A Non-Interactive Dual Channel Authentication Protocol for Assuring Pseudo-confidentiality

David Irakiza, Md E. Karim, Vir V. Phoha
Center for Secure CyberSpace
Louisiana Tech University
{dir003,mdekarim,phoha}@latech.edu

Abstract

We introduce a non-interactive dual channel authentication protocol and apply it to long distance communication for assuring pseudo-confidentiality, a criteria that prevents a malicious agent from exfiltrating information to unauthorized destinations. Unlike previously proposed protocols that assume a manual (human-aided) or equivalent authenticated channel, our protocol utilizes a non-manual authenticated channel. We analyze the security properties for a possible realization of this protocol and develop a prototype. Through a Raspberry-Pi implementation, we show how the incorporation of the proposed scheme into the future design of keyboard interfaces may impact authentication practices.

1. Non-interactive dual channel authentication protocols

Non-interactive dual channel authentication protocols employ two channels and authenticate information r_1 received through one presumably insecure channel using a piece of brief information r_2 (computed from r_1) received through the other presumably authenticated channel. To remain lightweight these protocols employ one way communication from a message sender, Alice, to a message recipient, Bob. It can be shown that in these protocols, imposters cannot authenticate themselves if, (i) only one of the channels is compromised or (ii) both channels are compromised but attacks on them are not coordinated.

Existing literature describes a family of non-interactive message authentication protocols (e.g., [1, 2, 4, 3, 5]), known as NIMAPs, that use a manual (human-aided) authenticated channel. In these protocols, a hash value for each message being sent is generated. Alice then transmits the message and hash over the insecure and authenticated channels respectively to Bob. In some protocols, a key is applied to the hash function when generating the hash value and this key is sent with the message over the insecure channel [3, 5] or with the hash value over the authenticated channel [2].

NIMAPs assume that the authenticated channel is human-aided hence adversarial influence on this channel is significantly limited. This assumption is sufficient for short distance communications requiring infrequent authentication of messages. However, this limits their application in long distance communications (e.g, over the internet) where frequent authentication is required and a human-aided authenticated channel is unrealistic and/or infeasible.

2. Proposed protocol

Replacement of the human-aided authenticated channel in NIMAPs erodes the security benefit assured by such a channel. This exposes them (NIMAPs) to channel spoof attacks where an adversary, Eve, could spoof the identity of the authenticated channel when sending her authentication messages to Bob. Because of the non-interactive nature of

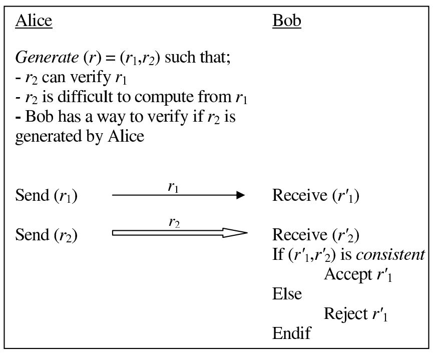

Alice
Bob
Generate $(r) = (r_1,r_2)$ such that;
- r_2 can verify r_1
- r_2 is difficult to compute from r_1
- Bob has a way to verify if r_2 is generated by Alice
Send (r_1)
r_1
Receive (r'_1)
Send (r_2)
r_2
Receive (r'_2)
If (r'_1,r'_2) is *consistent*
Accept r'_1
Else
Reject r'_1
Endif

Figure 1. Proposed protocol (Generic)

This work is supported by AFOSR Grants FA9550-09-1-0479 and FA 9550-09-1-0165.

the protocol, Bob has no way to know whether these authentications are sent by Alice hence, he will accept a message if its authentication is valid (i.e., it authenticates the message sent through the insecure channel).

We address this issue by introducing the following: (i) we assign the task of Alice to a small hardware attachment that we assume not to be affected by a channel compromise, and (ii) we assume that it is difficult for the adversary, Eve to compute r_2 given r_1.

Figure 1 presents the proposed generic protocol. In this protocol, Alice is a hardware attachment to the keyboard with the following properties: (i) She has processing capabilities to parse typed keystrokes and apply a method *generate* to identify r_1 and to compute r_2 from r_1; (ii) She only receives input from the keyboard; (iii) She sends output to Bob using two different channels (one is insecure that runs through the host and the other is authenticated that bypasses the host). Bob is a verifier program located in a remote computer.

