively constant with respect to the intervening line regardless of fingerprint rotation. Thus, the angles are considered matching if the probe and gallery angle fields are within 11 degrees. This stage can match the distance and the two angle fields, to determine compatibility, and if they are compatible, enters the relative rotation (computed from the difference in θkj between the probe and gallery) and the indexes of the minutiae in each pair as the next entry in the inter-fingerprint pair-pair compatibility table.

The final stage is to traverse the inter-fingerprint pair-pair compatibility table forming clusters and computing the score. The algorithm sorts the pair list on rotation angle and then does a complex traversal building forest clumps that have approximately consistent rotations and such that each element of the cluster has a common minutiae vertex, in some compatible pair, with another element of the cluster. The score is the sum of the number of minutiae in the best matching web.

4. Bozorth-based Biotoken

This section briefly describes the implementation of the Bozorth-based Cryptographically Secure Biotoken and its performance. To convert the Bozorth representation to a public key biotoken, we transform the pair-table, requiring only minor changes to steps 1 and 2 and 3b. The overall steps are show in Figure 1.

An important aspect of the transform is being able to apply the transform consistently. We could just choose a single transform for all data for an individual, but as discussed in the next section this decreases the effort needed for a brute force attack. In the tests presented herein, each person has 64 separate transforms T_i, with the choice of which transform to apply being determined by the initial pairwise distance $\boldsymbol{d_{jk}}$. Each transform T_i determines its translation by generating a random number. The scale is deterministic such that each "bin" is then mapped into an interval that is at least as large as the full range of input data, to ensure overlap and aliasing, but also different for each transform. The scale depends on the expected range of minutiae values and sensor resolution. For each field, we apply the translation first, then the scaling, then separate it into r_{jk} and $q_{j.}$.

The index for the transform T_n is computed as a function of both the input distance and the angles. Because a small perturbation in the input could result in a different index, we test if the index is near a boundary, and if so, expand the input pair into two encoded pairs, one for each of indexes to which it is close. Similarly, if r_{jk} is close to zero or to E, then q may be off by 1 error and a secondary row can be generated with the next index to improve matching

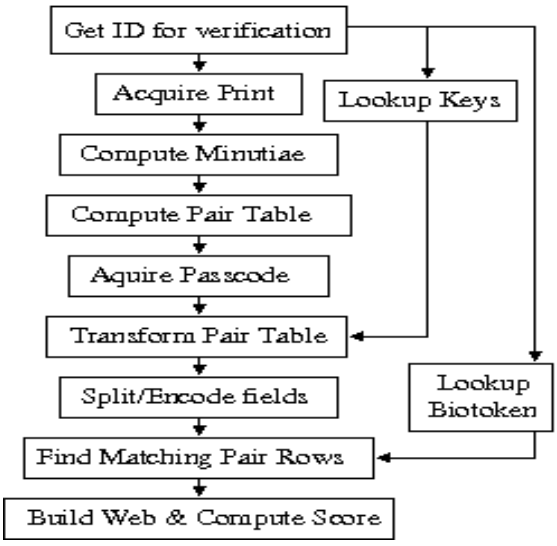

Figure 1: Steps in biotoken creation and matching

ability. When a secondary row is used, it is possible for both rows to match.

When matching rows, we first check if the associated *q* fields match exactly, and only then do we check the distances for each field. Once the "row matching" is done, the rest of the Bozorth algorithm continues with no change. However, since we introduced duplication of rows, we also define a normalization that adjusts for the fraction of rows actually used in row matching.

5. Finite Field Effects and Security

The entire process of converting back from the Bozorth pair-table to the initial minutia is not obvious, but security requires stronger design and analysis, which we briefly review. While the general concept of biotoken generation is straightforward, there are two important issues when working with small fields. One is a question of "bloat". The second is a security concern.

While [Boult-06] described the process by discussing "encryption" of an encoded field, using Public Key Infrastructure (PKI), the size of the encrypted data must be at least as long as the key. Thus, a 64bit "length" double field, when encrypted with RSA256, would need to be padded and end up being 256bits long. The added bloat can be eliminated if, say 4 of these fields are combined before encryption, but then the fields are coupled and a failure to match in one impacts the ability to match in others. For face, the feature "ordering" allows a consistent "grouping" of fields. For fingerprints, this cannot be done since consistent grouping without alignment is impractical and some level of bloating seems inherent for fingerprints.

