A Protocol for Secure Public Instant Messaging*

Mohammad Mannan and Paul C. van Oorschot

School of Computer Science
Carleton University, Ottawa, Canada

Abstract. Although Instant Messaging (IM) services are now relatively long-standing and very popular as an instant way of communication over the Internet, they have received little attention from the security research community. Despite important differences distinguishing IM from other Internet applications, very few protocols have been designed to address the unique security issues of IM. In light of threats to existing IM networks, we present the Instant Messaging Key Exchange (IMKE) protocol as a step towards secure IM. A discussion of IM threat model assumptions and an analysis of IMKE relative to these using BAN-like logic is also provided. Based on our implementation of IMKE using the Jabber protocol, we provide insights on how IMKE may be integrated with popular IM protocols.

1 Introduction and Overview

Instant Messaging (IM) is a popular Internet based application enabling individuals to exchange text messages instantly and monitor the availability of a list of users in real-time. Starting as a casual application, mainly used by teenagers and college students, IM systems now connect Wall Street firms [9] and Navy warships [8]. The Gartner Group predicts that IM traffic will surpass email traffic by 2006 [31]. A survey report from the Radicati Group suggests that 85% of businesses use public IM services but only 12% use security-enhanced enterprise IM services and IM-specific policies [15].

Protocols currently used in popular public IM systems (e.g. AOL, Yahoo!, MSN and Google Instant Messenger) are open to many security threats [21]. Relying on SSL-based solutions – the most common security protocol of corporate IM systems – for security in public IM services has major limitations, e.g., messages may not be *private* when they go through the IM server [16]. Shortcomings of public and business IM protocols highlight the need of a secure IM protocol.

Contributions. We present a novel protocol called Instant Messaging Key Exchange (IMKE) for *strong* authentication and *secure* communications (see Table 1 for definitions) in IM systems. IMKE enables mutual strong authentication between users and an IM server, using a memorable password and a known server public key. IMKE provides security (authentication, confidentiality and integrity) for client-server and client-client IM connections *with* repudiation.

* Version: March 30, 2006. Contact author: `mmannan@scs.carleton.ca`.

Although pairs of users generally share no secret between themselves, IMKE enables secure and private communications among users through a trusted IM server, without revealing the contents of users' messages to the server.

An analysis of the protocol in terms of security using a BAN (Burrows-Abadi-Needham)-like logic [7] is provided.[1] The protocol has also been tested (with no flaws found) by the AVISPA (Automated Validation of Internet Security Protocols and Applications) formal analysis tool [1]. IMKE may be implemented using any well-known public key cryptosystem (e.g. RSA, ElGamal, elliptic curve) that supports encryption, without requiring any additional special constraints (unlike e.g. SNAPI [20]) for a safe protocol run.[2] In contrast, the majority of existing Password Authentication and Key Exchange (PAKE) protocols which require no known server public key are based on Diffie-Hellman (DH)-based key agreement; these must be carefully implemented to avoid many known attacks which exploit the structure of many choices of parameters in DH-based key agreement (e.g. [19]). Although IMKE has been designed as a secure IM protocol, it may also provide an alternative to other two- and three-party PAKE protocols (e.g. EKE [4]) beyond IM. IMKE may be used in server-mediated peer-to-peer (P2P) communications as well.

We have implemented a prototype of IMKE using the Jabber [30] open-source IM protocol (for details of the implementation and execution performance, see [23]). Although implementing IMKE requires changing both the IM server and client, our implementation provides evidence that IMKE may be integrated with existing public IM protocols without a large implementation effort, and keeping underlying messaging structures intact.

Organization. The sequel is organized as follows. §2 outlines motivation for IMKE and related work. In §3, we briefly discuss threats considered in IMKE, and list terminology, end user goals, and long- and short-term secrets of IMKE. The protocol messages are discussed in §4. §5 provides our IM threat model and a partial security analysis. §6 concludes.

2 Motivation and Related Work

We now discuss the motivation for IMKE, similarities and differences of IMKE with existing secure IM protocols and two- and three-party PAKE protocols.

Relationship of IMKE to Pluggable and Independent Secure IM Protocols. A *pluggable* security protocol – i.e. one that is implemented in a third-party client "add-on module" without requiring any changes to popular IM clients and servers – could easily be deployed at the client-end in addition to default IM clients. Therefore several initiatives, e.g., Off-the-record messaging [5],

[1] We do not claim to give a full proof of the security of IMKE; and moreover, no such complete formal proof would be conclusive.

[2] However, general requirements for secure choice of public key parameters must of course be fulfilled.

`Gaim-e` [25], have been taken to make IM secure using pluggable security protocols. Limitations of those proposed to date include: client-server messages remain plaintext, and the requirement of long-term client private keys, whose secrecy must be maintained.

Independent secure IM protocols developed in practice, e.g., Secure Internet Live Conferencing (SILC) [28], do not appear to have been peer-reviewed in an academic sense, nor designed to be integrated with popular IM protocols. A lightweight protocol which can easily be embedded into existing IM protocols (by IM service providers, changing both the IM client and server) seems practical to achieve security without limiting usability or requiring a large implementation effort. We propose IMKE to achieve such objectives. Although IMKE requires changes in both the client and server software, users do not need to maintain or *carry* any long-term public key. IMKE also secures client-server communications.

