Preventing Guessing Attacks Using Fingerprint Biometrics

Sreekanth Malladi, Jim Alves-Foss
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID - 83844
{msskanth,jimaf}@cs.uidaho.edu

Sreenivas Malladi
Satyam Computers Private Ltd.
Hyderabad - 500044, A.P
India.
Sreenivas_Malladi@satyam.com

***Abstract*— Security protocols involving the use of poorly chosen secrets, usually low-entropy user passwords, are vulnerable to guessing attacks. Here, a penetrator guesses a value in place of the poorly chosen secret and then tries to verify the guess using other information. In this paper we develop a new framework extending strand space theory in the context of these attacks to analyze the effect using fingerprint biometrics in those protocols. In particular, we will prove the efficacy of biometrics in preventing some known forms of guessing attacks which differ in the way the guess is verified. Interestingly, our approach shows a remarkable increase in security of selected protocols, subject to off-line guessing attacks. We illustrate these concepts on some examples.**

***Index Terms*—security protocols, guessing attacks, weak secrets, passwords, fingerprint biometrics.**

I. Introduction

Entity authentication and Authenticated Key Exchange are two important security functions that security protocols are often designed to achieve. User authentication is achieved through three approaches: Something that a user,

1) *knows* (eg. password)
2) *has* (eg. ID card)
3) or *is* (eg. fingerprint or retinal scan).

The first of the three—based on user's knowledge is the most popular, due to its simplicity, adaptability, minimum hardware requirement and social resistance. It is more convenient for users since they can move without having to carry hardware tokens with them, as in the second approach (token-based authentication).

Although the third approach (biometrics) seems to offer intriguing person-authentication benefits, they cannot be used directly in place of passwords, because unlike passwords, they are not secrets. For example, facial images can be directly downloaded from the web and fingerprints can be easily extracted from stolen laptops.

However, using knowledge based approach has inherent disadvantages due to the low entropy often observed in user-chosen passwords [1], [2]. These low-entropy passwords make a security protocol that is using them, vulnerable to *guessing attacks* [3]. Here, a penetrator guesses a value in place of the weak secret or password in a protocol and then seeks to verify the guess using other information presented by the protocol or the system that is implementing the protocol.

Restricting the users to use passwords with high entropy, such as machine-generated, are usually discouraged, so as not to trouble them by forcing to remember those obscure passwords. For example, ATMs (automatic teller machines) usually operate with user PINs of only four to six digits.

Hence, the emphasis is on improving the quality of the protocols using weak secrets, making them more resistant to guessing attacks.

In this paper, we analyze the effect of using fingerprint biometrics in protocols that are vulnerable to guessing attacks. In particular, we will consider introducing biometrics into protocols as message elements, taking advantage of their inherent properties of redundancy and uniqueness. To this end, we develop a new framework extending strand space theory in the context of guessing attacks. We consider various forms of guessing attacks, model those actions in the framework and then develop a systematic procedure to analyze protocols within this framework.

In the next section (2) we will present some background on guessing attacks and fingerprint biometrics. In section 3 we will explain the concepts in our framework. Specifically, we will introduce strand spaces and present the various forms of guessing attacks that we would consider. Then we will show how we model guessing attacks in our framework and present a systematic way of analyzing a protocol for guessing attacks within our framework. In section 4 we illustrate our concepts on the Mellovin and Berritt protocol [4] as an example and end with a discussion (section 5).

II. BACKGROUND

A. Guessing Attacks

A guessing attack on a protocol is feasible if the protocol involves the use of weak secret(s) and presents enough information, allowing the process of verification of a guess (in place of a weak secret) to finish within a reasonable amount of time and effort.

As an example of a guessing attack, consider the following simple protocol:

Msg 1. $a \rightarrow s : a$
Msg 2. $s \rightarrow a : n_s$
Msg 3. $a \rightarrow s : \{n_s\}_{passwd(a,s)}$

Here, user a aims to authenticate itself for a connection to server s. (n_s is a nonce and $\{m\}_k$ represents m encrypted with k).

