© 2008 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 the Biometrics Symposium 2008.

The published article can be accessed from:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4655516

BIO-CRYPTOGRAPHIC PROTOCOLS WITH BIPARTITE BIOTOKENS

W. J. Scheirer and T. E. Boult

VAST Lab, University of Colorado at Colorado Springs and Securics, Inc.
Colorado Springs, CO.

ABSTRACT

Cryptographic protocols are the foundation of secure network infrastructure, facilitating authentication, transactions, and data integrity. In traditional cryptographic protocols, generated keys (and, in most cases, passwords) are used. The utility of biometrics as a convenient and reliable method for authentication has emerged in recent years, but little work has been performed on a serious integration of biometrics with cryptographic protocols. In this paper, we review the notion of revocable biotokens, explain their nesting properties, and extend them to *bipartite bitokens* and use these to develop protocols for transactions, digital signatures, and a biometric version of Kerberos. We show bipartite biotokens offer a convenient enhancement to keys and passwords, allowing for tighter auditing and non-repudiation, as well as protection from phishing and man-in-the-middle attacks.

1. INTRODUCTION

The cryptographic infrastructure that has been developed and deployed on various computer networks over the past decade is very good at solving the problems within its capabilities, but is continually thwarted by attacks involving key exchanges and user interaction. Exchanging keys between two parties that do not share a secret is difficult. Witness the man-in-the-middle attack of figure 1, whereby an attacker inserts themselves between two parties doing key-exchange and secure communication. These are not idle concerns. In 2003, Microsoft's familiar Remote Desktop Protocol was shown[1]to be vulnerable to a man-in-the-middle attack; the client provided no verification of the server's public key. In the course of "fixing" the problem, Microsoft hard-coded the private key used to sign the remote desktop server's public key into the operating system - a leak of information that continues to facilitate the attack to this day on pre-Vista machines. The quality of "secure" protocol implementation varies, especially when it comes to key management.

On the Internet, the all-too-familiar "phishing" ploy presents a related dilemma to the innocent user, whereby a masquerade-in-the-middle spoofs the legitimate site, in order to harvest sensitive information (passwords, account numbers, SSNs, etc.). With a bit of social engineering an attacker can lure a victim to a bogus site with a convincing email, instant message, forum post or adware. With DNS poisoning the user could even type the URL and be transparently misdirected. Loss from these attacks is estimated to be as high as 3.2 billion dollars [2].

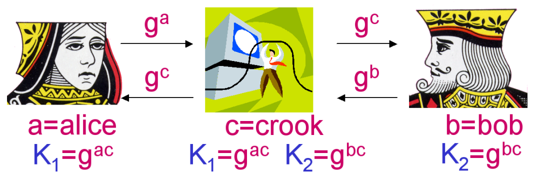


Fig. 1. The "man-in-the-middle" attack against a Diffie-Hellman-like key exchange. The attacker takes up a position between Alice and Bob. Alice believes she is exchanging information with Bob, while Bob believes he is exchanging information with Alice. The attacker receives both g^a and g^b, and returns a malicious third key, g^c to both Alice and Bob. The attacker is now free to steal any information flowing through this falsely secured channel.

If a site presents itself as authentic, how does one verify the authenticity? The solution to this problem has been Public Key Infrastructure (PKI). PKI makes use of a certificate authority to manage public keys and identities of known entities. When a user wishes to verify the authenticity of a site providing PKI, they check its certificate, which includes a digital signature that binds a public key with an identity. Unfortunately, digital certificates remain poorly (if it all) understood by the common user, and are rarely checked. Organizations routinly deploy self-signed or mismatching certificates and users are quite used to "accepting" them (using our campus wireless network requires accepting a certificate signed for a different domain). Users, often out of necessity, are being trained to ignore the warning sign of an attack. In an attempt to address these common security problems, we pose a question: can biometrics solve some of the problems inherent in current cryptographic protocols?

Using biometrics to solve some of the authentication aspects of security is, of course, well known. Unfortunately, traditional biometric data cannot be used for remote authentication with a potential unknown source as it would need to be encrypted to stop a man-in-the-

This work was supported in part by NSF STTR Award Number 0750485 and NSF PFI Award Number 0650251

[1]http://www.securiteam.com/windowsntfocus/5EP010KG0G.html
[2]http://www.gartner.com/it/page.jsp?id=565125

middle or phishing attack from acquiring and using the biometric data. The asymmetric and non-revocable nature of traditional biometrics makes them unsuitable for non-attended verification. The asymmetry is that one party, say Alice, maintains the store of biometric data for matching. The other, say Bob, has the live sample for verification. In a traditional biometric, Alice has to send the matching data to Bob and then trust the result when Bob says it matches or not. Alternatively, Bob can send his raw "live" biometric data to Alice and trust she is the proper source for matching, while also trusting her stated results. Either way, one side must place considerable trust in the other, for both matching "results" and for protection of the privacy of the data. Their exchanges require encryption to protect the data in transit, which requires a solution to the key exchange problem rather than providing one. Traditional biometrics are thus not protected from phishing or man-in-the-middle attacks, and cannot solve the key-exchange problem.