Relationship of IMKE to Two- and Three-Party Protocols. IM is essentially a three-party system. The IM server's main role is to enable trusted communications between users. In traditional models, a third-party is often considered a *disinterested* party [3]. In contrast, the IM server plays an active role in users' communications (e.g. forwarding users' messages). Therefore we take advantage of the presence of an active IM server in IMKE, e.g., by using the server as a trusted public key distribution center for clients.

Another major difference of IMKE with other three-party systems is that, although the IM server in IMKE helps establish a secure session between two clients, the server does not know the session key shared between the clients. This is a desirable property for consumer IM networks; users may want their conversations to be inaccessible to the IM server even though they must trust the server for login, sharing user profiles, etc.

In a typical three-party case, two users start a session[3] only when they need to communicate. The IM scenario is a bit different in the following way: users authenticate themselves only when they login to the IM server; then users initiate sessions with other online users whenever they wish to – i.e. logging in to the IM server does not necessarily precede IM sessions (e.g. text messaging, file transfer).

Two-party PAKE protocols that use a known server public key (e.g. [14]) have similarities with IMKE. These, as well as two-party password-only protocols (e.g. [4]) may be transformed into a three-party protocol in the following way: run two two-party protocols between the server and each of the users; then use the established secure channel to distribute communication primitives, e.g., public keys among users, thereby providing the communicating users a secure channel. The advantage of this approach is that several PAKE protocols are well-scrutinized, and some even come with *proofs* of security. However, we are interested in more efficient practical protocols, whereas these solutions may require up to three extra messages per protocol run – one for sending a client's public key to the server and two for verifying the public key. Also, even minor

[3] i.e. authenticating themselves to a trusted server, and each receiving a server-generated client-client session key.

modifications to an existing protocol may invalidate its security attributes (not to mention any related security proofs).

An important idea behind IMKE is to avoid number theoretic relationships between a public key and a password. IMKE uses a known server public key to encrypt a random (session) key (e.g. 128 bits) and uses that key to encrypt the (weak) user-password and the user's dynamic public key. This enables IMKE to avoid partition attacks [4].

In summary, the design of IMKE is inspired by following considerations: (1) existing IM security solutions are inadequate to address IM threats; (2) existing PAKE protocols do not directly fit into the IM communications model; and (3) a lightweight security protocol, which can conveniently be embedded into popular IM protocols without breaking underlying messaging structures, is essential for a greater integration.

3 Setup for IMKE

In this section, we discuss threats considered in IMKE. We list the notation and terminology used, end user goals, and long- and short-term secrets for IMKE.

3.1 Threats Considered in IMKE

We summarize significant IM threats which are addressed by IMKE. We defer a more concrete discussion of the IM threat model to §5.1. IMKE provides no protocol level protection against general software and platform attacks. Further discussion of IM threats is provided elsewhere (e.g. [21]).

IM connections generally involve a client and a server, or two clients. Most IM threats arise from these connections being easily compromised. IMKE aims to provide security (confidentiality, authentication and integrity protection) for all IM connections. Impersonation attacks based on compromised connections are also prevented in IMKE, assuming no theft of users' passwords, including, e.g., through the use of keyloggers. The security related goal of availability is beyond the scope of our work – i.e. denial of service (DoS) attacks against IM clients or the server are not fully addressed by IMKE. However, IMKE helps the server and clients to limit the extent of these attacks. Replay of captured messages (from an ongoing session or older sessions) is also detected in IMKE. An attacker may spoof DNS entries in a user machine (the local DNS cache) to redirect all communications to a rogue IM server. IMKE prevents this attack from being successful by authenticating the IM server to users by using a password, and verifying the known server public key (online). IMKE helps complementary techniques to restrict the propagation of IM worms[4] to be more effective by securing IM connections.

[4] e.g., throttling file transfer and URL messages, challenging the sender of a file or URL message with an automated Turing test; see [22] for details.

3.2 Notation, Goals and Secrets

We specify IMKE notation and terminology in Table 1. A password is shared between an IM server and a user. This is the only long-term secret for users and they choose their initial passwords during the IM account setup. A user may change the password whenever he/she wishes to do so. The server stores original passwords.[5] The other long-term secret is the IM server's private key (for decryption). A server public key generally remains valid for a long time (a year or more), and a key renewal is done by a client-update, i.e. by sending users the updated key when they attempt to log in. Clients' private keys (for decryption), session keys, and MAC keys are short-term secrets in IMKE. We assume that IM clients are installed with the digital certificate of the IM server.