The second, and more important, issue is the security against a brute force attack. While PKI encryption may be computational intractable to invert, if the data encoded is a small finite field, say a 10, 16 or even 32 bit number, it is quite practical to try encoding all possible inputs and see if they match. When addressing this issue with traditional encryption, the data is padded with random data, before encoding, the pad data ignored after decoding. However, in our case, we don't decode the data for matching and there is no way to separate the encoded data from the pad. The encrypted padded field cannot be matched unless the same pad is used for the probe and the gallery, which would then mean, if it was compromised, it could be used for a brute force attack.

One solution to these two issues for fingerprints is a mixed approach allowing both PKI invertability and multiple encoding. For a given "row" in the Bozorth matcher, there are three 8-bit fields we leave alone (k, j, θ_{kj}) and 3 primary fields that we need to encode: d_{kj} (a 16 bit integer distance) and two angles β_1, β_2 (formally represented as 16 bits, but practically 9 bits). To protect these fields, we transform them as described earlier. In summary, after transform, we have 3 control bytes that were not protected (or transformed), 4 bytes of residuals, i.e., *r* values, and 4 bytes of *q* values. See Figure 2 for a diagram of the fields.

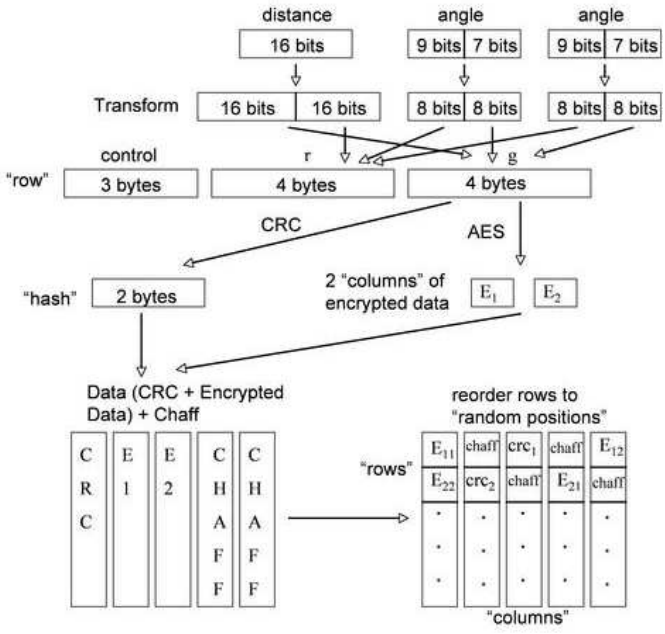

Figure 2: Data mapping to provide for protection of the small bit fields in the "row pair" table representation.

Since all three "q" fields must match, we could covert the four protected q bytes into a single number to protect without changing the matching. The data to be protected would only be 32 bits, too small to properly protect on its own. The process by which a "row" is transformed uses 64 different potential transforms, with the transform index chosen based on the original data, hence unknown to brute force attackers. Even with that 64 fold ambiguity, we must do something more.

To support full PK inversion, we use PK to encrypt an AES key, a random index, plus padding. Then we use AES to encrypt the concatenated 4N bytes data, 4 bytes from each row. This produces two "columns" of data. However, because of mixing, this data cannot be used to test the window location for any row. To address this, we take the raw data to be protected for a row, and compute a CRC of a company specific key followed by any user passphrase, if any, followed by the 4-byte data to be protected. (We use the term CRC, or cyclic redundancy check as a general concept. Any "checksum", including cryptographic checksums such as MD5 or SHA1, could be used.) The CRC folds the data in such a way that in a brute force attack there will be many inputs, say *p*, which all produce the same encoded result as the correct data. It also ensures that with any change in key or passphrase the results are unlikely to be the same. In our case, we take the 32 bits of

real data, plus the keys and passphrase, down to an 8 or 16 bit hash. Because the keys are used in the process and may be known by the company, we cannot count them in the projection gain (and the passphrase is optional), so going from 32 unknown bits down to 8 or 16 yields $p=2^{24}$ or $p=2^{16}$. Thus, to recover the original high-order bit data for the row, a brute-force attack will need to resolve the *p*-fold ambiguity. We can further increase the ambiguity by having multiple columns in the encoded data, one for the CRC-result of protected data, 2 for the AES encrypted data (or 4 if we use 8 bit CRCs) and, if desired, additional columns of chaff (random data). In the "enrollment template", we randomize the columns (separately for each row) so there is no apparent ordering, but using the key that was in the encrypted block, we can define the "random positions" for the AES encrypted-invertability data within each row. (Obviously, the PKI encrypted AES key and index must be determinable location).