Now an attacker observing these communications might guess a value for $passwd(a, s)$ and encrypt n_s in message 2 with that guess. He can then compare it with message 3. For example, if the user is one of the authors of this paper, then the attacker might guess "`sreek123`" as the password and then compare $\{n_s\}_{passwd(a,s)}$ with $\{n_s\}_{\texttt{sreek123}}$. A successful comparison might indicate with high probability that this might be the user's password.

He can automate this procedure as well by trying passwords from a list, possibly compiled from a suitable dictionary, repeating it a number of times. Of course, if he repeats the process on-line, then the server might notice those attempts, log them, alarm the user, and mount additional counter measures like, shutting down the connection etc.

In this paper, we address only those guessing attacks that are feasible to launch entirely off-line, where failed attempts are undetectable.

Although the above-presented example is on a simple protocol, guessing attacks have been found even on complex protocols, often in different forms. Also, the verification and guess can be done in a cascading way, with a vulnerable key making it feasible to break an otherwise strong key occurring else where in the protocol.

Gong *et. al* [3] suggest using "confounders", which are redundant random values to be included before encryption in crucial messages. In a sense confounders act as one-time pads, making it infeasible to verify a guess.

It is interesting to see how other techniques having properties that can foil verification attempts be used in a similar way. Typically, the techniques should make important messages used for verification of the guess to contain sufficient redundancy so that any verification attempt is computationally infeasible [5]. As we will see, biometrics are ideal candidates to consider for use in such situations. Often they contain "unforgeable" information to properly identify and authenticate an individual but at the same time, sufficient redundancy that frustrates guessing attempts.

B. Fingerprint Biometrics

Biometrics are values that encapsulate certain physiological or behavioral characteristics of the human body that are relatively unique from individual to individual (eg. fingerprint, hand geometry, signature, voice). Although biometrics offer solid person-authentication benefits, improper integration of biometrics into security systems was often found to be resulting in little or no increase in security. Most importantly, they should not be used in place of secrets like passwords, since anybody with access to a person can easily procure biometrics from him/her. Hence, there is definitely a need to examine and analyze the behavior of systems when biometrics are used to improve security.

In this paper we limit our discussion to fingerprint biometrics and analyze their effect when they are introduced into security protocols, particularly, in resisting guessing attacks. Fingerprint biometrics are the most popular and hold the most promise among all other biometrics (eg. iris pattern, retinal scan, voice patterns) in increasing the knowledge level (versus trust) in a protocol. This is mainly due to the maturity of the technique, minimal invasiveness to privacy and potentially the most socially acceptable of all biometrics.

Fingerprint biometrics is a pattern of ridges and furrows on the surface of the finger that composes unique individual patterns defined as ridges, whorls, archs or collectively as *minutiae*. These minutiae provide a unique pattern for fingerprint sensors which can be digitally stored as a template for each individual. Users are required to input their fingerprint each time they request access to the system. This is called a *live scan*. (The exact implementation details and hardware requirements are out of scope of our discussion).

Kovacs-Vajna in [6] and Jain *et. al* in [7], [8] address the issue of minutiae extraction. Jain *et. al* suggest that, to identify and verify an individual, one may not need the total number of minutiae in a fingerprint. Supportingly, one can deduce from Kovacs-Vajna's article that 40 to 60 minutiae from 500 to 800 may provide sufficient identification.

These attributes provide interesting and useful properties to use in a variety of situations. For example, we did a statistical analysis on the above numbers to determine the

chances of replay from the same subset. We find that it would take approximately from 10^{59} to 10^{81} years to statistically find a match during a session[1]. This obviously provides enough protection from an electronic capture of a fingerprint image by a penetrator and replaying or spoofing by guessing at the possible subset for the session.

To illustrate how these properties are useful in the context of guessing attacks, consider the following simple protocol presented in [3]:

Msg 1. $a \rightarrow s : \{n\}_{k1}$
Msg 2. $s \rightarrow a : \{f(n)\}_{k2}$

with $f(n) = n + 1$, $k1$ a public key and $k2$, a weak key derived from the user password.

Now, an attacker can attempt an offline guessing attack on this protocol. He can initially guess a value for $k2$ and decrypt message 2 to obtain some value for $f(n)$ (say x). He can then encrypt $x - 1$ with public key $k1$ and match it with $\{n\}_{k1}$ in message 1 to verify the guess.