A, B, S	Two IM users (*Alice* and *Bob* respectively), and the IM server.
ID_A	User ID of A (unique within the IM service domain).
P_A	Password shared by A and S.
R_A	Random number generated by A.
$\{data\}_K$	Symmetric (secret-key) encryption of *data* using key K.
$\{data\}_{E_A}$	Asymmetric (public-key) encryption of *data* using A's public key KU_A.
X, Y	Concatenation of X and Y.
K^s_{AS}	Symmetric (s) session (encryption/decryption) key shared by A and S.
K^m_{AS}	Symmetric MAC key shared by A and S (m is short for MAC).
$[X]_{AS}$	MAC output of data X under key K^m_{AS}.
"Strong" password protocol	A passive or active attacker should be unable to gather enough information to launch an offline dictionary attack even if a relatively *weak* password is used [4].
Secure communications	Communications where authentication, integrity and confidentiality are achieved.
End-to-end security	Securing messages cryptographically across all points between an originating user and the intended recipient.
Repudiation	A way to ensure that the sender of a message *can* (later) deny having sent it. Some [5] believe this is important for casual IM conversations.
Forward secrecy	The property that the compromise of long-term keys does not compromise previously established session keys.

Table 1. Notation and terminology used in IMKE

End-user Goals. The following are security-related goals (from end-users' perspectives) in IMKE. Terms denotated by asterisk ($*$) are defined in Table 1. Fulfilling the end-user goals corresponds to the threats we consider in §3.1. We outline how IMKE achieves these goals in §5.

G1. Assurance of server's and clients' identities to the communicating parties without exposing clients' passwords to offline dictionary attacks.
G2. Secure communications* between a client and the IM server.

[5] Alternatively, the server could store only an image or one-way hash of passwords to minimize the impact of the password (image) file exposure, although this typically still does not prevent brute force attacks on passwords.

G3. Secure communications for messages directly sent between clients (cf. G5).
G4. Forward secrecy and repudiation.⋆
G5. End-to-end security⋆ for messages that are relayed through the IM server.
G6. Detection of replay attacks on clients and the IM server.

4 The IMKE Protocol

We now introduce the IMKE protocol, along with a discussion on protocol messages. We defer a more specific security analysis of IMKE messages to §5.2.

An IM session (e.g. text messaging) between two users is established in the following phases. A and B first authenticate to the server S, then S distributes A's public key to B and vice-versa, and then the users negotiate a session key to follow an IM session. Table 2 summarizes the protocol messages for these phases. Assume for now that f_i denotes a one-way cryptographic hash function (publicly known, see further discussion below). We describe the protocol messages in the following way: (1) the password authentication and key exchange, and client-server communications, and (2) client-client communications.

Phases	*Message Labels*	*Messages*
Authentication and Key Exchange	$a1$	$A \rightarrow S : ID_A, \{K_{AS}\}_{E_S}, \{KU_A, f_1(P_A)\}_{K_{AS}}$
	$a2$	$A \leftarrow S : \{R_S\}_{E_A}, \{f_2(P_A)\}_{K_{AS}}$
	$a3$	$A \rightarrow S : f_3(R_S)$
Public Key Distribution	$b1$	$A \leftarrow S : \{KU_B, ID_B\}_{K^s_{AS}}, [KU_B, ID_B]_{AS}$
	$b2$	$B \leftarrow S : \{KU_A, ID_A\}_{K^s_{BS}}, [KU_A, ID_A]_{BS}$
Session Key Transport	$c1$	$A \rightarrow B : \{K_{AB}\}_{E_B}, \{R_A\}_{K_{AB}}$
	$c2$	$A \leftarrow B : \{R_B\}_{E_A}, \{f_6(R_A)\}_{K_{AB}}$
	$c3$	$A \rightarrow B : f_7(R_A, R_B)$

Table 2. Summary of IMKE messages (see Table 1 for notation)

4.1 PAKE and Client-Server Communications

In the PAKE phase, A and S authenticate each other using P_A, establish a secret session key, and transport a verified dynamic public key from A to S. The server's public key KU_S is verified *online*, using e.g., the *public password* [14] method, whereby users verify the hash of the server public key represented in plain English words. Then the login process between A and S proceeds as follows:

1. A generates a dynamic public/private key pair (KU_A, KR_A), and a random symmetric key K_{AS}, and then encrypts K_{AS} with the server's public key. A sends message $a1$ (see Table 2 for message labels) to S.

2. S calculates $f_1(P_A)$ independently (S looks up P_A using ID_A), compares it with the corresponding value received in $a1$, and disconnects if they mismatch. Otherwise, S generates a random challenge R_S and responds with $a2$.
3. A calculates $f_2(P_A)$ independently and compares it with the corresponding value received in $a2$, and disconnects if they mismatch. Otherwise, A calculates the session key (encryption key) K^s_{AS} and MAC key K^m_{AS} as in (4.1), and responds with $a3$.

$$K^s_{AS} = f_4(K_{AS}, R_S),\ K^m_{AS} = f_5(R_S, K_{AS}) \tag{4.1}$$

4. S independently calculates $f_3(R_S)$ and compares it with the quantity received in message $a3$. If they mismatch, S disconnects; otherwise, S also calculates K^s_{AS} and K^m_{AS} as in (4.1). S now indicates A a successful IM client login using a message of the form (4.3).