With a total of *c* possible match positions for the data in the columns of data+chaff, this produces a (64**pc*)-fold ambiguity a would-be attacker must resolve to recover the data on that row. (The factor of 64 is from the number of transforms that might have been applied to a row). Importantly, we don't have to resolve this ambiguity when matching, because we consider a match of any field against any field (without replacement), plus require the three residuals fields to match. For a true positive, the process will match the encoded result. The chance of an imposter matching the CRC, even given the potential ordering and chaff is less than 1 in 20,000 and when one adds the requirement of simultaneously matching the 3 residuals, it is small enough to not significantly impact the overall matching performance. It is important to note that accidentally matching a few rows (even a few dozen) does not have a significant impact in a Bozorth-like matcher, because a spurious row can only impact the final answer if it can form a part of a larger "web" of results with a consistent overall rotation/translation. The formation of a web makes it very unlikely random matches could produce a significant false match score.

In terms of the security analysis, however, it is critical to note that to recover the original data requires much more than just resolving the pc-fold ambiguity per row. There is no test, per row, that can help decide which is correct. While we don't know if it can be done, it seems plausible that by combining different rows simultaneously one might construct a consistent "web" of underlying minutiae that may provide a test for constancy. It is unknown if this would be unique, and hence identify the true data, or if it would still result in a many-fold or infinite ambiguity. If it is possible to develop such a test it would require simultaneously resolving the $\boldsymbol{pc}$-fold ambiguity for $\boldsymbol{n}$ rows, each of which is independent. Thus, a brute force search would require $\boldsymbol{n}^{64pc}$ attempts. In the worst case, one might generate $\boldsymbol{m}^2$ rows from $\boldsymbol{m}$ minutiae, though in practice we limit to 512 rows independent of the number of original minutiae, deleting less significant rows. It is clear that to recover something that might be an acceptable subset of a print would require at least 1 row per desired minutiae and more likely 2 rows per minutiae. To get a relatively minimal "16 point" print would require recovering at least 16 rows, and a more realistic estimate is that one would have to recover at least 32 rows to even have a small chance of forming a web that properly interrelates 16-minutiae points, and n=64 to have a decent chance to recover a larger point match. (Again we don't know how to go from rows back to minutiae so the numbers needed are somewhat uncertain.)

In our current implementation $64\boldsymbol{pc} = 2^6*2^{16}*2^7=2^{29}$, so for a brute force attack to recover 16 minutiae would requirea minimum $n=2^4$, $\boldsymbol{n}^{pc} = 2^{(4*25)} = 2^{100}$ and more realistically it would be $\boldsymbol{n}=2^{7,}$ $\boldsymbol{n}^{pc} = 2^{(7*25)} = 2^{175}$ brute force attempts to recover 16 original minutiae. Again, all of this analysis is presuming that after generating hypotheses for each of the unknown items in a row there is a testable hypothesis to confirm the collection of rows as correct. No such algorithm is known, but since we have no analysis to suggest it is computationally infeasible, we do not include its difficulty in our security analysis.

The important thing is that the approach has provided sufficient ambiguity that, unlike the encoding of individual fields or the folding approach of (Ratha et. Al. 07), it provides a reasonable protection against a brute force attack. Given that there are other ways to "acquire" fingerprint biometrics, such as following a person around and picking up a left object, the 2^{100}-2^{175} seems sufficient protection. If it is not, adding more chaff columns (i.e. increasing $\boldsymbol{c}$) and or having additional data (e.g. minutiae type) that is part of the "stable data" (i.e., increasing $\boldsymbol{p}$), exponentially increases the protection.

Of course, multiple variations on this idea allow tradeoffs between storage size, computational cost and security. Originally, we implemented this as described, but because the inversion of the pair-table is not obvious, and because it is actually larger than the original minutiae data, we moved to a more efficient form wherein we encrypt a compressed form of the raw minutiae data rather than the 32 bit fields to be protected. (I.e. E1/E2 in Figure 1 are replaced with AES encrypted raw minutiae). We still generate the CRC-version of the q fields and use it in matching. Since the raw minutiae data is block encrypted, it is properly protected. Of course, none of that data can be used for approximate matching, but it does make good chaff. Since both the encrypted minutiae and the AES encrypted "q" data appears as random chaff to the matching algorithm,

the difference is immaterial to the performance but it makes the PK- inversion much simpler and requires less space.