In Figure 1 (and in subsequent protocol figures), the "$'$" added to information received by Bob indicates that the sent and received values might be different and the insecure and authenticated channels are represented by "$\rightarrow$" and "$\Rightarrow$" respectively.

The protocol is designed based on the following assumptions: (i) an r_2 can verify the associated r_1; (ii) an adversary can generate an r_1 or snoop the r_1 generated by Alice but it is difficult for her to compute the associated r_2 from a given r_1; (iii) Bob can compute r_2 from r_1; (iv) r'_1 is accepted only if (r'_1, r'_2) is *consistent* i.e., r_{2_b} (the expected r'_2 computed by Bob for the r'_1 he received) is the same as r'_2.

Alice *generates* r_1 and r_2 from the keystrokes typed by a user and sends r_1 through the possibly compromised host using the insecure channel and r_2 directly to Bob using the authenticated channel. When Bob receives r_1 and r_2 from Alice, he accepts r'_1 only if (r'_1, r'_2) is *consistent*. To defeat this protocol, Eve either; (i) waits for Alice to transmit an r_1 and replaces r_1 with $r_{1_{eve}}$ and expects that it will be verified by the associated r_2 (generated and transmitted by Alice), or; (ii) estimates an $r_{2_{eve}}$ for her $r_{1_{eve}}$ and transmits both of them to Bob pretending that the $r_{2_{eve}}$ is transmitted over the authenticated channel (by spoofing the authenticated channel's identity).

Existing NIMAPs do not make assumption (ii), perhaps because of the challenge involved with its realization. Instead, they assume that Eve cannot successfully use (or masquerade to be using) the manual authenticated channel. These protocols can be defeated if the manual channel is replaced by a non-manual physical channel since Eve then can compute the right $r_{2_{eve}}$ for her $r_{1_{eve}}$ and masquerade as Alice.

3. A hash based realization for assuring pseudo-confidentiality

The exfiltration of information by a malicious agent from a compromised host to an external adversary can be restricted by blocking all access from the host to unauthorized destinations. We refer to an authorized destination as a domain name/IP that: (i) has been typed at least once by a human user (establishing consent/approval), or (ii) has been returned in a response packet from an authorized destination. We define ***pseudo-confidentiality*** as the security property satisfied when an adversary cannot exfiltrate information to unauthorized destinations.

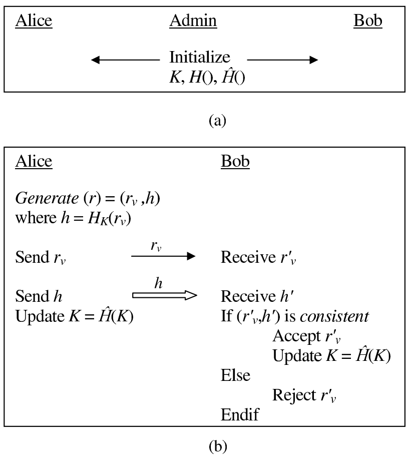


Figure 2. Hash based realization

A hash based realization of the proposed protocol assuring *pseudo-confidentiality* is shown in Figure 2. In this realization, the *generate* function used by Alice produces components r_v and h. r_v is the destination (IP/domain name) of a request being made and $h = H_K(r_v)$ is the hashed value of the requested destination. This realization works in two phases as described below:

(i) Initialization phase
This is a one step setup process required prior to the establishment of the rest of the protocol. During this phase, a human operator (i) informs Bob of the functions H and $\hat{H}$ used by Alice and (ii) initializes a K for Alice and Bob. During the protocol phase, function H uses K as a parameter to produce h and function $\hat{H}$ is used to update K by both Alice and Bob. Due to some failure or interruption in communication, if Alice and Bob do not have the same updated K, all destination requests will be rejected and the value of K will have to be re-initialized.

(ii) Protocol phase
In this phase, components r_v and h are transmitted from Alice to Bob via the insecure and authenticated channels

respectively. When Bob receives both r'_v and h', he computes his own h using his K i.e., $h = H_K(r'_v)$ and considers (r'_v, h') to be *consistent* if the received $h' = h$. The value of K is updated by both Alice and Bob using another function $\hat{H}$. Alice updates her K after sending h to Bob and Bob updates his K every time he receives an $h' \neq \emptyset$ from Alice and accepts r'_v. $\hat{H}(K)$ transforms the value of K based on the current value of K to produce a new K.

During experimentation, we relax the definition of *consistent* by accepting requests to destinations that are (i) non-typed but had been authorized before and (ii) received in response to requests to authorized destinations, to improve the usability of the solution.

