

What are Multi-Protocol Guessing Attacks*

And

How to prevent them

Sreekanth MALLADI, Jim ALVES-FOSS
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID - 83843
{msskanth,jimaf}@cs.uidaho.edu

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

April 1, 2002

Abstract

A guessing attack on a security protocol is an attack where an attacker guesses a poorly chosen secret (usually a low-entropy user password) and then seeks to verify that guess using other information. Past efforts to address guessing attacks in terms of design or analysis considered only protocols executed in isolation. However, security protocols are rarely executed in isolation and reality is always a case of mixed-protocols. In this paper, we introduce new types of attacks called multi-protocol guessing attacks, which can exist when protocols are mixed. We then develop a systematic procedure to analyze protocols subject to guessing attacks. Using this procedure, we will present a method of deriving some syntactic conditions to be followed in order for a protocol to be secure against multi-protocol guessing attacks. Lastly, we use the strand space framework to prove that a protocol will remain secure, given that these conditions are followed, by modeling the conditions within the strand space framework. We illustrate these concepts using the Mellovin and Berritt protocol (EKE) as an example.

1. Introduction

Since people tend to choose poor passwords [17], security protocols using them are vulnerable to *guessing attacks* [9]. As an example, 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 to server s . (n_s is a nonce. $\{m\}_k$ represents m encrypted with key k).

*This work was funded in part by DARPA under grant no. MDA972-00-1-0001.

Now an attacker observing these communications can mount a guessing attack by guessing the user's password. For example, if the user is one of the authors of this paper, then he might guess "alwaysalves" as the password. He can then do $\{n_s\}_{alwaysalves}$ and compare it with message 3 that he obtained ($\{n_s\}_{passwd(a,s)}$). A successful comparison indicates with high probability that this might be the user's password.

Past efforts to address guessing attacks in terms of design [9] or analysis [15] focussed only on protocols executing in isolation. However, security protocols are rarely executed in an isolated environment, without interaction from other protocols. Some of the main reasons for "mixed" operation of protocols include:

- deliberate use of sub-protocols such as Kerberos, Neuman-Stubblebine etc. [13, 18], which use those sub-protocols for re-authentication, or
- protocols having different options and hence multiple sub-protocols [5, 12, 16] or
- accidental execution of different protocols on the user's machine (many times with the same parties, through the same communication channels and having the same message formats and/or keying material).

Together with these, re-using the same keying material (due to the high cost of certified keys), multiple uses of keying formats and keys (due to the widespread use of cryptographic APIs) and using the same password for different applications (which is a commonly observed characteristic of human chosen passwords), make a mixed environment, hostile for protocols.

Hence, the interesting questions to ask are,

- can mixing of two protocols result in new attacks that were not known to exist when either of them executes in isolation?

- each of the protocols may be easy to analyze independently. But, how should we analyze them when they are operating in a varying, mixed environment?
- what are the conditions (if there exist any) under which a protocol can operate securely, without fear of being attacked by mixing information from other protocols?

In this paper we introduce new attacks called “multi-protocol guessing attacks” which are guessing attacks that are launched when protocols are mixed. Firstly, we will present a background and motivation, together with some examples of multi-protocol guessing attacks in section 2. Next, we will give an overview of strand space framework and present a systematic procedure to analyse a protocol for guessing attacks (section 3). We will then use this procedure to derive some syntactic conditions under which, a protocol can remain secure against guessing attacks even in a mixed environment. We will model these conditions within the strand space framework and prove that, as long as these conditions are satisfied, a protocol cannot be attacked through multi-protocol guessing attacks (section 4). We sum up with a conclusion and some possible extensions for future work in section 5.

2. Background

In [1, 2, 3] we have presented multi-protocol attacks on security protocols which can exist on interleaved protocols. We have also suggested some techniques that must be adopted for a protocol to remain secure in a mixed environment. Subsequently, Guttman et. al [11] proved a useful result, that protocols are independent if they use disjoint encryption—one of our suggestions earlier to resist multi-protocol attacks. Thus, Guttman et. al virtually buried the threat of multi-protocol attacks.