To illustrate how biometrics can help prevent this attack, let $\mathsf{BIOMETRIC_A}$ represent the fingerprint template[2] for an individual A. Also let, $biometric_A$ be a typical live scan for A. Clearly, $biometric_A \in \mathsf{BIOMETRIC_A}$ when A is identified to a system having $\mathsf{BIOMETRIC_A}$.

Now let, $n = biometric_a$. i.e. a user in role 'a' inputs a live scan of his fingerprint in place of n. Let $f(n) = bio_a$, where bio_a represents the subset of minutiae that the server has extracted and used for identification from the total set of minutiae (ignoring the remaining minutiae positions). For example, if $\mathsf{BIOMETRIC_a}$ consists of 500 minutiae then bio_a may consist of 40 minutiae derived from that set, used for identification of a.

So now the protocol becomes,

Msg 1. $a \rightarrow s : \{biometric_a\}_{k1}$
Msg 2. $s \rightarrow a : \{bio_a\}_{k2}$

An intruder attempting to attack the protocol as above may obtain some value in place of bio_a. However, it is computationally infeasible for him to guess at the possible set of minutiae ($biometric_a$) using which, this subset was created. As mentioned above, he would have to spend time in the order of $10^{59}-10^{81}$ years to find a matching set of $biometric_a$. Thus, the guessing attack is successfully prevented.

The technique prevents the attack due to the fact that, replacing f with the extracted minutiae, bio_a, makes it an excellent replacement for f. Contrast this with the previous case—where f was a function whose inverse was easy to compute. In a way, conversion of $biometric_a$ to bio_a can be considered as a computationally irreversible, one-way transformation.

Since we will often use this property, we frame it in the following assumption:

Assumption 1: For any user 'A', it is computationally infeasible to determine $biometric_A$ from bio_A.

Put in other words, the assumption states that, it is impossible to find a matching set of the fingerprint minutiae from which a subset of minutiae, bio_A was extracted and used to identify A. Observe that this has some facets—Not only that it is considered infeasible to find a matching set of minutiae, but also, the "exact" set of minutiae from which bio_A was extracted. Since many mappings of minutiae in $\mathsf{BIOMETRIC_A}$ might correspond to bio_A, an attacker would still need to find the original set of minutiae from which bio_A was obtained, in a reasonable amount of time—From the values we obtained through our calculations, we consider this operation as well, to be, computationally infeasible.

III. The Frame Work

In this section we will show how we adapt the existing strand space framework [9] to analyze guessing attacks. We will first introduce strand spaces and then illustrate the various possible forms of these attacks. We will then model these actions in our model and use them to analyze protocols subject to guessing attacks. In particular, we will study possible penetrator paths from bundles (sequences of penetrator actions) corresponding to the forms of guessing attacks.

To start with, let **Fact** denote the set of all possible elements in a protocol[3] defined as,

$$Fact ::= Atom \mid \mathsf{JOIN}\ Fact\ Fact \mid \mathsf{ENCR}\ Fact$$

Atom is the set of atomic values (eg. $Alice$, Bob, N_A, $\mathsf{PubKey}(A)$ etc.) assumed to contain in a protocol. We will adopt fairly standard notation—JOIN and ENCR represent concatenating two data items and encrypting a data item respectively. When two data items a, b are to be concatenated, we will write, $a\ .\ b$ or (a, b). When a data item a is to be encrypted with a key k, we will write $\{a\}_k$. Also, we will denote the inverse of a key k as, k^{-1}.

When we talk about the first or second component in a fact with two components we will ue subscripts "1" and

[1]Calculations found by using `www.io.com/~ritter/JAVASCRIP/PERMCOMB.HTM#Permutations` calculator for permutations and combinations. Range determined by use of 40 to 60 possible combinations from totals range of 500 to 800

[2]From here on, we will use "biometrics" to mean "fingerprint biometrics".

[3]with 'message' referring to the entire collection of facts sent in a protocol step.