Security Analysis

Bob accepts r'_v if (r'_v, h) is *consistent* for the current K (that Bob has knowledge of). Bob can be made to accept a request from Eve in the following cases:

Case 1: Channel spoof attack

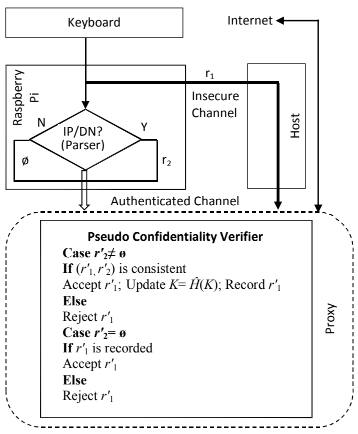
A malicious agent, Eve, residing in the host, can no longer launch a channel spoof attack simply by sending her own r_v and h to Bob if the K she uses for computing h (at a certain instance) is not what Bob is expecting Alice to use. Since our assumption is that Eve has no knowledge of K, Alice can only expect that the K she is using matches with the K Bob is expecting. For an n-bit K, the probability of such a match is 2^{-n}.

Case 2: Second preimage Attack

Eve can replace the destination r_v generated by Alice with her own destination r_{eve} and expect that there exists a K'' such that $H(r_v, K) = H(r_{eve}, K'')$. A second preimage resistant hash function is relevant in our case because for a second preimage resistant hash function, given an input r_v, it should be difficult for an adversary to find another input $r_{eve} \neq r_v$ such that $H(r_{eve}) = H(r_v)$. If H is ϵ_s second preimage resistant (i.e., ϵ_s is the probability of $H(r_{eve}) = H(r_v)$ occurring), then the probability of a successful attack is ϵ_s.

4. Results

We setup a prototype as shown in Figure 3 using Raspberry-Pis (model B) as the hardware attachments to the keyboard and collected data for over four months. As mentioned in Section 3, our experimentation considers a relaxed definition of *consistent* as reflected in the logic for the *Pseudo Confidentiality Verifier* in Figure 3. Figure 4 shows typical results for a representative seven day period. As expected, all illegitimate requests were denied (requests not initiated by the user). Some legitimate requests were also denied because associated destinations were not typed once before their use. Figure 4 (b) shows the breakdown of the legitimate typed and un-typed requests.



```
Pseudo Confidentiality Verifier
Case $r'_2 \neq ø$
If $(r'_1, r'_2)$ is consistent
Accept $r'_1$; Update $K = \hat{H}(K)$; Record $r'_1$
Else
Reject $r'_1$
Case $r'_2 = ø$
If $r'_1$ is recorded
Accept $r'_1$
Else
Reject $r'_1$
```

Figure 3. Raspberry-Pi based prototype

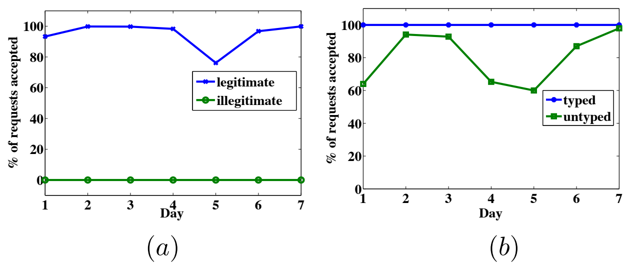


Figure 4. % of requests accepted per day

References

[1] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*, 2002.

[2] C. Gehrmann, C. J. Mitchell, and K. Nyberg. Manual authentication for wireless devices. *RSA Cryptobytes*, 7(1):29–37, Spring 2004.

[3] A. Mashatan and D. R. Stinson. Noninteractive two-channel message authentication based on hybrid-collision resistant hash functions. *IACR Cryptology ePrint Archive*, 2006:302, 2006.

[4] S. Pasini and S. Vaudenay. An optimal non-interactive message authentication protocol. In *Proceedings of the 2006 The Cryptographers' Track at the RSA conference on Topics in Cryptology*, CT-RSA'06, pages 280–294, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] M. R. Reyhanitabar, S. Wang, and R. Safavi-Naini. Non-interactive manual channel message authentication based on etcr hash functions. In *Proceedings of the 12th Australasian conference on Information security and privacy*, ACISP'07, pages 385–399, Berlin, Heidelberg, 2007. Springer-Verlag.