In this paper, we revive multi-protocol attacks, but this time considering them in the context of guessing attacks on protocols with weak secrets. We name these newly found attacks as *multi-protocol guessing attacks*.

The result by Guttman et. al in [11] cannot always be applied to multi-protocol guessing attacks. This is because, unlike all other attacks including multi-protocol attacks, guessing attacks can be launched entirely off-line, with mere eavesdropping (without blocking or modifying the messages) and even without finishing any protocol run.

Most importantly as we will illustrate, they can be launched without replaying messages. Since Guttman et. al’s result is based on showing that all inbound linking paths (messages from other protocols into the primary protocol) can be removed, it doesn’t apply for attacks that can be launched using mixed protocols where there are no inbound linking paths.

We will illustrate how multi-protocol guessing attacks can be launched the using EKE protocol. Firstly, we make two assumptions about the guessing attacks we consider in this paper:

1. The passwords being guessed have low-entropy (typically, chosen by humans). Contrast this with high-entropy passwords such as machine-generated, where the passwords are chosen from a large space, making guessing infeasible.
2. The verification of a guess does not need repeated on-line interaction with other parties—typically, for repeated unsuccessful attempts, servers raise an alarm and mount additional countermeasures like shutting down the connection etc. In this paper, we consider only those guessing attacks that can be launched entirely off-line, where failed attempts are undetectable.

Now consider the following EKE (Encrypted Key Exchange) protocol presented by Mellovin and Berritt in [4]:

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

Here, a and b try to agree on a shared session key k , with $passwd(a,b)$ representing the password that a shares with b and pk_a , an asymmetric key of a . Lowe [15] analyzed this protocol using FDR [14] and found no attacks.

Now consider another protocol presented in [9]:

Msg 1. $a \rightarrow b : \{c, n\}_{k1}$
 Msg 2. $b \rightarrow a : \{f(n)\}_{passwd(a)}$

$k1$ is a ’s public key, and c is a “confounder” (a redundant random number) to prevent guessing. n is any number and f is a function which is publicly known. This protocol as well was not known to have any flaws when executed in isolation.

It is interesting to ask if c is really necessary. Without c , a penetrator can guess $passwd(a)$ and decrypt message 2. He can then do $f^{-1}(f(n))$ and encrypt the result with $k1$. A successful comparison with message 1 (without c , just $\{n\}_{k1}$) would verify the guess.

Gong et. al [?] suggest that using c is unnecessary if $k1$ is unknown— i.e. given that $k1$ is unknown to the penetrator (as assumed in the EKE protocol above), the protocol is secure.

However, if the protocol is combined with the EKE protocol, i.e. in a mixed environment, the following attack can be visualised:

Attack 1. Let $\mathbb{P}1$ represent the first protocol (EKE) and $\mathbb{P}2$, the second protocol. A penetrator can initially guess $passwd(a)$ in $\mathbb{P}1$ and decrypt message 1 to obtain pk_a . He can then guess $passwd(a)$ in $\mathbb{P}2$ and get $f(n)$. From this value he can obtain n and encrypt it with pk_a that he obtained from $\mathbb{P}1$. Finally he can match this value with its recorded value in message 1 of $\mathbb{P}2$ to verify his guess.

Now consider another identification protocol¹ similar to the one presented in [9]:

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

Let this protocol be represented as $\mathbb{P}3$. Here user a aims to identify itself to server s (pv_a is the private key of a). This protocol as well is secure, given that the corresponding public key of a (pk_a) is unknown to the penetrator,

However, when it is mixed with $\mathbb{P}1$ due to any of the reasons mentioned in section 1, the following attack can be visualised:

Attack 2. The penetrator can initially guess $passwd(a, b)$ and decrypt message 1 in $\mathbb{P}1$ to obtain pk_a . He can then guess $passwd(a)$ and decrypt message 3 in $\mathbb{P}3$ to obtain $\{a, n_s\}_{pv_a}$. Finally he can decrypt $\{a, n_s\}_{pv_a}$ with pk_a (from $\mathbb{P}1$) and match n_s in it with n_s sent in message 2 to verify his guess.