"2" as:
$(f1, f2)_1 \widehat{=} f1, (f1, f2)_2 \widehat{=} f2$

Also, the $subfact$ relation $\sqsubset$ is defined as the smallest relation on facts such that,
$f \sqsubset f$;
$f \sqsubset \{f'\}_{k'}$ iff $f \sqsubset f'$;
$f \sqsubset (f1, f2)$ iff $f \sqsubset f1 \vee f \sqsubset f2$.

Definition 1: A *strand* is a sequence of communications by any agent in a protocol run, represented as $\langle \pm f1, \pm f2, \ldots, \pm fn \rangle$. Each node in the set of nodes $\mathcal{N}$, receives (represented as $-$) or transmits (represented as $+$) a fact (fi) and belongs to a unique strand.

1) An edge $\Rightarrow$ is drawn between all consecutive nodes on the same strand.
2) An edge $\rightarrow$ is drawn between nodes belonging to different strands, if one node transmits a fact and the other node receives the same fact.
3) A strand space Σ is a directed graph with all the nodes in $\mathcal{N}$ as vertices and $(\rightarrow \cup \Rightarrow)$ as edges.

A *bundle* represents partial or complete history of the network. Let C be a bundle and $(\rightarrow_C \cup \Rightarrow_C)$ be a finite set of edges. Then,

1) If $n2 \in \mathcal{N}_\mathcal{C}$ receives a fact, then there exists a unique $n1$ with $n1 \rightarrow n2$.
2) If $n2 \in \mathcal{N}_\mathcal{C}$ with $n1 \Rightarrow n2, \exists\, n1 \Rightarrow_C n2$;
3) C is acyclic.

A node n is an *entry point* to a set of facts $\mathbf{F}$, if there is no node previous to n transmitting a fact in $\mathbf{F}$. A fact *originates* on n if n is an entry point to all possible facts. A fact is *uniquely* originating in a bundle if it does not originate on any other node in the bundle.

The penetrator is assumed to possess some message elements, $\mathbf{M_P}$ and keys $\mathbf{K_P}$. We now define penetrator strands:

Definition 2: A *penetrator strand* is one of the following:

M	***Text message*** $\langle +f \rangle$ with $f \in \mathbf{M_P}$.
F	***flushing*** $\langle -f \rangle$.
T	***Tee*** $\langle -f, +f, +f \rangle$.
C	***Concatenation*** $\langle -f_1, -f_2, +f_1 f_2 \rangle$.
S	***Separation*** $\langle -f_1 f_2, +f_1, +f_2 \rangle$.
K	***Key*** $\langle +k \rangle$ with $k \in \mathbf{K_P}$.
E	***Encryption*** $\langle -k, -f, +\{f\}_k \rangle, k \in \mathbf{K_P}$.
D	***Decryption*** $\langle -k^{-1}, -\{f\}_k, +f \rangle, k \in \mathbf{K_P}$.

We will later extend these standard capabilities of the penetrator by adding additional powers to guess and verify values from a protocol run. First, we will define the possible forms of guessing attacks:

A. Guessing Attack Forms

In this section we will present the various forms of guessing attacks that we consider to analyze the efficacy of fingerprint biometrics. This is not an exhaustive list of guessing attacks but consists of the most widely known forms of it (obtained from [3], [10]). However, the list serves very well for our purpose of illustrating the efficacy of using fingerprint biometrics.

Let $\mathbf{S}$ be the set of all secrets and $\mathbf{W}$, the set of weak secrets (typically, user-chosen passwords). Also, let v be the verifier in the set of verifiers, $\mathbf{V}$ and g be a guess in the set of guesses, $\mathbf{G}$.

For example, in the example presented in section 1.1, $v = n_s$ and $passwd(a, s) \in \mathbf{W}$.

Note that, $\mathbf{M_P} \cap \mathbf{V} \subset \mathbf{Fact}, \mathbf{K_P} \cap \mathbf{S} = \phi$ and $\mathbf{W} \subset \mathbf{S}$. In addition, we will denote $\doteq$ as a binary relation that compares any two facts, and returns `true` if there is a match or a `false` otherwise. Let $w \in \mathbf{W}$.

