

CS6490 Project Report: Multiple Service Authentication Protocol

Jeff Brock, Roozbeh Gholizadeh, Montgomery Carter

1 Introduction

We rely on large tech organizations more and more to facilitate our daily communication. As a result, these large tech organizations have access to our personal information, from the mundane to the deeply personal and revealing.

While operating off the grid may work for some individuals, the convenience of such services provide a compelling argument in favor of their use. The typical attitude is to accept the loss of privacy in exchange for use of the service. Rather than forgoing the convenience of utilizing these services, or acquiescing our privacy, it is desirable to have a solution that protects our privacy as we use these services.

In this paper, we focus on instant messaging (IM) communication because it is a real time communication that allows exchange of messages to establish privacy for the duration of a conversation. (Our proposed solution can easily be extended to email services, though it seems more natural in the context of real time communication.) There are many solutions to provide a layer of authentication and encryption that run on top of such services as Google Hangouts, Facebook Chat, Skype, etc. However, these solutions are complicated, and generally rely on an external authentication service, or public-key (RSA, DSA, PGP) based authentication to establish shared session keys for encrypting and integrity protecting the ensuing communication. Configuring such a solution is complex enough to dissuade the typical user from employing them, and can have cost associated that provides further barrier to use.

It would be desirable to have a method for encryption and integrity protection that does not require external authentication (which means an additional service to trust), and does not use complicated public key signature schemes for integrity protection.

We present the Multiple Service Authentication Protocol (MSAP), a protocol for establishing shared session keys to be used for encryption and integrity protection of instant messaging (IM) communications. However, unlike existing solutions, MSAP does not, itself, rely on public key authentication. Instead, MSAP relies on the authentication performed by service providers such as Google, Microsoft, or Facebook to verify the identities of IM conversation participants. IM service providers have an incentive to deliver messages to the intended recipient, as well as to ensure that only the intended recipient has access to delivered messages. Without this, their service is of little use and they would have no user base. To ensure these criteria, an IM service must authenticate users. (However, there is no incentive for IM service providers not to read these messages for their own internal purposes, such as generating behavioral models of users, and selling ads.)

Further, the protocol ensures that service providers do not have access to the shared session keys, and importantly, prevents service providers from employing a man in the middle (MitM) attack and acting as an interfering intermediary between the two communicating parties.

Unsurprisingly, MSAP utilizes a Diffie-Hellman (DH) key exchange for the establishment of shared session keys. The more interesting feature of MSAP is its use of multiple, separate messenger services to perform the requisite DH

exchange. By sending half of the DH exchange via one IM service, and the response via a separate IM service, no one service is privy to both sides of the exchange, rendering both services incapable of mounting a MitM attack.

Although the main goal of protecting against MitM attacks from service providers is achieved simply by moving the DH response to a separate service, there is still a threat of MitM attacks from an external intruder who may have control of some node on the route between communicating parties, and is able to modify messages between one participant and multiple chat services. To address this threat, MSAP includes two mechanisms to protect against MitM attacks performed by external parties (e.g., one chat participant's ISP network). The first (and preferable) mechanism works by requiring SSL encryption between chat client and chat host (which is available with many IM providers such as Facebook and Google) for at least one of the DH exchange messages. By using SSL in this manner, such an intruder will not have all of the information necessary to mount a MitM attack.

If no chat service offering SSL encryption is available, the second mechanism comes in to play, where each participant remembers a temporary "long term" secret (TLTS). This temporary "long term" secret is used in lieu of a DH exchange to encrypt the exchange of the session keys for every conversation subsequent to the initial conversation between the two parties. During each subsequent conversation between the parties, this TLTS is updated with a new TLTS. Of course, this assumes that no intruder performs an MitM attack during the initial conversation.

We first review existing solutions and related work, followed by a definition of our adversary model. Next we define the MSAP protocol, and follow this with a discussion of the various features and insights of the protocol. This is followed by a description of a prototype implementation of the protocol.