After authentication, a client and server communications include, e.g., a server sends a user's contact list, a client requests to communicate with other users. To exchange data, A and S use:

$$A \rightarrow S : \{ClientData_A\}_{K^s_{AS}}, [ClientData_A]_{AS} \tag{4.2}$$

$$A \leftarrow S : \{ServerData\}_{K^s_{AS}}, [ServerData]_{AS} \tag{4.3}$$

Caveats. f_1 and f_2 must differ; otherwise, if an attacker can replace KU_S in A's system, he can deceive A without knowing P_A, i.e. the attacker can make A *readily* believe that she is communicating with the legitimate server. Nevertheless, even when f_1 and f_2 differ, replacing KU_S with the attacker's public key in a user's machine enables an offline dictionary attack on P_A. Having different f_1 and f_2 makes the attacker's active participation in the protocol harder.

R_S and K_{AS} must be large enough (e.g. 128-bit) to withstand an exhaustive search. A must encrypt KU_A in message $a1$. Otherwise the following attack may succeed. Suppose an adversary generates a new private-public key pair, and is able to replace KU_A with the fraudulent public key in message $a1$; this enables the adversary to decrypt R_S in $a2$ and send a correct reply to S in $a3$. Hence, IMKE requires the secrecy of A's public key in the PAKE phase. Examples of secret "public keys" exist in the literature (e.g. [13]). At the end of the PAKE phase, A and S zero out K_{AS} and R_S from the program memory to help in achieving forward secrecy (see §5.3).

The duration of the session key (K^s_{AS}) should be set carefully. This is important for clients in an *always-connected* mode, wherein clients stay logged in to S for a long period of time (e.g. days or weeks). A new session key should be negotiated after a certain period (e.g. a couple of hours) depending on the expected security level and size of the session key (e.g. a shorter period for 80-bit keys than 128-bit keys) to reduce consequences from cryptographic (e.g. brute-force) attacks on the key. To do so, A and S exchange two random values K_{AS1} and R_{S1} in the following way and generate the new session key and MAC key as before (cf. (4.1)). Either A or S can begin the key renewal process. The initiator must stop sending any messages before the new keys are established.

$$A \rightarrow S : \{\{K_{AS1}\}_{E_S}\}_{K^s_{AS}}, [\{K_{AS1}\}_{E_S}]_{AS} \tag{4.4}$$

$$A \leftarrow S : \{\{R_{S1}\}_{E_A}\}_{K^s_{AS}}, [\{R_{S1}\}_{E_A}]_{AS} \tag{4.5}$$

4.2 Client-Client Communications (Direct and Relayed)

Client to client communications include, e.g., server mediated/relayed messages, file transfer, audio/video chat. If A wants to send $ClientData_A$ to B (both must be logged in to S), she first sends her request to communicate with B to S (using message type (4.2)), and then the messages below follow:

1. A and B receive the other party's current dynamic public key from S through messages $b1$ and $b2$. Note that B and S authenticate each other and derive K^s_{BS} and K^m_{BS} in the analogous way described above for A.
2. Having each other's current public key, A and B exchange messages $c1$, $c2$ and $c3$. Then A and B derive the session key K^s_{AB} and MAC key K^m_{AB}:

$$K^s_{AB} = f_8(K_{AB}, R_B),\ K^m_{AB} = f_9(R_B, K_{AB}) \tag{4.6}$$

3. Now, A sends $ClientData_A$ to B:

$$A \rightarrow B : \{ClientData_A\}_{K^s_{AB}}, [ClientData_A]_{AB} \tag{4.7}$$

Caveats. Although client-to-client connection setup messages ($c1$, $c2$ and $c3$) can be exchanged directly between A and B, we suggest they be relayed through the server using messages (4.2, 4.3) – i.e. with the additional encryption and MAC – to reduce threats from DoS attacks on clients. However, while relaying the setup messages, a malicious IM server can launch a typical man-in-the-middle attack in the following way. When A notifies S that she wants to communicate with B, S generates a public key pair for B and distributes the rogue public key to A, and vice-versa. Now S can impersonate A to B and vice-versa, and thereby view or modify messages exchanged between the users. Apparently, if users exchange the connection setup messages directly, this attack could be avoided; but, if A and B get each other's network address for direct communication from S (which is the most usual case), then this attack is still possible. The attack is made possible – albeit detectable (see below) – by the facts that, (1) pairs of users do not share any long-term secret, and (2) they do not use any authenticated (long-term) public key. Note that, this is an *active attack* where the server needs to participate in a protocol run online.

In general, IM accounts are anonymous, i.e. users can get an IM account without giving explicit identification information to the server.[6] Therefore, the

[6] From the IP address of a particular user, the server may be able to retrieve the user's location in many cases (e.g. [26]), and thereby associate an IM account to some (albeit indirect) identifying attributes of a real-world user.

motivation to launch the aforementioned man-in-the-middle attack against random users appears less rewarding for the server. In a public IM service, if the server launches this attack against any pair of users, the attack could be exposed, e.g., if that pair attempts to verify their (per-login session) public keys through, e.g., a dynamically updated web site or another service. In contrast, if using SSL (see §1), the server has direct access to end-user content, and such an attack is not necessary. Complex methods, e.g., the *interlock* protocol [29], may also be considered to expose an intruding server. An area of future research is how to reduce the trust assumptions required on the server, and yet still have an efficient relaying protocol.