The topic of revocable (or "cancelable") biometric tokens (hereafter biotokens) has been gaining attention in the research literature. Involving a synthesis of biometrics and computer security, biotokens often present a usable solution on the pattern recognition side of things, but unfortunately, to date, most leave something to be desired in their security analysis. Some of the earliest work is found in [1], which introduces the idea of "biometric encryption". Attacks against "biometric encryption" include hill-climbing [2], and the observation that the scheme reduces to a one-time pad where the pad is re-used [3]. [4] extends earlier work in non-invertible transforms; though formally non-invertible, this scheme is functionally invertible within matcher tolerance [5]. The ever popular "fuzzy vault" [6] relies on locking data in a collection of chaff, but in its original form, shows significant weakness to brute force attacks aided by the behavior of the error correcting codes within the scheme, [7], chaff point identification [8], known key, record multiplicity, and blended substitution attacks [3]. Fuzzy extractors [9] allow for reliable key release from biometric matching, but are largely constrained to the theoretical literature, and may suffer from practical constraints during error-prone data collection [10]. Robust distance measures supporting revocable biotokens are introduced in [11] and [5], along with a thorough security analysis in each presentation. These two works provide the biotoken basis for the new concepts in this paper.

While prior works on security and biometrics address the privacy issues and improve security in many ways, most do not address the inherent asymmetric nature of matching, where one side must still trust the other, which limits remote authentication (web-based biometric authentication, for example). Fuzzy extractors have been proposed [12] for secure remote authentication and secure key exchange, but they do not support one-time transactions or hierarchies of trust. The impact of these two issues is described in detail in this paper. Similar in spirit to digital signatures and certificates, this paper introduces the bipartite biotoken approach, which allows both parties to mutually validate the transaction. The approach addresses privacy, secure matching and non-repudiation. The result of matching releases a token, which is signed and returned to the non-matching party. The approach completely prevents replay, phishing and man-in-the middle attacks; no transmitted data is ever reused except the public-key. With bipartite biotokens, we can develop protocols incorporating both cryptography and biometrics.

The rest of this paper is as follows. In section 2, we revisit the work of [11] and [5], in order to understand the properties of revocable biotokens, and how they may be used in cryptographic protocols. In section 3 we explain the important nesting properties of revocable biotkens needed for our transactional constructs. In section 4, we introduce bipartite biotokens and develop a protocol for bipartite biotokens for mutual validation. In section 5, we review the concept of digital signatures, and then extend the idea of bipartite biotokens for enhanced biometric digital signatures. Finally, in section 6, we introduce a version of the Kerberos trusted third-party authentication protocol that incorporating bipartite biotokens.

2. REVOCABLE BIOTOKENS

At the heart of any biometric matching is the computation of distance between noisy samples. In the matching process, a distance is computed between the probe (the submitted sample) to the gallery (stored data). For verification, the resulting distance is thresholded. For recognition, minimum distance over the gallery can be used. The work of [11] and [5] introduces a new form of revocable biotokens for face and fingerprint respectively. The core concept behind the process is the same for both: a robust distance computation that preserves distance when matching in encoded space.

Since biometric data is inherently noisy, variation must be accounted for in any data protection scheme. Intuitively, if the data can be split into stable and unstable components, the stable portion can be encrypted in a reliable fashion, while the unstable portion is left in the clear. This observation allows for the definition of a biotoken transform that scales/translates the data, and then separates it into a quotient q and modulus or remainder, r. Since q is stable, we can encrypt or hash it for both probe and gallery data, and require exact matching. This transform induces a distance measure in encoded space: first test if the encoded q values match; if they do, the residuals r are then used to compute distance. If the q values don't match distance for this field is a constant. The security and privacy is provided through the use of strong encryption/hashing, protecting q.

The residual region r facilitates the aliasing of all data into itself. The q component, transformed via scaling and translation to a general wrapping number, is encrypted via public key cryptography, leaving the encoded value g, and mapped back into r. The number of times the data is wrapped is the value of g. The mapping hides the actual value, but leaves an unencrypted value within a window with which we can compute local distance. The robust distance measure supporting the match in encoded space gives a constant penalty c to outliers (when the encrypted portions of each biometric signature field do not match, or when the residuals fall outside of the window b), and thus, prevents them from strongly impacting the overall distance metric. Assuming fields p, q from two different biometric signatures, encoded using scaling s and translation t yielding $r(p), r(q), w(p), w(q)$, the robust dissimilarity measure metric $d(p, q)$ is defined as:

$$d(p,q) = c \textit{ if } (w(p)! = w(q)||abs(r(p) - r(q)) >= b)$$

$$d(p,q) = (r(p)/s(p) - r(q)/s(q))^2 \textit{ otherwise}$$

A proof that these privacy enhancements do not decrease, but may increase, the accuracy of the recognition system is given in [11]. A treatment of the issues impacting scaling and translation is also given in [11].

Because the "protected" data and the residuals are clearly separated, the "protected" data can be encoded multiple times. In particular, it can be transformed using the end-user's public key, and, if a verification-only token is desired, mix in a pass-phrase (or a second factor) that is never stored. Then, using a company key, another transform is applied, with the result stored. The company can also transform again to an operational token so that it can be routinely modified without bothering the end customer to reissue. Simple reissue is critical; a company would never commence large-scale re-issues if every user had to submit another biometric sample.