A secondary advantage of our biotoken approach is that it supports a simple "company" level re-issuance. When the data is encoded, the "index" that allows one to identify the CRC-data among the chaff can be encoded with the company's master public key. If the company stores that as an enrollment master-public key biotoken, they can then use their private key to recover the order and issue an operational biotoken. The operational biotoken is generated by using an additional key as a post-pad to the CRC-computation of the data fields, e.g., take the encoded 16 bit CRC from the user, append the new keys and compute a new 16 bit CRC. Since this is non-invertible, they can then PK/AES encrypt the original CRC-values, replacing chaff columns with the results. If an operational biotoken is compromised, or if the companies' biotoken policy usage limit is reached, the company can use their key to recover the original CRC values (i.e., go back to their original master public key biotoken) and then reissue a new operational biotoken from their master. Still the users data is protected, though knowledge of the order removes $c(2^3)$ from the *64pc* factor and reduces the brute force effort needed by the company, but it still requires very significant effort (of 2^{88}) for even an insider to try to a brute force attack. However, it provides an improved operational security model with no customer inconvenience.

The post-pending of keys, or a more general multi-stage process, can be used to support per-transaction unique public key biotokens. For example, with the CRC model we can take the operational public key biotoken, appended a transaction-specific key, and produce a new encoded field. For the transaction-level, the system does not need to understand the order or re-encode the original CRC data because no additional transforms will be applied after the transaction, so there is no need for inversion and it can just apply the final CRC computation to all the columns, and does not reduce the security at this level at all. For matching, the user's biometric is then subjected to a similar process and the results can be matched. While the true traditional CRC-based approach may be sufficient for basic transactions, higher security applications could use more advanced cryptographic hashes, which require larger storage and more computation. They can also use a CRC/hash such that the operational and transaction key, though applied separately, can be combined into a single key/transform to be applied so that the user's machine never receives the separate keys.

6. Performance

We implemented the fingerprint-Cryptographically Secure Biotoken by extending the NIST/FBI Bozorth matcher (also called NIST VBT). There are at least 2 major

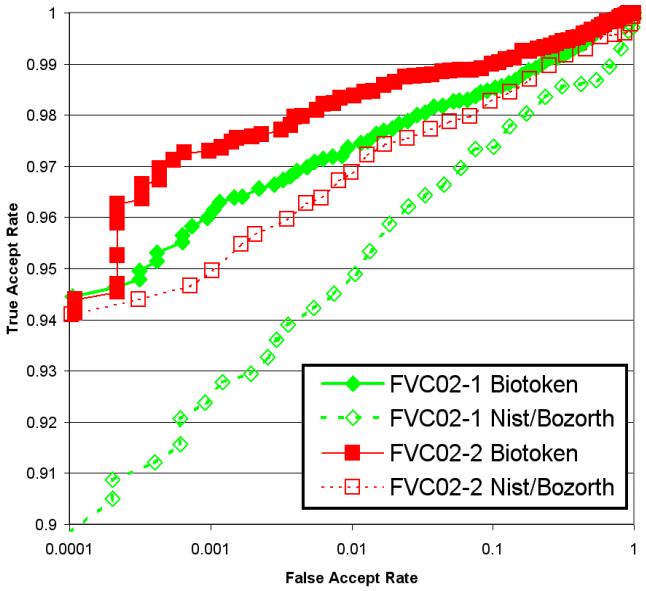

Figure 3: ROC curves comparing Biotope™ biotokens and the NIST/Bozorth matcher on FVC2002 data

aspects of performance, speed and accuracy, which are discussed separately.

During enrollment we require the generation of an RSA key and full PK encryption, which is the most expensive step. On a 380x380 image, the computational aspects of enrollment take approximately 750ms to 3000ms on a 1.6Ghz Pentium 4 processor depending on the size of the chosen RSA key (512-2048 bits). Of this 250ms to 2500ms for the key generation and encoding the AES key, 350ms is for the minutiae extraction and image processing, and other parameters and 50ms is for the AES encoding and secure biotoken generation. Matching does not require the PK encoding steps, greatly reducing the time to a total average of 423ms, of which 394ms is for the image processing and 29ms is the biotoken generation and matching. This is only 8ms more than the time for the standard NIST implementation of the Bozorth matcher on which our biotoken is based.

More important that speed, however, is how the biotoken process impacts matching accuracy. Accuracy is a strong function of the number of minutiae or table size maintained. For the Bozorth algorithm, we use the pre-supplied defaults, which allows for 150 minutiae and 10,000 pairs. For the biotoken, we limited the table size to keep the biotoken storage size below 24K, with an average size of 13K. Limiting was done first using the defaults for pruning on NIST-computed quality of the minutiae but also trying to ensure that each minutiae was included in a few pairs rather than letting the best minutiae take part in all of their pairs. This was done to ensure better spatial coverage. While a few may consider a 20K token excessively large, we believe a little storage is a small price to pay for the

Dataset	Biotoken Verification EER	Improvement Over EER of NIST VBT
FVC 2000 db1	.029	30%
FVC 2000 db2	.025	37%
FVC 2002 db1	.021	34%
FVC 2002 db2	.012	30%
FVC 2004 db1	.086	39%
FVC 2004 db2	.075	33%
Table 1: Finger Biotope™ accuracy		

security and privacy enhancements of biotokens. Then again, with 20K biotokens, 50 Million tokens fit on a Gigabyte card and tokens for all of the US fit on a laptop.