At the end of the session key transport (i.e. after $c3$), A and B also zero out ephemeral values R_A, R_B and K_{AB} from the program memory. Message (4.7) is used to send $ClientData_A$ directly from A to B. For relaying data through the server, the same message type can be used. If two clients communicate for a long time (in a session), they may re-negotiate a session key (and a MAC key) in a similar way as described for the client-server key renewal.

5 Security Analysis

In this section, we provide a partial BAN-like [7] analysis intended to provide a baseline of confidence in the security of IMKE. The setup for our analysis, and other security properties of IMKE are also discussed. While BAN analysis is somewhat informal in certain aspects and is well-known to have shortcomings (e.g. [6]), it is nonetheless helpful in explaining the reasonings behind security beliefs of protocol designers, and often leads to security flaws being uncovered. However, a more rigorous security analysis as well as a *proof* of security of IMKE using alternate (non-BAN) techniques would be preferable to provide supplementary confidence. (Note however, that such a proof does not necessarily guarantee security; see Koblitz and Menezes [17] for an interesting analysis of *provable security.*) We thus consider the BAN-like analysis to be a first step.

As an important additional confidence-building analysis step, we have had the protocol tested[7] using the AVISPA (Automated Validation of Internet Security Protocols and Applications) [1] formal analysis tool. The AVISPA tool claims to be a push-button, industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications. The tool did not to find any attack against IMKE.

5.1 Setup for the Analysis

Table 3 lists definitions used in the IMKE analysis (borrowed in part from Burrows et al. [7]). Table 4 lists the technical sub-goals of IMKE which are, although idealized, more concrete and specific than the end-user goals (recall §3.2), and are of the type which can be verified from a BAN analysis point of view. The

[7] Test code is available at http://www.scs.carleton.ca/∼mmannan/avispa-imke/

analysis in §5.2 shows how IMKE achieves the technical sub-goals, and leading to the end-user goals. We also provide operational assumptions and an informal IM threat model for IMKE.

A *believes* X	User A behaves as if X is true.
A *once said* X	User A at some past time sent a message including X.
X is *fresh*	A message X is said to be *fresh* if (with very high probability) it has not been sent in a message at any time before the current protocol execution.
A *controls* X	User A is an authority on X (she has *jurisdiction* over X) and should be trusted on this matter.

Table 3. BAN-like definitions used in the IMKE analysis

T1. A and S believe that they share a (secret) password P_A.*
T2. A believes that she is communicating (in real-time) with a other party that knows S's private key.
T3. S believes that it is communicating (in real-time) with a other party that knows A's private key.
T4. A believes that she is communicating (in real-time) with a other party that knows B's private key.
T5. B believes that he is communicating (in real-time) with a other party that knows A's private key.
T6. A and S believe that they share a (secret) session key and a MAC key.
T7. A and B believe that they share a (secret) session key and a MAC key.

* See assumption A1 below; this goal is fulfilled when both parties demonstrate knowledge of the pre-established password P_A.

Table 4. Technical sub-goals of IMKE

IM Threat Model and Operational Assumptions. A *threat model* identifies the threats a system is designed to counter, the nature of relevant classes of attackers (including their expected attack approaches and resources, e.g., techniques, tools, computational power, geographic access), as well as other environmental assumptions. This IM threat model is not what would typically be expected of a *formalized* (academic) threat model, but it nonetheless provides a practically useful and clear definition of what types of attacks we intend that IMKE provides protection against. Now we list the IM threat model assumptions.

M1. The IM client software is *trusted*. By *trusted* we mean the IM client software has not been tampered with and the underlying operating system protects the IM client's memory space (RAM and virtual memory) from other programs (including malicious programs). This assumption is required as ephemeral secret keys are stored in the program memory.
M2. Communications between IM servers are secure using e.g., encryption and MAC. IMKE does not provide security for server-to-server messaging.
M3. Software and hardware keyloggers are not installed in a client system.
M4. Clients' keys stay only in program memory which are zeroed out while terminating the program.

M5. The server public key stored in client machines is verified at each login attempt (using e.g. the *public password* method [14]).
M6. Underlying communication channels need not be secure; attackers are assumed capable of viewing, altering, inserting and deleting any bitstream transfered from IM clients or servers.
M7. We consider *realistic attackers* [14] who can exhaustively search over a password dictionary (e.g. 2^{64} computational steps) but cannot defeat (in a reasonable amount of time) the cryptographic primitives (e.g. 2^{80} computational steps) used in the protocol.

We provide a few additional comments related to the above assumptions. Modern operating systems provide reasonable protection for process-memory spaces; yet, accessing a process's memory from the context of a compromised privileged (*root* or *administrator*) process is not difficult [2]. Zeroing out memory-resident secrets is not trivial [11] as well. An attacker can capture a user's password using a keylogger, i.e. a program or hardware device specialized in (secretly) recording keystrokes. Very few, if any, security guarantees can be provided in environments susceptible to keyloggers. However, threats from keyloggers are not insignificant. Also, attackers may collect passwords using social engineering techniques. Therefore, meeting the threat model assumptions in reality is not trivial. Nonetheless, these challenges are faced by many security protocols in practice. We now list operational assumptions of IMKE.