3. NESTED BIOTOKENS

The re-encoding of a cryptographic structure is a desirable property, as it allows for hierarchies of trust, and the secure release of data without necessarily revealing other underlying secrets. Traditional Public Key Infrastructure (PKI) [13] is built upon the notion of a web of trust, whereby any number of entities, both related and unrelated, can share and sign keys, in order to validate the authenticity of a key, or series of keys. Certificates are used to manage public key / private key pairs, associated digital signatures, and revocation, should the need arise. A root certificate is issued by a top-level certificate authority; it represents the base of all derived certificates in the PKI tree structure (depicted in figure 2). Other entities lower in the certificate hierarchy can re-sign the certificate they receive from the next highest level. The utility of this is in revocation / key control process - if a certificate becomes compromised, the level above it can revoke and re-issue. Should the root certificate become compromised, all certificates in the tree must be revoked.

Fuzzy extractors [9] theoretically represent a strong alternative to revocable biotokens, but do not possess the same properties. The largest difference is that fuzzy extractors cannot be used on a transactional basis (we discuss biometrics for transactions in section 4), and they

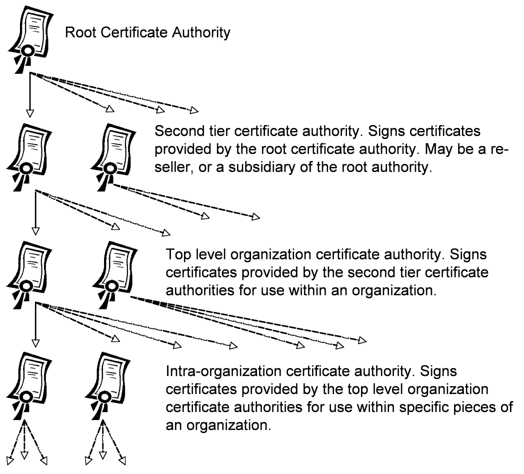

Root Certificate Authority
Second tier certificate authority. Signs certificates provided by the root certificate authority. May be a re-seller, or a subsidiary of the root authority.
Top level organization certificate authority. Signs certificates provided by the second tier certificate authorities for use within an organization.
Intra-organization certificate authority. Signs certificates provided by the top level organization certificate authorities for use within specific pieces of an organization.

Fig. 2. PKI issue/re-issue tree.

cannot be re-encoded for nested security and hierarchical management, in a manner similar to PKI. A fuzzy extractor incorporates a *secure sketch* construct to allow the precise reconstruction of a noisy input w given an instance of the sketch s and a sample w'. A secure sketch SS bound with a random number i forms the basis of the fuzzy extractor, which returns a key R, when approximate input matching is successful. From [9], we can gain an understanding of the limitations of fuzzy extractors with the following Lemma in that work:

Lemma 3.1. *Suppose we compose an* (m, $\tilde{m}$, t)*-secure sketch,* (SS, REC) *for a space M and a universal hash function* EXT : $M \rightarrow \{0,1\}^l$ *as follows: In* Gen, *choose a random i and let P* = (SS(w), i) *and R* = Ext(w; i)*; let* Rep(w', (s, i)) = Ext(Rec(w', s), i)*. The result is an* (m, l, t, ε)*-fuzzy extractor with* $l = \tilde{m} + 2 - 2log(1/\varepsilon)$.

It is the need for both the original biometric data w and a unique random number i that prevents fuzzy extractors as defined from being constructed on a transactional basis. To release a new key for each transaction, a full enrollment P = (SS(w), i) must take place, incorporating a unique i and requiring the original biometric features be stored for automatic use - the very problem fuzzy extractors are trying to address. For the same reason, there is no provision in Lemma 3.1 allowing for the ability to build hierarchies of fuzzy extractors. Any output from one fuzzy extractor used as input to another would be subject to error correction, allowing a set of incorrect released keys to approximately match, preventing the use of R as w for input into a fuzzy extractor. We note that the variables $l, m, \tilde{m}, t, \varepsilon$ related to theoretical bit security strength are not relevant to this discussion; the reader may refer back to [9] for their specific meanings.

Revocable biotokens can be re-encoded, much like the re-signing of digital certificates in figure 2. In section 2 we reviewed the notion of data splitting to support revocable biotokens. Using this knowledge, and the concept of public key cryptography, we can develop the re-encoding methodology for revocable biotokens. Assum-