2 Related Work

<DH>
<OTR>

3 Adversary Model

In our adversary model we assume two nefarious players. The relationship between these players and the chat participants are only described for Alice, however, apply equally well to Bob.

The first is Hank the host, the server hosting a chat service. We assume Hank securely and accurately authenticates users, however, is motivated to read communications in order to build behavioral models and sell ads (or perhaps more sinister reasons). Hank is capable of mounting a MitM attack on a DH exchange conducted exclusively over a single chat service, because he necessarily must receive and relay all messages between Alice and Bob. Even when encryption is used between client and host, Hank still has access to read and modify (or replace) the plaintext messages between participants. We refer to this type of MitM attack as an internal MitM attack. Hank, however, doesn't have control of any

node that sits on the route between Alice and multiple chat services (including Hank). It is conceivable that Hank may have control of a few nodes in his periphery network, but not some node that allows Hank to modify messages between Alice and some other chat service. Further, we assume that there is no collusion between IM services where Hank may begin an MitM attack, and inform the second service that a DH exchange is occurring, and what the first half of the MitM attack keys are.

Trudy has control of some node along the route that is between Alice and every chat service (perhaps Trudy is Alice's ISP), and is capable of performing a MitM attack. We refer to this more traditional type of MitM attack as an external MitM attack.

We assume that both Hank and Trudy know MSAP, as well as the encryption and message digest algorithms used by MSAP.

In short, we consider the chat service providers to be capable of modifying messages only as they relay them, and external intruders capable of modifying any message sent/received to one of the chat participants. With that said, the consideration of external intruders was more of an afterthought — the main objective of MSAP is to prevent the service providers from reading conversations.

4 Protocol Definition

In this section we first introduce some terminology and book-keeping. Let S be a list of chat services. Alice maintains a list of usernames for Bob, U_B , where each U_{B_x} is Bob's username for S_x . Bob maintains a similar list, U_A , for Alice. Alice and Bob also keep track of a temporary "long term" secret, $TLTS_{AB}$. Encryption and integrity protection algorithms between Alice and Bob can be any accepted, secure algorithm. Let $f(a, b)$ be a secure function for generating a session key. Let $g(a, b)$ be a second secure function for generating a key, where $f(a, b) \neq g(a, b)$, and the output of $f(a, b)$ cannot be predicted from $g(a, b)$, and vice versa.

MSAP is described below, with Alice initiating communication with Bob

1. Alice and Bob both sign in to S_0 , with encryption required between client and host.
2. If $TLTS_{AB}$ is defined, Alice chooses a nonce, N_0 and a next service, S_n where $S_n \neq S_0$, and sends $\{U_{B_0}, TLTS_{AB}\{N_0\}\}_{SSL S_0 A}$ to S_0 , which relays $\{TLTS_{AB}\{N_0\}, S_n\}_{SSL S_0 B}$ to Bob. If $TLTS_{AB}$ is undefined, proceed to step 5.
3. Bob sends $\{U_{A_0}, TLTS_{AB}\{N_0 - 1\}\}_{SSL S_0 B}$ to S_0 , which relays $\{TLTS_{AB}\{N_0 - 1\}\}_{SSL S_0 A}$ to Alice.
4. If Bob's response decrypts to $N_0 - 1$, both parties set $DH_{AB} := TLTS_{AB}$, and proceed to step 11. Otherwise, continue with step 5.
5. Alice chooses a DH base g , modulus p , and secret exponent a , as well as a next service, S_n where $S_n \neq S_0$.
6. Alice sends $\{U_{B_0}, p, g^a \bmod p, S_n\}_{SSL S_0 A}$ to S_0 , which relays $\{g, p, g^a \bmod p, S_n\}_{SSL S_0 B}$ to Bob.
7. Alice and Bob both sign in to S_n . (Encryption unimportant)

8. Bob selects a DH secret exponent b .
9. Bob sends $\{U_{A_n}, g^b \bmod p\}_{SSL S_0 B}$ to S_0 , which relays $\{g^b \bmod p\}_{SSL S_0 A}$ to Alice.
10. Alice and Bob both calculate $DH_{AB} := g^{ab} \bmod p$.
11. Alice and Bob choose nonces, N_A and N_B .
12. Alice sends $DH_{AB}\{N_A\}$ to Bob, and Bob sends $DH_{AB}\{N_B\}$ to Alice, both via S_n .
13. Alice and Bob both calculate the session key, $K_{AB} = f(N_A, N_B)$.
14. Both Alice and Bob calculate and update $TLTS_{AB} = g(N_A, N_B)$.