A1. Each IM user shares a user-chosen password only with the legitimate IM server (e.g. established *a priori* using out-of-band methods), and the password is not stored long-term on the user machine.
A2. The IM server's valid, authentic public key is known to all parties.
A3. Each party controls the private key for each public key pair they generate, i.e. the private key is not known or available to other parties.
A4. IMKE clients use fresh keys and challenge values where specified by the protocol, e.g., they do not intentionally reuse old values.
A5. The IM server relays clients' public keys correctly.

5.2 Analysis of IMKE Messages

We analyze IMKE messages and their possible implications in different phases of the protocol run. Refer to the earlier protocol description (§4) for the actions each party takes upon receiving a message. We start by analyzing message $a1$ (recall the message labels in Table 2). Upon successful verification of $f_1(P_A)$ by S, the locally calculated $f_1(P_A)$ by S is the same as the $f_1(P_A)$ retrieved from $a1$. Message $a1$ thus implies the following. (1) A believes that K_{AS} and KU_A are fresh, as they are freshly generated by herself. (2) Before the protocol run, S knows that it shares P_A with A. Here, S gains the evidence that the keys K_{AS} and KU_A which message $a1$ links to P_A, were generated by and associated with A. Hence, S believes the identity of A, which partially satisfies goal **T1**. (3) S believes that A once said that K_{AS} and KU_A are fresh. (4) S believes that A has a valid copy of its public key KU_S.

The successful verification of message $a2$ means that the locally calculated $f_2(P_A)$ by A is the same as the $f_2(P_A)$ decrypted from $a2$. This implies the following. (1) A believes that S knows P_A, thus satisfying goal **T1**. (2) Knowing the private key KR_S enables S to decrypt K_{AS} and KU_A in message $a1$. S encrypts $f_2(P_A)$ using K_{AS}; hence, the successful verification of $f_2(P_A)$ by A implies that A is communicating (in the current protocol run) with a party that knows S's private key, thus satisfying goal **T2**. (3) A believes that the current message $a2$ is fresh as KU_A is fresh; this provides assurance to A that the current protocol run is not a replay. (4) A believes that S once said that R_S is fresh.

The successful verification of message $a3$ by S means that the locally calculated $f_3(R_S)$ by S is the same as received in $a3$. This and the login success response from S to A imply the following. (1) S receives the evidence that A knows her private key KR_A, otherwise A could not decrypt R_S in message $a2$. Hence, goal **T3** is established. (2) The current message $a3$ is fresh as R_S is fresh; this guarantees S that the current protocol run is not a replay. (3) In message $a2$, A retrieves R_S using her dynamic private key for the current protocol run. At this point only S has a copy of A's public key. Therefore from the login success message, A believes that S possesses a valid copy of KU_A. (4) As both A and S derive the session key K^s_{AS} and MAC key K^m_{AS} from their ephemeral shared secrets (K_{AS} and R_S), goal **T6** is achieved.

From messages $b1$ and $b2$, A and B get each other's public keys from S securely. In $b1$, A receives the public key of B (KU_B) encrypted under the shared key K^s_{AS} providing confidentiality of KU_B. Also, the MAC in $b1$ provides integrity of KU_B. Message $b2$ provides similar guarantees to B for A's public key.

The successful verification of messages $c1$, $c2$ and $c3$ implies the following. (1) A believes that she shares K_{AB} with B, as only B could decrypt R_A in $c1$ and respond with a function of R_A in $c2$. (2) B believes that he shares K_{AB} with A, because only A knows KR_A which is necessary to recover R_B for use in message $c3$, and the chain of messages links R_B with R_A, and R_A back to K_{AB}. (3) A and B achieve some assurance of freshness through the random challenges R_A and R_B respectively. (4) A and B receive each other's public keys securely from a trusted source S (in messages $b1$ and $b2$). The successful verification of message $c2$ provides the evidence to A that B knows the private key corresponding to B's public key which A received earlier from S, thus satisfying goal **T4**. Message $c3$, when verified, provides the similar evidence to B, thus satisfying goal **T5**. (5) A and B derive the session key K^s_{AB} and the MAC key K^m_{AB} from their ephemeral shared secrets (K_{AB} and R_B), thus goal **T7** is achieved.

Satisfying End-user Goals. We now provide informal reasonings regarding how end-users' goals (recall §3.2) are satisfied. We argue that in the PAKE phase of IMKE, it is computationally infeasible to launch offline dictionary attacks on P_A (assuming our assumptions in §5.1 are not violated). To recover $f_1(P_A)$ from $a1$, an attacker apparently has to guess K_{AS}, which is computationally infeasible if K_{AS} is generated from a large key space (e.g. 128-bit). Another way to recover $f_1(P_A)$ is to learn K_{AS} by guessing the server's private key. Brute-force attacks on K_{AS} or KR_S appear to be computationally infeasible if the key length is chosen appropriately. To recover $f_2(P_A)$ from $a2$, an attacker must guess K_{AS}, which is infeasible. This apparently makes P_A resistant to offline dictionary attacks. As goal T1 is fulfilled in messages $a1$ and $a2$ without exposing P_A to offline dictionary attacks, IMKE achieves goal **G1**. Goal T6 establishes that A and S achieve confidentiality, and integrity (with authentication) using the secret session key K^s_{AS} and the MAC key K^m_{AS} respectively. Technical sub-goal T6, along with G1, now satisfies goal **G2**.