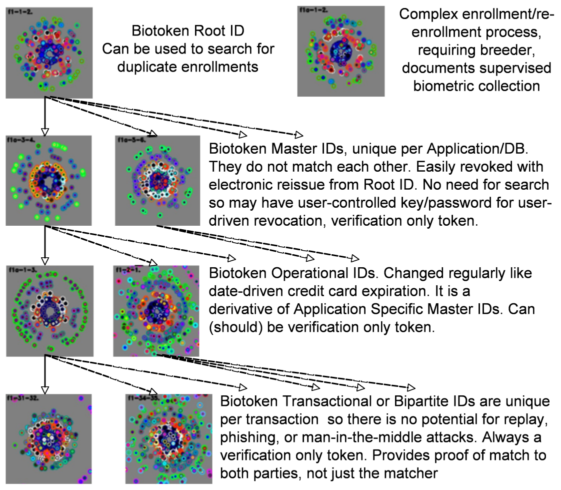

Biotoken Root ID
Can be used to search for duplicate enrollments
Complex enrollment/re-enrollment process, requiring breeder, documents supervised biometric collection
Biotoken Master IDs, unique per Application/DB. They do not match each other. Easily revoked with electronic reissue from Root ID. No need for search so may have user-controlled key/password for user-driven revocation, verification only token.
Biotoken Operational IDs. Changed regularly like date-driven credit card expiration. It is a derivative of Application Specific Master IDs. Can (should) be verification only token.
Biotoken Transactional or Bipartite IDs are unique per transaction so there is no potential for replay, phishing, or man-in-the-middle attacks. Always a verification only token. Provides proof of match to both parties, not just the matcher

Fig. 3. Biotoken issue/re-issue tree. The multiple dimensions of the visualized biotoken constructs are represented by different colors. Note the variation in the biotokens as they are transformed from the Root ID.

ing the biometric produces a value v that is transformed via scaling and translation to $v' = (v - t) * s$, the resulting v' is split into the fractional component r and the integer component g. In the base scheme, for a user j, their residual $r_j(v')$ is always left in the clear. For the initial transformation $w_{j,1}(v', P)$ of g, a public key P is required. For nested re-encodings, w_j is re-encoded using some transformation function T (which may be a hash function, or another application of public key cryptography) creating a unique new transformation for each key that is applied:

$$\textit{1st encoding: } w_{j,1}(v', P)$$

$$\textit{2nd encoding: } w_{j,2}(w_{j,1}, T_2)$$

$$\textit{nth encoding: } w_{j,n}(w_{j,n-1}, T_n)$$

Since we are using public key cryptography, the nesting process can be formally invertible as long as the private key associated with the first stage of encoding is available.

Figure 3 shows a biotoken issue/reissue tree similar to the PKI tree in figure 2. Like the root certificate for PKI, a root biotoken Root ID generated at enrollment time forms the top of the tree, and the basis for all subsequent biotokens encoded from it. From the biotoken Root ID, it is possible to search for duplicate enrollments at the base level. The second level represents the biotoken Master IDs, which are unique per application or database, and may incorporate user generated information for user controlled revocation. The third level represents the biotoken Operational IDs, which are verification only tokens that are changed regularly (akin to a date driven credit card number expiration). The re-encoding process also allows revocable biotokens to be used for transactions, described in section 4. Revocation follows up the tree in the same manner as PKI.

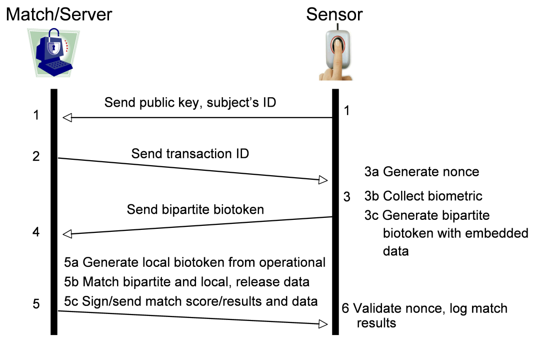

Match/Server
Sensor
Send public key, subject's ID
1
1
2
Send transaction ID
3a Generate nonce
3
3b Collect biometric
3c Generate bipartite biotoken with embedded data
Send bipartite biotoken
4
5a Generate local biotoken from operational
5b Match bipartite and local, release data
5c Sign/send match score/results and data
5
6 Validate nonce, log match results

Fig. 4. Sequence diagram for the bipartite biotokens.

4. BIPARTITE BIOTOKENS

The core of our bio-cryptographic protocols is an extension to the revocable biotoken to provide bipartite match confirmation with data embedding. The underlying mechanism is a mixture of the biotoken matching and securing process with ideas from the *fuzzy vault* area [6] using polynomial-based shared secrets and hashes for validation. This approach addresses more significant privacy and security issues than biotokens alone and completely prevents replay, phishing and man-in-the middle attacks; no non-public transmitted data is ever reused.

We define three properties for the bipartite biotoken:

1. Let B be a secure biotoken, as described in section 2. A bipartite biotoken B_p is a transformation $bb_{j,k}$ of user j's kth instance of B. This transformation supports matching in encoded space of any bipartite biotoken instance $B_{p,k}$ with any secure biotoken instance B_k for the biometric features of a user j and a common series of transforms $P, T_1, \ldots, T_k$.
2. The transformation $bb_{j,k}$ must allow the embedding of some data D into B_p, represented as: $bb_{j,k}(w_{j,k}, T_k, D)$.
3. The matching of B_k and Bp_k must release D if successful, or a random string r if not successful.

A sequence diagram of one form of the bipartite biotoken protocol is shown in figure 4. A transaction starts with transmission of the user's identifier and public key. Based on these credentials, the server responds with a transaction specific ID and other key. The sensor generates a bipartite biotoken, including some embedded data (such as a nonce). The server matches the received bipartite biotoken to a stored biotoken, releasing the embedded data if matching is successful. The sensor side ensures the released data is correct, validating the remote match, before the transaction proceeds. An alternative form has the server sending the bipartate biotoken to the sensor which matches locally and sends back the embedded as proof. Either way, both sides have confirmation of the match and neither shared any data that will be used after the transaction.