Similar attacks can be found on many existing protocols. Observe that these attacks were possible even though all the protocols, $\mathbb{P}1$, $\mathbb{P}2$, $\mathbb{P}3$ were otherwise secure in an isolated environment. Also, none of the attacks assumed that the user used the same passwords in each protocol—Thus, in general, the effect when the same passwords are chosen in different protocols cannot be overstated. Ofcourse, both assumed that users use the same public key in more than one application, which is not unreasonable to assume, given the facts in section 1.

3. Strand Space Frame Work

In this section, we will give an introduction to strand space framework of [6, 7, 8]. We chose strand spaces since it is a particularly suitable framework to derive and demonstrate the results required in these contexts.

To start with, let **Fact** denote the set of all possible elements in a protocol² and **Atom**, the set of atomic values (eg. *Alice*, *Bob*, N_A , $\text{PubKey}(A)$ etc.) assumed

¹Typically these are protocols used by ATMs (automatic teller machines).

²with ‘message’ referring to the entire collection of facts sent in a protocol step.

to contain in a protocol. When two data items a and 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 $\{a\}_k$ and the inverse of a k as, k^{-1} .

When we talk about the first or second component in a fact with two components we will use subscripts “1” and “2” as: $(f1, f2)_1 \hat{=} f1$, $(f1, f2)_2 \hat{=} f2$

Also, *subfact* relation \sqsubset is defined as the smallest relation on facts such that,
 $f \sqsubset f$; $f \sqsubset \{f'\}_{k'}$ iff $f \sqsubset f'$; and
 $f \sqsubset (f1, f2)$ 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 $\langle \pm f1, \pm f2, \dots, \pm fn \rangle$. 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 a 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 $n2 \in \mathcal{N}_C$ receives a fact, then there exists a unique $n1$ with $n1 \rightarrow n2$.
2. If $n2 \in \mathcal{N}_C$ 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 F , if there is no node previous to n transmitting a fact in 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, M_P and keys K_P .

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

- M** Text message $\langle +f \rangle$ with $f \in M_P$.
- F** flushing $\langle -f \rangle$.
- T** Tee $\langle -f, +f, +f \rangle$.
- C** Concatenation $\langle -f1, -f2, +f1f2 \rangle$.
- S** Separation $\langle -f1f2, +f1, +f2 \rangle$.
- K** Key $\langle +k \rangle$ with $k \in K_P$.
- E** Encryption $\langle -k, -f, +\{f\}_k \rangle, k \in K_P$.
- D** Decryption $\langle -k^{-1}, -\{f\}_k, +f \rangle, k \in K_P$.

A *regular strand* in contrast, is one corresponding to an honest agent. An *ideal* captures all possible operations of an honest agent on a given set using a given set of keys. In particular, a smallest k -ideal for an element $h \in \mathbf{Fact}$ using set of keys k is denoted as $I_k[h]$. It consists of all possible operations (concatenation and encryption) between h and all the elements of \mathbf{Fact} .

Since we deal with mixed protocols, we use the concept of mixed strand spaces as well.

Definition 3. A mixed strand space *consists of combined strands from different protocols. Some particular strands in this space are called primary strands, implying the primary protocol under consideration and secondary strands, consisting of all the remaining protocols. Set of facts $I \in \mathbf{Fact}$ is unserved in a strand space Σ if an entry point for I does not lie on a secondary strand. Similarly I is strongly unserved in Σ if no element in I ever originates on a secondary strand.*

We will now present our method to analyze protocols for guessing attacks using this framework and illustrate it on an example. The basic idea for the method is derived from [10] where Guttman uses the concept of identifying the unintended services by honest agents to find attack scenarios. The method we give below is such a practical recipe for an informal analysis.