Now, the penetrator may,

1) know $v, \{v\}_w$ and do: $\{v\}_g \doteq \{v\}_w$
   (or $v \doteq \{\{v\}_w\}_{g^{-1}}$);
2) Have $\{(v1, v2)\}_w, \{v1\}_w$,
   Do: $\{\{v1\}_w\}_{g^{-1}} \wedge \{\{(v1, v2)\}_w\}_{g^{-1}}$,
   $v1 \doteq (v1, v2)_1$
   Observe that[4], the comparison is also possible in a slightly different form if he has $\{v2\}_w$:
   $v2 \doteq (v1, v2)_2$.
3) Have $\{g\}_w$ and Do $\{g\}_g \doteq \{g\}_w$; (i.e. the protocol might itself encrypt the guess g with the weak password, w)
4) Has $\mathsf{PK}(a), \{\mathsf{SK}(a)\}_{passwd(a)}$ (public key of a and private key of a, encrypted with $passwd(a)$. These two elements are included for any user in `/etc/publickey` under `SunOS 4.0`):
   Do: $\{\{\mathsf{SK}(a)\}_{passwd(a)}\}_{g^{-1}}$; He can then encrypt an arbitrary string y with this value and decrypt it with the public key, $\mathsf{PK}(a)$. If he gets back y, that verifies the guess.

B. Modelling Actions

We will now model all the actions in the previous section within the strand space model. Specifically, we will add strands **G1, G2, G3, G4** corresponding to each of the forms presented above to the penetrator strands defined in definition 2.

1) Now, consider the first form above. The corresponding strand for this action can be represented as: $\mathbf{G1} = \langle -v, -\{v\}_w, -g, +\{v\}_g \rangle$.
   The scenario can be visualized as in figure 1.

[4]Recall that subscripts '1' and '2' return the first and second elements respectively, from a fact with two elements.

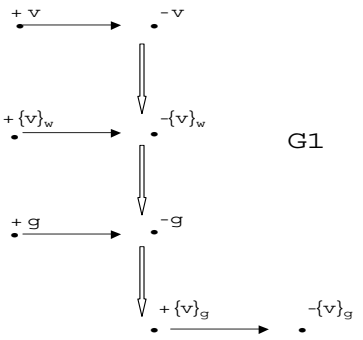

Fig. 1. G1 Strand

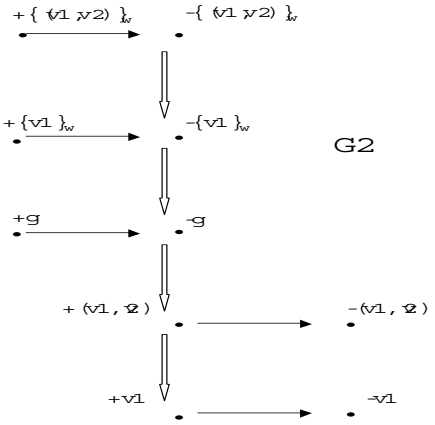

Fig. 2. G2 Strand

2) Similarly, the second form above can be represented as:
   $\mathbf{G2} = \langle -\{(v1, v2)\}_w, -\{v1\}_w, -g,$
   $+(v1, v2), +v1\rangle$
   This can be visualized as in figure 2.
3) Similarly, $\mathbf{G3} = \langle -g, -\{g\}_w, +\{g\}_g\rangle$, and
4) $\mathbf{G4} = \langle -\mathsf{PK}(a), -\{\mathsf{SK}(a)\}_w, -g,$
   $+\{\{\mathsf{SK}(a)\}_w\}_{g^{-1}}\rangle$,

From any given bundle, we can construct $\mathbf{G}$ strands using the set of facts in the bundle. It is not necessary that the required facts to construct a particular $\mathbf{G}$ strand need to be readily present. However, we can determine if the penetrator can deduce the required facts using a sequence of penetrator strands defined in definition 2. For example, to construct a $\mathbf{G1}$ strand, a bundle needs to have v and $\{v\}_w$, but the bundle may have (v, v') and $\{v\}_w$. However, we can construct this initially as an $\mathbf{S}$ strand and then use v to match it with $\{\{v\}_w\}_{g^{-1}}$ (or $\{v\}_g$ with $\{v\}_w$).