To see how we can implement this consider fingerprints. As discussed in [5], the small fields inherent in

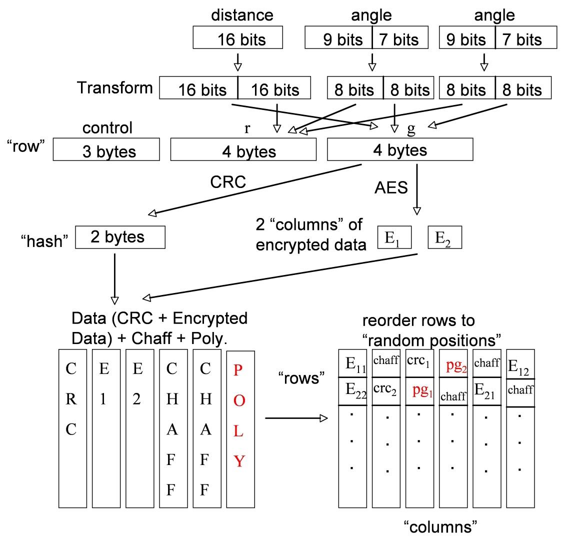


Fig. 5. Data mapping to provide for protection of the polynomial and also of the small bit fields in the "row pair" table representation.

biometrics, especially fingerprint data, present an added challenge for protection. Fingerprints also require alignment and only match a subset of the data, which must also be addressed. In [5], the "Bozorth" matcher [14] is extended using biotoken concepts and the existing "pair table" representation, allowing matching without prior alignment or special features. The base algorithm matches 3 "fields" (columns) per row and collects many matching rows to determine if a print matches. Just hashing the fields per row is insufficient to protect such small fields. For the real data, after a transform, we have 3 control bytes that are not protected (or transformed), 4 bytes of residuals (r values), and 4 bytes of q values which are hashed as a 32-bit block (see figure 5).

The post-pending of keys, or a more general multi-stage process, can support per-transaction unique public key biotokens. For example, with a CRC as the hashing, one can take the operational biotoken, appended a transaction-specific key, and produce a new encoded field by projection. 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. Rather it can just apply the final CRC computation to all the columns, so it 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 lower security transactions, higher security applications could use more advanced cryptographic hashes, recognizing that they will require larger storage. 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.

Each "row" is transformed using one of 64 different potential sets of random transforms. A hash then folds the q fields, producing a p-fold ambiguity per field, with $p = 2^{24}$ or $p = 2^{16}$. The ambiguity can be increased further by having multiple columns in the encoded data. In [5], a security analysis was presented showing that, depending on p and the attacker's knowledge, a brute force search would require at least 2^{100} to 2^{175} attempts to recover 8 or more minutiae from the biotoken. This assumes only a single encoding - multiple-encoding passes would increase the effort needed. Further, to recover minutia from the rows presumes that after generating hypotheses for the roles of each pair-table row of each field, there is a process to recover minutiae with a testable hypothesis to confirm the minutiae are correct. No algorithm for either pair-row inversion or testing is known.

For the bipartite biotoken, we modify the process to incorporate an embedded polynomial that will encode our data. Assume c columns including data, chaff, and polynomial (figure 5), and that r rows (coefficients) are needed to recover the polynomial; then the brute force attack is c^r. For example, if we have 16 columns and assume we have 10 columns in the data and 24 rows, then the brute force effort needed to recover the polynomial is 10^{24} or about 2^{80}. Increasing c increases the chance for a false match per row, which increases the demand on the error correction for the polynomial for true matches. Increasing the number of rows increases the number of biometric points that must be matched. In general, assuming a biotoken of about 10K bytes, we see a good match providing 50-250 matching rows, thus it is easier to increase the number of rows to increase security, which also increases the embedded data capacity.

By introducing a fuzzy vault-like component to revocable biotokens, we must also consider the security threats that come along with it. While a full security analysis is beyond the scope of this protocol driven paper, we do make an important security observation. In a traditional fuzzy vault scheme, we presume that some data x and subsequent encodings $p(x)$ are mixed together with a large amount of chaff data for security. The work of [3] shows that the presence of the original data x in a fuzzy vault leads to several vulnerabilities. For bipartite biotokens, x is not part of the encoded data - an attacker has access to just some encoding $p(x)$ and a hashed representation $h(x)$ of x.

5. BIOMETRIC DIGITAL SIGNATURES

By using digital signatures [15], we can provide some level of proof of authorship, as well as data integrity for a message transmitted over an unsecured channel. There are five properties for an ideal digital signature:

1. The signature should be authentic.
2. The signature should provide proof of signer action.
3. The signature should not be reusable.
4. The signature should be non-repudiable.
5. The signed data cannot be altered.