Firstly, we denote \doteq as a binary relation on facts that returns **true** if there is a match between two facts and a **false** otherwise. Also, we define some more sets of facts, classified into:

1. Facts that can be guessed (as set G);
2. Facts that can be derived by the penetrator P in all possible roles and combinations (as set D);
3. Facts that can be verified (as set V).
eg. $\{n_a\}_{passwd(a)}$ can be compared with $\{n_a, n_b\}_{passwd(a,b)}$ by guessing $passwd(a, b)$ and³ doing $n_a \doteq (n_a, n_b)_1$;
4. Facts that can be constructed (as set C)—These are typically those that the penetrator can construct using his knowledge and derived knowledge (D and facts obtained by guessing, O).

Firstly, we identify all unintended services offered by honest agents that increase the penetrator knowledge and capability. Then, we will list out the initial penetrator knowledge in terms of facts that he knows and use the unintended services to derive all possible facts that he can derive—For this, we consider all possible roles that a penetrator can play. Then, we will use set G to enumerate the facts that he can obtain corresponding to decrypting facts using guesses in place of G (set O). Finally, we will list out all possible verification attempts using V to verify a guess and all possible facts that can

³Recall that subscripts “1” and “2” return the first and second elements respectively from a fact with two elements.

be constructed using the updated penetrator knowledge, similar to the form of any recorded messages—which again would verify a guess.

Now consider the following “demonstration protocol” presented by Gong et. al [9]:

Msg 1. $a \rightarrow b : \{a, b, na1, na2, ca, \{ta\}_{K_a}\}_{K_s}$
 Msg 2. $s \rightarrow b : a, b$
 Msg 3. $b \rightarrow s : \{a, b, nb1, nb2, cb, \{tb\}_{K_b}\}_{K_s}$
 Msg 4. $s \rightarrow a : \{na1, na2 \oplus k\}_{K_a}$
 Msg 5. $s \rightarrow b : \{nb1, nb2 \oplus k\}_{K_b}$
 Msg 6. $a \rightarrow b : \{ra\}_k$
 Msg 7. $b \rightarrow a : \{f1(ra), rb\}_k$
 Msg 8. $a \rightarrow b : \{f2(rb)\}_k$

Lowe [15] analyzed this protocol and found no attacks. To illustrate our method, let us assume that the server cannot detect replays.

Let *init*, *resp* and *serv* denote the regular strands of a , b and s in the protocol. We will remove lot of redundant steps in the illustration because of the obvious symmetry visible in the protocol steps— a and b have the same message formats in four messages.

Step 1. We first identify the unintended services provided by *serv*: From the protocol it is evident that messages 1 and 3 are “junk”. i.e. any one can replay them spoofing as a or b (provided that the server cannot detect those replays which was part of our initial assumptions). Hence, strand *serv* gives a message $\{na1, na2 \oplus k\}_{passwd(a)}$ each time with a different key for each of such replays.

Step 2. The facts that can be guessed (presumably, poorly chosen secrets) in this protocol are: $passwd(a)$ and $passwd(b)$.

Step 3. The facts initially known to P ($M_P \cup K_P$) are: $a, b, s, K_s, \{a, b, na1, na2, ca, \{ta\}_{K_a}\}_{K_s}, \{b, a, nb1, nb2, cb, \{tb\}_{K_b}\}_{K_s}, \{na1, na2 \oplus k\}_{K_a}, \{nb1, nb2 \oplus k\}_{K_b}, \{ra\}_k, \{f1(ra), rb\}_k, \text{ and } \{f2(rb)\}_k$.