We made only minimal changes to the matcher code, extending it to handle added columns and to test the encoded fields with the "subset matching" described earlier. Because the encoding of the tables and quality pairing can change the number of entries, we added normalization to the scoring based on the number of rows used.

For analysis we applied both algorithms to the well-known Fingerprint Verification Challenge datasets, e.g., see (http://bias.csr.unibo.it/fvc2000). Each of the FVC200? verification tests has 8 prints each of 100 subject, producing 2800 true matches and 4950 false match attempts. Figure 3 shows ROC curves comparing the biotoken algorithm with the original NIST/Bozorth matcher. The new biotokens consistently outperform the original algorithm. (Note this is a semi-log ROC.) To define the improvement quantitatively we use the Equal Error Rate, shown in Table 1, and have an average of 33% reduction in the EER.

Including more features during matching (e.g., ridge counts) might improve biotoken performance but were not included because they are not used by Bozorth3 and would be an unfair comparison. Even without the added features, for FVC2000, these scores would have resulted in it being the 3rd place algorithm overall, and in the top ten in FVC2002. The FVC2004 data requested subjects to intentionally distort their prints, which may have moved minutiae outside the window used for matching and in building the feature-web in the Bozorth algorithm, which negatively impacted both algorithms' performance.

While the accuracy gains with our approach are not as significant we reported in [Boult-06] for face, they are still significant. In addition, prior fingerprint-based techniques attempting to provide privacy, e.g., [Ratha-et-al-07] or [Tuyls-et-al-05] have all had to trade accuracy for privacy.

7. Conclusions and future work

The paper introduced the use of a robust revocable fingerprint-based biotoken. We analyzed previous work and showed it lacks in bother accuracy and/or security. The paper introduces a "reflective modulus" operator with an important local neighborhood "nearness" preservation property, which is important to the effectiveness of the biotoken algorithm. The transforms combined with encryption maintain the privacy while the unencrypted part supports a robust distance measure, something that is critical to make biometrics effective. While the paper presents only fingerprints, the approach applies to almost all biometrics.

As Admiral James Loy, Head of Transportation Security Agency, stated at the 9th Annual Privacy & American Business Conference, 2003 "Don't be too quick to strike a balance between privacy and security. As Americans, we are entitled to a full measure of both". Secure biotokens show, that at least for biometrics, we don't have to accept the loss of privacy to gain security. Biotopes™ and other secure biotokens can solve the biometric dilemma and not only do they provide privacy, they actually improve the accuracy of the underlying biometrics!

8.0 References

[Boult-06] T. Boult, "Robust distance measures for face recognition supporting revocable biometric tokens", IEEE Conf. on Face and Gesture. 2006

[Krause-01] M. Krause, M. "The expanding surveillance state: why Colorado should scrap the plan to map every driver's face and should ban facial recognition in public places," Independence Institute, Issue Paper, Number 8-2001.

[Matsumoto-et-al-02] Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, Satoshi Hoshino "Impact of Artificial "Gummy" Fingers on Fingerprint Systems" SPIE Vol. #4677, Optical Security and Counterfeit Deterrence Techniques IV, January 2002

[Ratha-et-al-01] *N. Ratha, J. Connell, R. Bolle* "Enhancing security and privacy in biometrics-based authentication systems", *IBM Systems Journal*, vol. 40, no. 3, pp. 614-634, 2001

[Ratha-et-al-07] Ratha, N.K. , Chikkerur, S., Connell, J.H. and Bolle R.M. (2007) "Generating cancelable fingerprint templates", to appear IEEE PAMI, 2007

[Tuyls-et-al-05] Tuyls, P. Akkermans, A. H. , Kevenaar, T. A., Schrijen, G. J. , Bazen, A. M., and Veldhuis R. N. Practical biometric authentication with template protection. In AVBPA, pages 436–446. 2005

[Uludag-et-al-04] U. Uludag, S. Pankanti, S. Prabhakar and A.K. Jain, ,"Biometric Cryptosystems: Issues and Challenges", *Proceedings of the IEEE*, Vol 92, No 6, June 2004

[Watson-et-al-04]. C.I. Watson, M.D. Garris, E. Tabassi, C. L. Wilson, R. M. McCabe and S. Janet, "User's Guide to NIST Fingerprint Image Software 2"