In most situations, to implement these properties, we use standard cryptographic methods. Using our familiar players, Alice and Bob, we can define the steps necessary for producing a digital signature using public key cryptography and one-way hash functions:

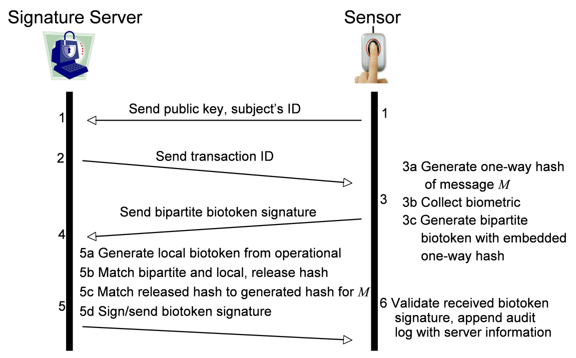
1. Alice takes a one-way hash of a message M, producing $H(M)$.
2. Alice encrypts $H(M)$ with her private key, producing $E_A(H(M))$, the digital signature.
3. Alice sends the message and signature $E_A(H(M))$ to Bob.
4. Bob produces a one-way hash, $H'(M)$, of the message he receives. He then decrypts the digital signature using Alice's public key: $D_A(E_A(H(M)))$. If $H'(M) = H(M)$, then the signature is valid.

This arrangement works well, but all of the five properties are not satisfied absolutely in the protocol listed above. Specifically, 2 & 4 do not provide the absolute expectation of integrity. In the standard protocol, we can only show that a message has been signed by Alice's private key, not necessarily Alice. Moreover, the signature $E_A(H(M))$ carries no audit information other than the action of encryption by Alice's private key. Can we use biometrics to not only implement a digital signature protocol, but to enhance the security of digital signatures as well?

Both [16] and [17] introduce a methodology for digital signature generation and usage incorporating biometrics. In [16], a method is described whereby a user's biometric and some other data are fused to generate a standard RSA digital signature in a two-factor security model. Any revocable aspect of the scheme is quite hazy, with only a suggestion of the use of fuzzy vaults as an extension to the presented work. The work of [17] is stronger in the protocol realm, introducing the idea of a Public Key Infrastructure incorporating Biometric certificates that are authenticated by comparing a user's sample with pre-stored biometric certificates of the physical characteristics of the user. Fuzzy vaults are used to protect the biometric data. As mentioned above, the security of this scheme is precarious at best since traditional fuzzy vaults are subject to multiple attacks [3]. Moreover, the model for key generation assumes that templates can be described as a sequence of n independent identically distributed random variables. This model is hardly consistent the actual distributions of physiological data.

In section 4, we developed bipartite biotokens. We can use them as the basis for a new class of bio-digital signatures. In essence, the server-side of the protocol will take the form of a *signature server*, where the signature itself will be generated from information sent by the sensor side of the transaction. The idea of a remote signature server is not new [18], but its usage here is; we solve the man-in-the-middle attacks that impacts [18].

The sequence diagram for the bipartite biotoken signature process is shown in figure 6. As in our bipartite



Signature Server
Sensor
1
Send public key, subject's ID
1
2
Send transaction ID
3a Generate one-way hash of message M
3
3b Collect biometric
3c Generate bipartite biotoken with embedded one-way hash
Send bipartite biotoken signature
4
5a Generate local biotoken from operational
5b Match bipartite and local, release hash
5c Match released hash to generated hash for M
5
5d Sign/send biotoken signature
6 Validate received biotoken signature, append audit log with server information

Fig. 6. Sequence diagram for the bipartite biotoken signature process.

biotoken protocol, the sensor side first sends a public key, and the subject's ID. The signature server responds with the transaction ID. The sensor side will generate a one-way hash H of a message M, then collect the biometric, and generate a bipartite biotoken with H embedded within. This biotoken is sent to the signature server. The signature server will generate a local bipartite biotoken from its base biotoken for the user, and match the bipartite biotoken it receives. If the two biotokens match, the hash H for M is released. The signature server then signs and sends a biotoken signature back to the sensor, where the signature is validated, and an audit log may be appended with server information.

The use of a biometrically-validated signature server is a novel approach to addressing the man-in-the-middle attack. For example, consider the situation of a digital certificate. According to the PKI protocol, a certificate authority signs a certificate for an entity, thus providing integrity for verification upon receipt of the certificate. Unfortunately, as we have noted earlier, end-users routinely disregard certificates, ignoring mismatched, self-signed, expired, or, in the worst case, comprised certificates. A man-in-the-middle can leverage any of these bad certificates to attempt to the fool the end-user. By using the signature server, we ensure that only a server possessing the legitimate biotoken data can generate a valid signature. Thus, in our protocol, a biotoken mis-match would force a termination of the transaction, as the invalid signature is immediately noted by the sensor side. Moreover, strong non-repudiation is introduced, allowing us to show that a message has been signed requiring the presentation of Alice's biometric data and associated biotoken keys.