A and B do not authenticate each other directly. They trust the other party's identity as they receive each other's public key from S and trust S on the authenticity of those public keys. Thus fulfilling sub-goals T4, T5 and T7 provides A and B a way to communicate securely and satisfies goal **G3**.

Message authentication between A and B is achieved by MACs, instead of digital signatures. The same session and MAC keys are shared between A and B, which provide confidentiality and authentication of the messages exchanged. Any message created by A can also be created by B. Therefore the sender of a message can *repudiate* generating and sending the message. Clients' public keys are also temporary, hence binding an IM identity with a real user is technically impossible. The confidentiality of communications channels between users is protected by session keys generated from random nonces, instead of users' long-term secrets; so, the exposure of long-term secrets does not compromise past session keys. Thus repudiation and forward secrecy (goal **G4**) of users' messages are achieved (for more discussion on forward secrecy see §5.3). Direct or relayed messages (cf. message type (4.7)) between A and B are encrypted with K^s_{AB}, which is shared only between A and B (goal T7). Therefore S (or other malicious parties) cannot decrypt them, and thus goal **G5** is apparently satisfied.

If message $a1$ is replayed to a server by an attacker, the attacker cannot decrypt message $a2$ without knowing A's private key and K_{AS}. If message $a2$ is replayed to A by an attacker in a separate run of IMKE, A will refuse to reply with $a3$ as she will fail to decrypt $f_2(P_A)$ (A randomly generates K_{AS} in each run of the protocol). After A has successfully logged in to the server, A receives only messages of type (4.3) from S. Therefore, if message $a2$ is replayed to A after she logs in, A can readily detect the replay, and discard that message. If message $c1$ is replayed to B by an adversary, the adversary gains no useful information from B's reply in message $c2$. To detect replay attacks in data messages, $ClientData_A$ and $ServerData$ are appended/prepended with time-stamps or sequence numbers, with appropriate checks by the receiver (e.g. [24, p.417–418]). Freshly generated session keys and clients' public keys help in detecting replays from earlier protocol runs. Hence, goal **G6** is apparently satisfied.

Hence we have provided informal sketches of how end-user goals are satisfied.

5.3 Other Security Attributes of IMKE

Below we discuss a few more security attributes of IMKE. These properties make IMKE resistant to several recently devised attacks on security protocols.

Chaining of Messages. In the PAKE phase, messages $a1$ and $a2$ are cryptographically linked by KU_A, and messages $a2$ and $a3$ are cryptographically linked by R_S. Moreover, both KU_A and R_S are dynamically generated in each protocol run. According to Diffie et al. [12] this kind of the chaining of protocol messages may prevent *replay* and *interleaving* attacks.

Insider-Assisted Attacks. If either of A or B is a rogue user[8] participating in IMKE, we need to guard against the following attack: A or B learns the password of the other party, and the session keys that they share with other users. In IMKE, users never receive a protocol message containing any element related to other users' passwords or session keys; thus, IMKE avoids these insider-assisted attacks even when IMKE assumptions are violated by malicious users.

Exposure of Secrets. IMKE provides forward secrecy (see Table 1 for definition) as the disclosure of a client-server password (long-term secret keying material) does not compromise the secrecy of the exchanged session keys from protocol runs (using that password) before the exposure. Exposure of the IM server's long term private key allows an attacker to launch offline dictionary attacks on $f_1(P_A)$ although the attacker cannot compromise the session key or readily impersonate S. If the session key K^s_{AS} between A and S is exposed, an attacker cannot learn P_A. However, the disclosure of an ephemeral key K_{AS} (which is supposed to be zeroed out from the program memory after the PAKE phase) enables an offline dictionary attack on $f_1(P_A)$. Although the disclosure of A's dynamic private key (which exists in the program memory as long as A remains logged in[9]) enables an attacker to reply correctly in message $a3$, IMKE still provides forward secrecy.

When both the IM server's long term private key and a user's dynamic private key are exposed, an attacker can calculate the session key from the collected messages of a successful protocol run; in this case, the notion of forward secrecy breaks (for the targeted session).

In addition, IMKE is (apparently) also resistant to the *Denning-Sacco attack* [10], *many-to-many guessing attack* [18] etc. as discussed elsewhere [23].

6 Concluding Remarks

IMKE enables private and secure communications between two users who share no authentication tokens, mediated by a server on the Internet. The session key used for message encryption in IMKE is derived from short-lived *fresh* secrets, instead of any long-term secrets. This provides the confidence of forward secrecy to IMKE users. IMKE allows authentication of exchanged messages between two parties, and the sender is able to repudiate a message. Also, IMKE users require no hardware tokens or long-term user public keys to log in to the IM server.

8 For example, someone who, maliciously or naively, exposes his/her private key, password, or session/MAC keys.

9 Private keys may easily be extracted from memory as Shamir and van Someren [32] outlined, if the operating system allows reading the entire memory space by any program. However, we assume that such an operation is not allowed; see assumption M1 in §5.1.