Step 4. We will now enumerate all derivable facts (set D) by considering all possible interactions of P : $(a, b), (Pa, b), (a, P), (Pa, P)$. (Here Pa represents P masquerading as a). We do not need to consider the remaining possibilities because of the symmetry in the protocol. From combinations (a, b) and (Pa, P) and since s_{serv} gives two instances of $\{na1, na2 \oplus k\}_{K_a}$ (from step 1 above on unintended services), P can now have, $\{na1, na2 \oplus k\}_{K_a}$ and $\{na1, na2 \oplus k'\}_{K_a}$ (with $k \neq k'$).

We leave it to the reader to check the remaining combinations to make sure that no useful terms can be derived.

Step 5. We now derive all possible facts obtainable by using elements in set G : These are, $(na1, na2 \oplus k), (nb1, nb2 \oplus k)$. Also, the penetrator can

obtain $na2$, since in combination (a, P) of P knows k and hence obtains na from $na2 \oplus k$.

Step 6. We now consider the set V (verifiable). From step 4 using set D , P can compare $na1$ $\{na1, na2 \oplus k\}_{passwd(a)}$ and $\{na1, na2 \oplus k'\}_{passwd(a)}$ guessing $passwd(a)$ and decrypting both. i.e. the guess, he can do, $(\{\{na1, na2 \oplus k\}_{passwd(a)}\} \{ \{na1, na2 \oplus k'\}_{passwd(a)}\}^{-1})_1$

Step 7. We will now show how we uncover an attack on this protocol and the importance of the ca and cb . We will try to find the facts that P can construct using his knowledge that he obtained in previous steps and in the format of some recorded messages. Firstly, remove ca from message 1. The remaining part of this message contains, $a, b, na1, na2$, which are known to the penetrator ($na1$ and $na2$ from step 5). Time stamp ts is arbitrary and hence he can encrypt it with his guess which he previously used to decrypt message 4 in step 5. He can now combine all these fields, encrypt with Ks (which is a public key and hence $\in K_P$) to construct a message of the form of message 1 and compare it with the actual recorded value.

It is here that the importance of ca would be perceivable. Observe that P could not derive it using any of the previous steps. Hence, if ca is present inside message 1, it thwarts a guessing attack by preventing P from constructing a similar message to verify the guess. Similarly, the case for cb .

4. Mixing EKE

In this section we will use the procedure we developed in the previous section on the EKE protocol, which we showed to be vulnerable to multi-protocol guessing attacks in section 2. The analysis would help in deriving rules about some ‘‘critical’’ messages which would be then framed within the strand space model to prove the correctness of EKE protocol in a mixed environment, when the rules are followed.

Figure 1 represents the EKE protocol.

Step 1. The unintended services here are only that of agent b 's. — corresponding to $\{pk\}_{passwd(a,b)}$, b gives, $\{\{k\}_{pk}\}_{passwd(a,b)}$.

Step 2. $G = \{passwd(a, b)\}$.

Step 3. $M_P = \{\{pk_a\}_{passwd(a,b)}, \{\{k\}_{pk}\}_{passwd(a,b)}, \{na\}_k, \{na, nb\}_k, \{nb\}_k\}$.

Step 4. Possible interactions: (a, b) , (a, P) , (a, Pb) , (P, b) , (Pa, b) . From (a, P) , P can obtain PK'_a . Hence $D = \{PK'_a\}$. (Again we leave it to the reader to check all the remaining combinations to verify that there are

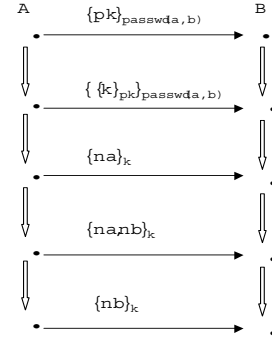


Figure 1: Message Exchange in EKE protocol

no other possible derivations).

Step 5. Facts obtained by guessing: $O = \{PK_a, \{k\}_{PK_a}\}$.