6. BIO-KERBEROS

Kerberos [15] is a popular trusted third-party authentication protocol designed to grant access throughout a network. Windows 2000 presents all users with Kerberos as an authentication protocol by default. In the classic Kerberos model, a client c, wishing to authenticate with some server s on the network, collects then hashes the user's ID u and password p and contacts a Kerberos Authenti-

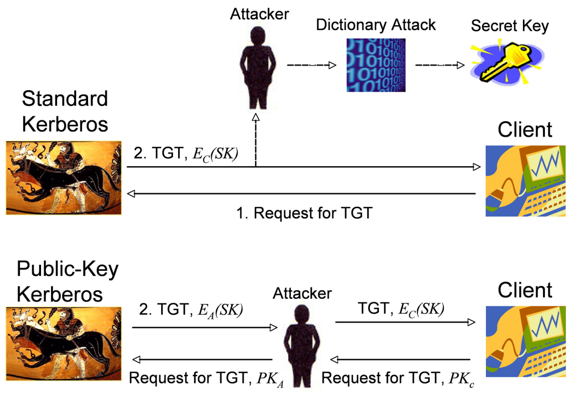


Fig. 7. A straightforward attack against Kerberos has the attacker performing a dictionary attack against the encrypted session key SK to guess the shared password (top), while the attack against Public-Key Kerberos has a man-in-the-middle substituting his own keys for the client's when communicating with the AS (bottom).

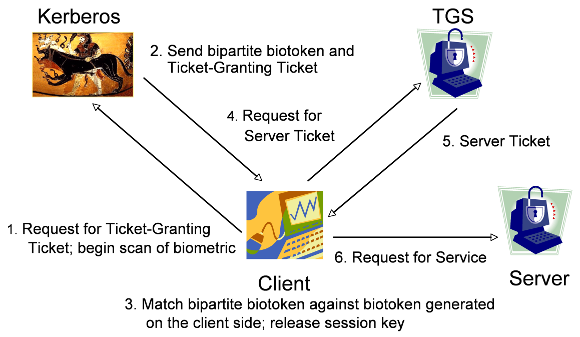

Kerberos
TGS
2. Send bipartite biotoken and Ticket-Granting Ticket
4. Request for Server Ticket
5. Server Ticket
1. Request for Ticket-Granting Ticket; begin scan of biometric
Client
6. Request for Service
Server
3. Match bipartite biotoken against biotoken generated on the client side; release session key

Fig. 8. Bio-Kerberos Authentication Steps.

cation Server (AS), requesting a Ticket-Granting Ticket tgt for a Ticket-Granting Service tgs. The AS looks up the user's hashed password u, and uses that as the session key to encrypt and return the tgt to the client. Only a client in possession of the correct secret key, i.e. the hashed password, will be able to decrypt the session key that will be used to obtain the ticket to communicate with the tgs. This pair of transactions completes the authentication stages of Kerberos. There are additional steps after authentication, but they are not impacted by the use of the biometrics, so are not described herein. It is important to note that in classic Kerberos the password is playing a dual role — it both authenticates the client and serves as the first session encryption key to protect the ticket. It is also important to note that the password (raw or hashed) is *never transmitted*, thus removing the potential of a man-in-the-middle attack.

Several weakness exist in the Kerberos scheme. With the use of passwords as secret keys, it is possible for an attacker to collect ticket messages and attempt to decrypt them with a simple password/dictionary brute-force attack. Because the user's password is used for the encryption, the attacker has a good chance of decrypting a ticket if he has access to a sufficiently large pool of tickets from different users, as some users inevitably choose weak passwords. Further, the notion of trust at the Kerberos server relies solely on stored secret keys to verify the identity of an authentication request. Verifying the authenticity of a client with such limited information is not an option.

Public-Key Kerberos [19] has been proposed as a solution to the aforementioned attacks. Without the need for a shared secret key, a session key can be generated by the AS, and encrypted with the client's public key. Thus, only a client possessing the corresponding private key can decrypt the session key. While Public-Key Kerberos solves the problems associated with shared secrets, it does not address the man-in-the-middle attack (see Fig 7, bottom). [20] presents this very attack against Public-Key Kerberos, with an attacker situated between the client and AS providing his own public key to the AS, allowing for the decryption of the ticket data with the attacker's private key, and the subsequent re-encryption of the ticket information with the client's public key. This attack works because there is no authentication between requesting and ticket-granting entities on the network for Kerberos protocol actions - public keys are simply passed around. [20] goes on to suggest a digital signature scheme to verify the source of the received data at the AS. But this brings us back to the issues we had with traditional digital signatures - we can only show that the data was signed with the user's private key.

Biometrics, and BioAPI have been used to augment the standard password used by Kerberos. However, unlike a password, where the hash is one-way with exact matching and dual use, the biometrics must be encrypted, shared, and decrypted to match. Since the traditional biometric cannot play the dual role of authentication and encryption key, using biometrics for Kerberos authentication requires secure exchange for the encryption key, which, as we have already seen, is a known problem subject to man-in-the-middle attacks. Not surprisingly, we could not find details of the key exchange process used by commercial biometric login vendors; they seem to be relying on security through obscurity. Using the bipartite biotokens as the core for the communication between the client and the Kerberos authentication server, we can address this problem in a open yet secure manner.