Group-chat and chat-room [21] are heavily used features in IM. A future version of IMKE would ideally accommodate these features, as well as an online server public key verification method. Introducing methods to ensure human-in-the-loop during login, e.g., challenging with an automated Turing test, can stop automated impersonation using compromised user name and password. However, deploying such a method for large IM networks may put an enormous load on IM servers; measures as outlined by Pinkas and Sander [27] can help minimize this.

The growing number of IM users in public and enterprise world provides evidence that IM is increasingly affecting instant user-communication over the Internet. We strongly advocate that security of IM systems should be taken seriously. IMKE is a step towards secure public IM systems. Note that typical end-users of IM systems are casual. A secure IM protocol, implemented in a restrictive user interface, might force such casual users to switch to a competing product that is less secure but more user-friendly. We emphasize that usability issues must be considered while designing a secure IM system.

Acknowledgements

We thank anonymous reviewers, as well as Liam Peyton, for their constructive comments which helped us improve the quality of this paper, and all members of Carleton's Digital Security Group for their enthusiastic discussions on this topic, especially Glenn Wurster, Anil Somayaji and Julie Thorpe. We thank Paul H. Drielsma of ETH, Zurich for carrying out a security analysis of IMKE using AVISPA [1]. The first author is partly supported by a Public Safety and Emergency Preparedness Canada (PSEPC) scholarship. The second author is Canada Research Chair in Network and Software Security, and is supported in part by an NSERC Discovery Grant, the Canada Research Chairs Program, and MITACS.

References

1. A. Armando et al. The AVISPA tool for the automated validation of Internet security protocols and applications. In *Computer Aided Verification - CAV 2005*, volume 3576 of *LNCS*, 2005. Project website, http://www.avispa-project.org.
2. R. Battistoni, E. Gabrielli, and L. V. Mancini. A host intrusion prevention system for Windows operating systems. In *ESORICS'04*, 2004.
3. M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *ACM Symposium on Theory of Computing (STOC '95)*, 1995.
4. S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symp. on Security and Privacy*, 1992.
5. N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *ACM Workshop on Privacy in the Electronic Society*, 2004.
6. C. Boyd and W. Mao. On a limitation of BAN logic. In *Eurocrypt 1993*, volume 765 of *LNCS*, 1993.
7. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *ACM Symposium on Operating Systems Principles*, 1989.
8. S. M. Cherry. IM means business. *IEEE Spectrum Online*, 39:28–32, Nov. 2002.

9. ComputerWorld staff. Instant Messaging takes 'financial' twist, Apr. 2002. News article, http://www.computerworld.com/.
10. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Comm. ACM*, 24(8):533–536, 1981.
11. G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson. How to forget a secret (extended abstract). In *STACS '99*, volume 1563 of *LNCS*, 1999.
12. W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.
13. L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Selected Areas in Comm.*, 11(5), 1993.
14. S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and Systems Security*, 2(3):230–268, 1999.
15. IT Strategy Center Staff. The coming IM threat, May 2005. News article, http://www.itstrategycenter.com/itworld/Threat/viruses/coming_im_threat.
16. H. Kikuchi, M. Tada, and S. Nakanishi. Secure Instant Messaging protocol preserving confidentiality against administrator. In *Advanced Information Networking and Applications (AINA'04)*, 2004.
17. N. Koblitz and A. Menezes. Another look at "provable security". *Journal of Cryptology (to appear, 2006)*.
18. T. Kwon. Practical authenticated key agreement using passwords. In *Information Security - ISC 2004*, volume 3225 of *LNCS*, 2004.
19. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2), 2003.
20. P. D. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. In *Asiacrypt 2000*, volume 1976 of *LNCS*, 2000.
21. M. Mannan and P. C. van Oorschot. Secure public Instant Messaging: A survey. In *Privacy, Security and Trust (PST'04)*, 2004.
22. M. Mannan and P. C. van Oorschot. On Instant Messaging worms, analysis and countermeasures. In *ACM Workshop on Rapid Malcode (WORM'05)*, 2005.
23. M. Mannan and P. C. van Oorschot. A protocol for secure public Instant Messaging (extended version). Technical Report TR-06-01, Jan. 2006.
24. A. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
25. Open Source. Gaim-e. http://gaim-e.sourceforge.net/.
26. V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for Internet hosts. *ACM Computer Comm. Review*, 31(4), 2001.
27. B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *ACM Computer and Communications Security*, 2002.
28. P. Riikonen. Secure Internet Live Conferencing (SILC), protocol specification, Feb. 2004. Internet-Draft. http://www.silcnet.org/docs/draft-riikonen-silc-spec-08.txt.
29. R. L. Rivest and A. Shamir. How to expose an eavesdropper. *Comm. ACM*, 27(4):393–394, 1984.
30. P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Core, Oct. 2004. RFC 3920, Status: Standards Track. http://www.ietf.org/rfc/rfc3920.txt.
31. SecurityPark.net Staff. Instant messaging: communications godsend or security back door?, July 2005. News article, http//www.securitypark.co.uk/.
32. A. Shamir and N. van Someren. Playing 'hide and seek' with stored keys. In *Financial Cryptography - FC '99*, volume 1648 of *LNCS*, 1999.