We will now give a formal definition for a guessing attack. We first define a set $\mathbf{X}$ of "verifiable" facts. The term "verifiable" means that the term can be used to compare and verify a guess. Typically, these are terms encrypted with a weak secret or a weak password[5]. The below definition then says, that a guessing attack is $feasible$ on a bundle if, the probability that the guess is correct, given that the verifier matches it's recorded value—is sufficiently large.

The phrase "sufficiently large" can be equated to a threshold probability, ϵ where ϵ gives a measure of a system's tolerance to off-line guessing attacks. When the threshold probability ϵ is equal to 1, a comparison between a verifier and it's recorded value immediately implies that the guess corresponds to the weak secret. This is the ideal scenario. However, in the worst case, if $\epsilon = 1/n$, where n is the total number of plausible passwords, then the attacker is forced to make some on-line attempts in order to verify the guess and hence an off-line guessing attack is considered infeasible. On the other hand, if ϵ ranges in between 1 and $1/n$ then the attacker can cut down the range of possible values significantly and so an off-line guessing attack is considered feasible.

Definition 3: Let C be a bundle, $\mathbf{W}, \mathbf{G}, \mathbf{V}, \mathbf{S}, \mathbf{X}$ be as previously defined, x' be the recorded value of a verifiable fact $x \in \mathbf{X}$ and $g \in \mathbf{G}$ be a guess in place of a weak secret $w \in \mathbf{W}$. Let $Pr(A \mid B)$ represent the probability of event A being `true` given that B is `true`. Then, a guessing attack is said to be *feasible* on C *iff*,

$$Pr(g = w \mid x = x') > \epsilon$$

where, ϵ is the threshold probability for the system under consideration.

This definition is parameterized by, g, w, $\mathbf{W}$, $\mathbf{G}$, $\mathbf{V}$, $\mathbf{S}$, $\mathbf{X}$ and ϵ.

C. Proof Methodology

We will now present a recipe that we would follow to construct a proof that a given technique prevents guessing attacks on a protocol. Specifically, for any given bundle, we show that certain messages cannot be constructed and hence the corresponding guessing strands that use those messages. For the remaining guessing strands, we show that it is infeasible to perform the verification step.

To be precise, we follow the steps below:

Step 1. Try all possible deduction steps to obtain facts corresponding to $\mathbf{G}$ strands. i.e. Find facts using the subfact relation $\sqsubset$ to correspond to the forms of facts found in those strands.

For example, for $\mathbf{G1}$ strand, we need to obtain v and $\{v\}_w$ in order to construct the strand. We examine all possible message forms that can be reduced to facts of

[5]or an otherwise strong key made weak through a cascading effect. eg. see section IV.

this form. This step would also ensure proving that facts of certain form are unattainable and hence impossible to construct some $\mathbf{G}$ strands.

Step 2. Next, having constructed all the possible $\mathbf{G}$ strands from the bundle, consider the verification step for that strand. For example, for $\mathbf{G1}$ strand, $\{v\}_w \doteq \{v\}_g$.

Step 3. Prove that the verification step is infeasible when the technique is implemented. In other words, prove that a guessing attack is infeasible (wrt definition 3) by showing that the probability of a match in the verification step is sufficiently low.

IV. An Example

We will now illustrate the concepts in the previous sections with Mellovin and Berritt protocol [4] as an example:

Msg 1. $a \rightarrow b : \{pk\}_{passwd(a,b)}$
Msg 2. $b \rightarrow a : \{\{k\}_{pk}\}_{passwd(a,b)}$
Msg 3. $a \rightarrow s : \{n_a\}_k$
Msg 4. $s \rightarrow a : \{(n_a, n_b)\}_k$
Msg 5. $a \rightarrow s : \{n_b\}_k$

In this protocol, pk is an asymmetric key. Lowe [10] changed it to a symmetric key and discovered an attack:

An attacker can guess $passwd(a, b)$. He then decrypts message 1 with the guess to obtain pk. He decrypts message 2 as well with the same guess to obtain $\{k\}_{pk}$ and decrypts this with pk obtained from message 1. Thus he obtains some value for key k. Now he decrypts message 3 and 4 with k and compares n_a with $(n_a, n_b)_1$ to verify the guess. Alternatively, he can decrypt message 5 to obtain n_b and compare it with $(n_a, n_b)_2$ to verify the guess.