In our Bio-Kerberos protocol, the trusted third-party (AS) releases bipartite biotokens to the client, which will perform the matching and secret release. Figure 8 depicts this protocol variation, with changes to the original Kerberos protocol appearing in steps 1-3. The client initiates the protocol by requesting a Ticket-Granting Ticket and then begins scanning the biometric data (reducing overall latency through concurrency). The Kerberos authentication server generates a transaction ID, t, and sends it to the client. The servers generates a random nonce n for the session key. It generates the standard tgt, and uses that session key to encode the tgt. It then generates a bipartite biotoken from the stored biometric data for user u, using parameter t, encoding n into the transactional biotope. The bipartite biotope $B_{c,t,n}$ in plain text followed by the encrypted tgt ticket are sent to the client. The client matches the biotoken, $B'_{c,t}$, it has just generated against $B_{c,t,n}$. If the tokens match, n is released and used to decrypt the tgt. The remaining steps are identical to the original Kerberos protocol. In our Bio-Kerberos protocol the bipartite biotoken play the role of both authentication and secure key-transmission, and, unlike prior biometric-based solutions, prevents man-in-the-middle or phishing attacks.

7. CONCLUSION

In this paper, we have reviewed the problems that have persistently plagued traditional cryptographic protocols. Man-in-the-middle and phishing attacks remain successful because of the nature of key exchanges and user interaction with cryptographic protocols. Revocable biotokens allow for multiple nestings from other biotoken encodings, something no other privacy enhanced biometric technology has been shown to do. Using nested revocable biotokens as a basis, we have extended familiar protocols using bipartite biotokens to support the release of an embedded secret upon biometric matching. Not only can we facilitate transactions with this new biotoken, but remote digital signatures and an enhancement to Kerberos as well. Multi-factor solutions make an attacker's job more difficult, at often little expense to the end-user. The Biotope® revocable biotoken, supporting all of the capabilities presented in this paper, is one example of such technology developed by Securics, Inc. Our work continues to analyze the strengths and weaknesses of existing systems, incorporating new and innovative multi-factor solutions as viable enhancements.

8. REFERENCES

[1] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, and V. Kumar, "Biometric Encryption," in *Chapter 22 of the ICSA Guide to Cryptography*, R.K. Nichols, Ed. 1999, McGraw-Hill.

[2] A. Adler, "Vulnerabilities in Biometric Encryption Systems," in *Audio- and Video-Based Biometric Person Authentication*, 2005, pp. 1100–1109.

[3] W. Scheirer and T. Boult, "Cracking fuzzy vaults and biometric encryption," in *In Proc. of the 2007 Biometrics Symposium, held in conjunction with the Biometrics Consortium Conference (BCC 2007), , Baltimore, MD.*, 2007.

[4] N. Ratha, S. Chikkerur, J. Connell, and R. Bolle, "Generating Cancelable Fingerprint Templates," *IEEE TPAMI*, vol. 29, no. 4, pp. 561–572, 2007.

[5] T. Boult, W. Scheirer, and R. Woodworth, "Secure Revocable Finger Biotokens," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[6] A. Juels and M. Sudan, "A Fuzzy Vault Scheme," in *Proc. of the IEEE Intl. Symposium on Information Theory*, 2002, p. 408.

[7] Preda Mihailescu, "The Fuzzy Vault for Fingerprints is Vulnerable to Brute Force Attack," 2007.

[8] W. Chang, R. Shen, and F. W. Teo, "Finding the Original Point Set Hidden Among Chaff," in *Proc. of the ACM Symposium on Information, Computer And Communications Security*, 2006, pp. 182–188.

[9] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors," in *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, P. Tuyls, B. Skoric, and T. Kevenaar, Eds., chapter 5, pp. 79–99. Springer-Verlag, 2007.

[10] L. Ballard, S. Kamara, and M. Reiter, "The Practical Subtleties of Biometric Key Generation," *Technical Report TR-JHU-SPAR-BKMR-090707, Johns Hopkins Univeristy*, 2007.

[11] T. Boult, "Robust Distance Measures for Face Recognition Supporting Revocable Biometric Tokens," in *In Proc. of the 7th IEEE International Conference on Automatic face and gesture recognition, Southampton, UK*, 2006.

[12] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure Remote Authentication Using Biometrics," in *In Proc. of EUROCRYPT, Aarhus, Denmark*, 2005.

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), May 2008.

[14] C. Watson, M. Garris, E. Tabassi, C. Wilson, R. McCabe, and S. Janet, "User's Guide to NIST Fingerprint Image Software 2," 2002.

[15] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc., 1996.

[16] T. Kwon, H. Lee, and J. Lee, "A Practical Method for Generating Digital Signatures Using Biometrics," *IEICE Transactions on Communications*, vol. 90, no. 6, pp. 1381–1389, 2007.

[17] J. Jo, J. Seo, and H. Lee, "Biometric digital signature key generation and cryptography communication based on fingerprint.," in *FAW*, Franco P. Preparata and Qizhi Fang, Eds. 2007, vol. 4613 of *LNCS*, pp. 38–49, Springer.

[18] S. Campbell, "Supporting Digital Signatures in Mobile Environments," in *Proc. of the 12th IEEE International Workshop on Enabling Technologies (WET 2003)*, 2003, pp. 238–242.

[19] L. Zhu and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)," RFC 4556 (Proposed Standard), June 2006.

[20] I. Cervesato, A. Jaggard, A. Scedrov, J. Tsay, and C. Walstad, "Breaking and Fixing Public-Key Kerberos," *Information and Computation*, vol. 206, no. 2-4, pp. 402–424, 2008.