Alice and Bob now have a session key with which they can encrypt and perform integrity protection for their ensuing conversation.

MSAP is summarized in Figure 1.

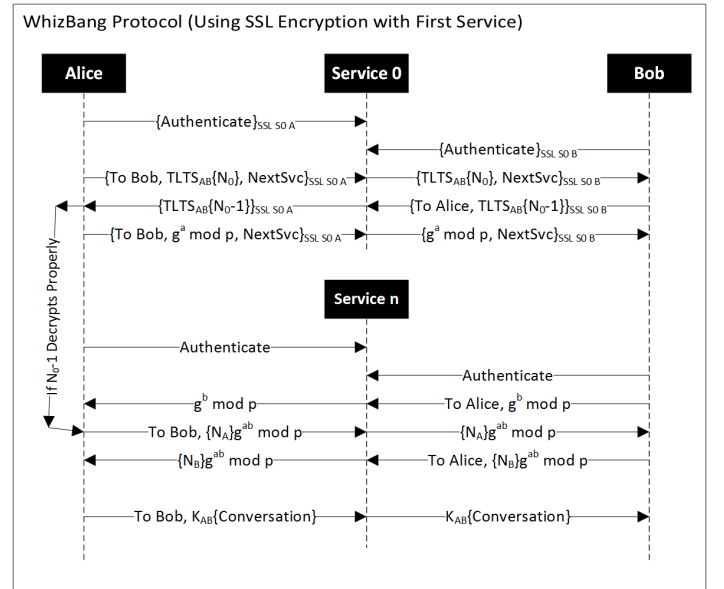


Figure 1: My caption

5 Discussion

Because we assume IM services faithfully perform authentication and message delivery, we know a message destined for Bob via S_0 will reach Bob. However, if S_0 notices a DH exchange occurring, it may opt to begin an MitM attack, and deliver a separate DH exchange for which it knows the DH exponent. To protect against this internal MitM threat, it is sufficient to simply complete the DH exchange on a separate service, as illustrated in Figure 2.

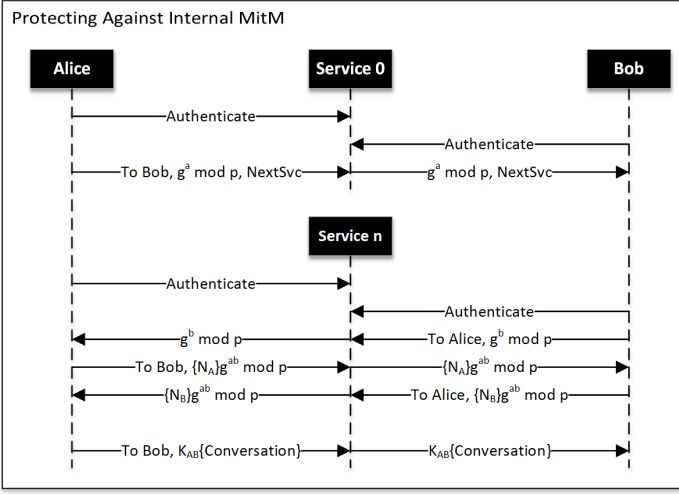


Figure 2: Protecting Against Internal MitM

Because the initiating party is not listening for a DH key exchange response on S_0 , any MitM response from S_0 to the initiating party is disregarded. This prevents this hosting server from performing an MitM. Similarly, because no party is listening for a DH exchange response on S_n , there is no opportunity this service to perform an MitM attack (Note that always choosing S_0 for the first exchange is sent is only for ease of discussion. Any service in S may be used for the first message, so long as it is different than S_n).