We will now prove that this protocol remains secure even if pk is a symmetric key, when biometrics are used. We will replace n_a in message 3 with $biometric_a$ and n_a in message 4 with bio_a. Also, we will replace n_b in message 4 with $biometric_b$ and n_b in message 5 as bio_b. So now this protocol becomes,

Msg 1. $a \rightarrow b : \{pk\}_{passwd(a,b)}$
Msg 2. $b \rightarrow a : \{\{k\}_{pk}\}_{passwd(a,b)}$
Msg 3. $a \rightarrow s : \{biometric_a\}_k$
Msg 4. $s \rightarrow a : \{(bio_a, biometric_b)\}_k$
Msg 5. $a \rightarrow s : \{bio_b\}_k$

We will now prove that no guessing attacks of the form presented in III-A can succeed on this protocol.

Proposition 1: Let C be a bundle formed by executing the above protocol using biometrics. Then, no guessing attack on C using $\mathbf{G}$ strands is feasible wrt def 3.

Proof:

We will first study the set of facts and verifiable facts (set $\mathbf{X}$) that can be constructed from C using the subfact relation, $\sqsubset$. We will then consider all the possible $\mathbf{G}$ strands that can be constructed using $\mathbf{X}$. Finally, we will show that the verification steps in those strands are infeasible.

$passwd(a, b)$ is considered as the only weak secret in this protocol since it is derived from a user password. Hence, we consider all the facts that can be derived when a guess g is used in place of $passwd(a, b)$. Firstly, observe that, an attacker can derive pk from $\{pk\}_{passwd(a,b)}$ since pk is encrypted with $passd(a, b)$ and $pk \sqsubset \{pk\}_{passwd(a,b)}$. Similarly, he can derive

$\{k\}_{pk}$ since, $\{k\}_{pk} \sqsubset \{\{k\}_{pk}\}_{passwd(a,b)}$,
k since, $k \sqsubset \{k\}_{pk}$,
$biometric_a$ since, $biometric_a \sqsubset \{biometric_a\}_k$,
$(bio_a, biometric_b)$ since,
$(bio_a, biometric_b) \sqsubset \{(bio_a, biometric_b)\}_k$, and
bio_b since $bio_b \sqsubset \{bio_b\}_k$.

Hence, corresponding to a guess in place of $passwd(a, b)$, the corresponding terms that can be derived are:
pk, $\{k\}_{pk}$, k, $biometric_a$, $(bio_a, biometric_b)$ and bio_b.
The verifiable facts (set $\mathbf{X}$) in C, however, are:
$\{biometric_a\}_k$, $\{(bio_a, biometric_b)\}_k$, and $\{bio_b\}_k$.

We will now consider all the possible $\mathbf{G}$ strands that can be constructed using these verifiable facts.

1) $\mathbf{G1}$ — Since, the set $\mathbf{X}$ obtained from C does not contain facts of the form, v, $\{v\}_w$, no $\mathbf{G1}$ strands can be constructed. It is interesting to ask, why $pk, \{k\}_{pk}$ cannot be used to construct a $\mathbf{G}$ strand. However, observe that obtaining k by decrypting $\{k\}_{pk}$ with pk does not give him any verification of the guess for $passwd(a, b)$ because he has not seen k before.
2) Similarly, it is straightforward to see that $\mathbf{G3}$ and $\mathbf{G4}$ strands cannot be constructed using $\mathbf{X}$.
3) However, we can construct $\mathbf{G2}$ strand using the set $\mathbf{X}$ as:
   $\mathbf{G2} = \langle -\{biometric_a\}_k, -\{(bio_a, biometric_b)\}_k, +biometric_a, +(bio_a, biometric_b)\rangle$ and the comparison— $biometric_a \doteq (bio_a, biometric_b)_1$