Step 6. Verifiable facts (V): $V = \emptyset$; However, from steps 4 and 5, P can do (with guess g), $pk'_a \doteq \{\{pk_a\}_{passwd(a,b)}\}g^{-1}$.

Step 7. Constructible facts: $C = \emptyset$. Since none of the terms in $M_P \cup D \cup O$ can be used to construct a recorded value for verification.

From the results in the procedure, and in particular, steps 6 and 7, it is evident that, the protocol remains secure as long as,

1. pk_a is never obtainable from a secondary strand (i.e. using any secondary strand s , $pk_a \not\sqsubseteq f, \forall f \in D, D$ obtained from s).
2. No term encrypted with $pv_a (= pk_a^{-1})$ should be obtainable as well.
3. No term encrypted with PK_a should be verifiable ($\notin D \cup M_P$).

We will now represent these conditions in a mixed strand space reflecting EKE protocol as a ‘‘primary’’ protocol and all others as secondary protocols.

Definition 4. Let Σ be a strand space.

1. $Init[pk_a, passwd(a, b), k, na, nb]$ is the set of strands in Σ whose trace is $\langle +\{pk_a\}_{passwd(a,b)}, -\{\{k\}_{pk}\}_{passwd(a,b)}, +\{na\}_k, -\{na, nb\}_k, +\{nb\}_k \rangle$
 Σ_{init} is the union of the range of $Init$.
2. $Resp[pk, passwd(a, b), k, na, nb]$ is the set of strands in Σ whose trace is $\langle -\{pk_a\}_{passwd(a,b)}, +\{\{k\}_{pk}\}_{passwd(a,b)}, -\{na\}_k, +\{na, nb\}_k, -\{nb\}_k \rangle$
 Σ_{resp} is the union of the range of $Resp$.

Also, $\Sigma_{init}, \Sigma_{resp}$ are pairwise disjoint and form the primary strands in Σ , denoted as $\Sigma_P (= \Sigma_{init} \cup \Sigma_{resp})$. Rest of the strands in Σ are secondary strands and are represented as $\Sigma \setminus \Sigma_P$ (\setminus is the set difference operator).

The following definition defines the required sets of items that we would need to define the rules that we derived above.

Definition 5. Let L_0 and I_D be defined such that,

- L_0 denotes the set of all terms such that $\forall pk \in PK, \exists f \in M_P \cup D \cup O. pk \sqsubset f$.
- $I_D = I_k[D]$ with $k = PV_a \cup PK_a$

We will now model the conditions we derived using the procedure above, within the strand space framework. Our main theorem states that a mixed protocol environment containing the EKE protocol as the primary protocol is secure against multi-protocol guessing attacks as long as the strand space respects those conditions:

Theorem 1. Let Σ represent a mixed strand space with the EKE protocol representing the primary strands. Let C be a bundle in Σ . Then, no guessing attacks can succeed in C if:

1. PK_a is unserved in Σ .
2. I_D is strongly unserved in Σ .

Proof. We will do a case analysis. Firstly, observe from our procedure that, a guessing attack is feasible if, either verification using set V (step 6) or set C (step 7) is successful. Hence, we will consider those sets and use the above conditions to prove that the sets will always be null under those conditions.

Part 1. PK_a is unserved. From step 5, verification of any $g \in G$ is possible, if $(D \cup M_P \cup K_P) \cap O \neq \emptyset$. In this case, from the our analysis, $(D \cup M_P \cup K_P) \cap O = \{pk_a\}$ iff $pk_a = pk'_a$. However, according to condition 1 in the theorem, PK_a is unserved in Σ . i.e., If pk_a originates in $(\Sigma_{init} \cup \Sigma_{resp})$ then $\forall pk'_a$ originating in $\Sigma \setminus \Sigma_P$, $pk_a \neq pk'_a$. Hence, $(D \cup M_P \cup K_P) \cap O \neq \emptyset$.