Having established a DH shared key, Alice and Bob can now exchange nonces that can be used to generate shared session keys for the encryption and integrity protection algorithms of their choice.

Still, this leaves opportunity for an external MitM attack. If an adversary has control of part of the route to one of the participants, such attack is still feasible, using only the protocol described by Figure 2. To address this concern, MSAP employs two strategies for addressing this vulnerability.

The first, and preferred, strategy is to use an IM service that normally performs encryption between the participants and the IM service host for at least one direction of the DH exchange. This mechanism is illustrated in Figure 3.

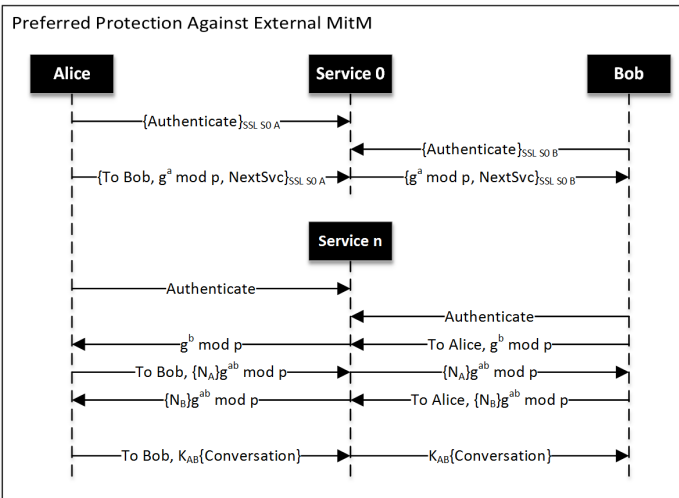


Figure 3: Preferred Protection Against External MitM

By wrapping the initial DH message from Alice to Bob in SSL established between Alice and S_0 , any attempt by Trudy to mount an MitM is thwarted, as SSL provides encryption and message authentication. As a result, it is impossible for Trudy to learn g , p , or $g^a \mod p$. This means that even if she is capable of modifying messages between Alice and all other IM services including S_n , the best she can do is make Alice and Bob disagree on $g^{ab} \mod p$. Note that this protection is also afforded when S_0 is not encrypted, but S_n is.

In the event that no IM provider offering SSL is available (or preferred by the participants), the second strategy is employed to exchange a temporary “long term” key between parties, which is then used to encrypt the DH exchange of subsequent conversations between the same parties. However, this second strategy assumes that an intruder was not present during the *initial* conversation between the parties (otherwise an intruder would know the temporary long term key). This temporary “long term” key is replaced after the DH key exchange of each conversation between the parties. This requires an intruder to be present for *every* conversation between the parties, as missing a conversation means the temporary “long term” key they knew has been replaced without them being able to observe the replacement.

Although mechanism 1 and 2 can be used independently to protect against external MitM attacks, the final version of the protocol, as defined in Section 4, calls for both to be used together, where possible. This allows for the secure establishment of a TLTS during a first SSL-encrypted session, addressing the assumption that no intruder is present during this initial conversation. Subsequent conversations are then free to perform SMAP over unencrypted connections between client and host.

Note that these ambiguities in the protocol allow for a user-configurable

6 Implementation

7 Conclusion

8 Appendix A: Protocol Extensions

There are number of ways that MSAP can be extended to add interesting functionality.

If we extend our adversary model to include collusion between multiple IM services, it is then possible for S_0 to do the first half of an internal MitM attack, and then provide the first half of the DH MitM exchange to S_n alerting it that the second half of the DH exchange will be performed on their network. S_n now has sufficient information to complete a DH MitM attack.

MSAP can also be trivially extended to work with group chats, in addition to one-on-one chats.

9 Appendix B: Username Syncing

10 References