Observe that, although k was initially a strong key, it would be considered a weak key through a cascading effect. Hence, it was possible to derive elements encrypted with it and construct this strand. Now, let g be the guess in place of $passwd(a, b)$ corresponding to the set $\mathbf{X}$. Considering $\mathbf{G1}$ strand, a guessing attack is feasible on C according to definition 3 if, $Pr(g = w \mid biometric_a \doteq (bio_a, biometric_b)_1) > \epsilon$. For this

to be `true`, $biometric_a \doteq (bio_a, biometric_b)_1$ should be computationally feasible and whenever a $biometric_a$ is found corresponding to bio_a, $Pr(g = w)$ should be sufficiently high ($> \epsilon$).

However, by contradiction, from assumption 1 we have $bio_a \doteq biometric_a$, is computationally infeasible. Whenever a $biometric_a$ is found such that bio_a is derived from $biometric_a$, since there are too many of them that can correspond to bio_a, $Pr(g = w)$ will be equal to a very low value, $1/n$ where, n is the total number of all such $biometric_a$. Hence, the guessing attack using strand $\mathbf{G2}$ is infeasible.

Similar proof argument can be constructed for the case when the comparison can be done as, $bio_b \doteq (bio_a, biometric_b)_2$.

■

V. Discussion

In this paper we have developed a new framework to prove the efficacy of fingerprint biometrics towards resisting guessing attacks. We have illustrated the concepts on the Mellovin and Berritt protocol as an example.

Some points are worth mentioning here. We have presented a method to achieve security goals using biometrics in an effective way, when they are present. Although in the past, other uses of biometrics were considered, this aspect of using them into security protocols in the context of guessing attacks seems to have caught little attention. Also, no attempt was made to analyze and prove their efficacy in protocols.

Next, although the idea of introducing biometrics seems powerful in resisting guessing attacks, it should not be considered a panacea. Firstly, if biometrics are used improperly in these contexts, they can show no improvement in security or even worse—might reduce security through presenting more information to an attacker. In this situation, the framework serves to formally analyze their effect when introduced into protocols.

Finally, although we have developed the framework to analyze the efficacy of fingerprint biometrics, the framework is also useful to study and analyze other similar techniques—in general, to prove the correctness of protocols in presence of guessing attacks. Also, our approach is stronger than Gong *et. al*'s [3], since they consider only fixed penetrator knowledge, whereas, as in [10], we consider all facts reducible to verifiable facts.

An extension to this work is related to considering the efficacy of biometrics other than fingerprints in protocols. Another extension would be to analyze the effect of biometrics in terms of other forms of guessing attacks. Although we have considered all known forms of guessing attacks, it is hard to be sure that there are'nt any others. Typically, for each of those attacks, we would have to construct corresponding $\mathbf{G}$ strands and reduce the facts in the protocol to verifiable facts of the form of corresponding $\mathbf{G}$ strands. The verification step will then have to be proved infeasible.

Acknowledgements. We would like to thank Hamid R. Arabnia and the anonymous referees of this paper. Also, special thanks to Anil K. Jain, Michigan State University, for some useful comments about the idea.

References

[1] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[2] D. Seeley. Password Cracking: A Game of Wits. *Communications of the ACM*, 32(6):700–703, June 1989.

[3] Li Gong, T. Mark A. Lomas, Roger M.Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[4] M. Merritt and S. M. Bellovin. Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Security Conference on Research in Security and Privacy*, pages 72–84, 1992.

[5] L. Gong. A Note on Redundancy in Encrypted Messages. *ACM Computer Communication Review*, 20(5):18–22, October 1990.

[6] Z. M. Kovacs-Vajna. A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, volume 22(11), November 2000.

[7] A. Jain and S. Pankanti. Biometrics: Promising Frontiers for emerging identification market. Technical Paper Number MSU-CPS-99-5, January 1999.

[8] A. Jain and S. Pankanti. Biometrics System: Anatomy of Performance. *IEICE Transaction Fundamentals*, E00-A(1), January 2001.

[9] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2,3):191–230, 1999.

[10] Gavin Lowe. Analyzing protocols subject to guessing attacks. *Workshop on Issues in the Theory of Security (WITS'02)*, January 2002.