Part 2. Again observe that, for verification, pk_a should be verifiable since $O = \{pk_a, \{k\}_{pk_a}\}$ and $k \notin (D \cup M_P \cup K_P)$. The procedure demonstrated that, pk_a is not verifiable in Σ_P . And part 1, demonstrated that pk_a is not verifiable in Σ if, pk_a is unserved in Σ .

A verification is still possible, if f encrypted with pk_a or pk_a^{-1} is verifiable and $\in \Sigma \setminus \Sigma_P$. However, in contradiction, condition 2 requires that, I_D , (all verifiable terms encrypted with pk_a or pv_a from definition 5), is strongly unserved in Σ . Or no such term ever originates on a secondary strand as well. Hence, a verification attempt is infeasible as long as the condition holds. \square

5. Conclusion

In this paper we have introduced new types of attacks called multi-protocol guessing attacks, on protocols using weak secrets. We have developed a systematic procedure to derive some conditions that would prevent these attacks. We then proved that—as long as these conditions are satisfied—a protocol can never be attacked through multi-protocol guessing attacks.

Some points are worth mentioning here. Our procedure to analyze protocols is stronger than Gong et. al's [9] since we consider all obtainable knowledge sets by the penetrator, as Lowe [15] does, whereas Gong et. al consider only fixed penetrator knowledge. The uniqueness in the procedure is that, it allows to pinpoint the exact kind of vulnerabilities that a protocol might present in terms of especially when operating in a mixed environment. By determining the conditions that prohibit any new vulnerabilities that can possibly arise, a more concrete designing of protocols is possible.

One immediate extension of this work is related to automating the procedure we have used to detect possible guessing attacks. Another possible extension would be to find out techniques of general applicability to prevent multi-protocol guessing attacks, as in [11]. It wou

References

- [1] J. Alves-Foss. Multi-Protocol Attacks and the Public Key Infrastructure. In *Proc. National Information System Security Conference*, pages 566–576, October 1998.
- [2] J. Alves-Foss. Provably Insecure Mutual Authentication Protocols: The Two Party Symmetric Encryption Case. In *Proc. 22nd National Information Systems Security Conference.*, pages 44–55, Arlington, Va, October 1999.
- [3] J. Alves-Foss. Cryptographic Protocol Engineering: Building Security from the Ground Up. In *Proceedings of International Conference on Internet Computing*, June 2000.
- [4] S.M. Bellare and M. Merritt. Encrypted key exchange: 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] T. Dierks and C. Allen. The TLS protocol. Technical report, January 1999.
- [6] F. J. THAYER Fábrega, J. C. Herzog, and J. D. Guttman. Why is a security protocol correct? *IEEE Computer Symposium on Security and Privacy*, 1998.

- [7] 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.
- [8] F. Javier THAYER Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Mixed Strand Spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, volume 27(2), pages 10–14. IEEE Computer Society Press, June 1999.
- [9] 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.
- [10] Joshua D. Guttman. Strand Spaces and Protocol Security Goals.
- [11] Joshua D. Guttman and F. Javier THAYER. Protocol Independence through Disjoint Encryption. *13th IEEE Computer Security Foundations Workshop*, pages 24–34, July 2000.
- [12] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Technical report, November 1998.
- [13] J. Kohl and C. Neuman. The Kerberos network authentication service (v5). RFC 1510, September 1993.
- [14] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, volume 1055, pages 147–166. Springer-Verlag, 1996. Also in *Software Concepts and Tools*, 17:93–102, 1996.
- [15] Gavin Lowe. Analyzing protocols subject to guessing attacks. *Workshop on Issues in the Theory of Security (WITS'02)*, January 2002.
- [16] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). Technical report, November 1998.
- [17] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.
- [18] B. Clifford Neuman and Stuart G. Stubblebine. A note on the use of timestamps as nonces. Technical report, April 1993.