Cryptography in Instant Messaging

INTRODUCTION

Instant messaging has become an important method of communication for private users as well as for professionals in companies worldwide. Instant messaging is a valid alternative to short message services and is becoming more popular in mobile appliances. The more wide spread this method of communication is, the more important is it to look at the security of the most common systems.

Cryptography can be applied to two different layers of the communication. Some protocols already have native build-in encryption. Others don't encrypt at all, which let to the development of cryptographic methods that work on top of the protocol between applications.

This paper gives a short analysis of different messaging protocols and compares them for their cryptographic properties. The second part will cover the introduction into cryptography beyond the protocol.

Cryptography in the protocol

I differentiate between cryptographic methods which are implemented in the message protocol and methods, that merely use the existing protocol and build on top of it. This paper will refer to the former techniques as applied to the protocol-layer independent from the OSI-protocol-layer. All instant messaging happens within the OSI-application layer.
The messaging protocol is the base implementation of the communication for the service in question. This specifies the method,message format and transport mechanism, and messages that are eventually encrypted in the process.

One very basic concept is the authentication of the user to the server, which is required by all protocols. This isn't covered in this paper.

The situation is, that most message protocols don't have any native encryption of messages. This includes many popular services like Windows Live Messenger (MSN) based on the Microsoft Notifying Protocol, Yahoo! Message Protocol, ICQ and AIM, both based on the OSCAR-protocol and finally XMPP, the only open source protocol in the list. The lack of encryption may be part of the companies' policy. AOL as well as Microsoft had or still have clauses in their Terms of Service[1][2]. which surrender copyrights of send material and information to the network's company. Another common case covered by the ToS is the disclosure of private information sent on the network to lawful requests or legal process.

On the other hand there are protocols that actually have built-in encryption. They will be described in more detail in the following sections.

1 http://www.icq.com/legal/policy.html

2 http://help.live.com/help.aspx?project=tou&mkt=en-us

Skype

Skype is the only mainstream instant-messaging service, that has native build-in encryption. It uses a combination of RSA-key-exchange and AES encryption[1].

The first part of Skype's cryptography begins with the registration of the Skype-account. The client generates an RSA-key pair and hashes the password. During the registration the password is hashed twice and sent to the global server, where it is stored for future authentication. The client and the server exchange keys, to verify and sign the user. Every username has to be unique.

At the beginning of the communication between two peers is an RSA-key exchange. After this both parties have the common 255 bit session key. The actual communication is encrypted in Integer Counter Mode. The conversation holds a stream counter and a salt. Both are encrypted using a 256 bit AES algorithm, this results in the stream key. This stream key is XORed with the plain-text of the message and creates the ciphertext, that is transmitted.

Skype published a paper on their security[2], which describe some of the procedures, however, the protocol itself is closed source and there are no public specifications. Therefore, despite the good theory, security can not be assured, because it can not be tested for backdoors.

Asked in interviews, developers neither declined nor confirmed the possibility of Skype reading conversations[3]. In october 2008 a report by Citizen Lab was released, which covered Tom-Skype censorship. TOM is the Skype-service distributer in China. It has been revealed, that the TOM-Skype-client searches messages for keywords, declines transmitting messages or sends copies of the messages and the encryption key to TOM-servers for storing. Thereby compromising messages can be decrypted and connected with the registered user[4][5]. This suggests, that a backdoor exists, although it is unknown to what extend and in which versions it's included.

XMPP

XMPP is an open source Instant Messaging Protocol. It's implementation is known as Jabber. The protocol itself has no encryption, but has hooks for client-applications to sent encrypted data. This is covered in the section about OpenPGP below.

XMPP is also the one of the few protocols, that actually allow a SSL/TLS connection to the server[6].

1 http://en.wikipedia.org/w/index.php?title=Skype_security&oldid=250807245

2 http://www.skype.com/security/files/2005-031%20security%20evaluation.pdf

3 http://www.zdnet.de/mobile/voip/0,39029944,39151472-1,00.htm

4 http://www.heise.de/newsticker/Skype-in-China-filtert-und-speichert-politische-Mitteilungen--/meldung/116853

5 http://blog.kairaven.de/archives/1655-Vorratsdatenspeicherung%2C-UEberwachung-und-Zensur-mit-TOM-Skype-in-China.html

6 http://en.wikipedia.org/w/index.php?title=Extensible_Messaging_and_Presence_Protocol&oldid=255479189#Development

SILC

SILC[1] (Secure Live Internet Conferencing) is a protocol specification for synchronous conferencing services. As such instant messaging is only one part of the whole specification[2].

SILC differentiates between communication that is established with a single individual and communication with a group (a so called channel). In the latter case several people are involved in the chat, so the communication has to exchange keys accordingly to still ensure a secure conversation for everyone.

In case of a regular chat with just one party, it uses a hybrid system based on an asymmetric key exchange and symmetric message encryption. The key exchange is handled with a modified Diffie-Hellman-key-exchange called “SILC Key Exchange”[3] (SKE). This returns the session key used for encryption in this chat session. To protect against man-in-the-middle-attacks, the DH is extended with the additional use of digital signatures to ensure the identity of the parties involved in the key-exchange.

Dealing with a channel is slightly different. The channel has a one channel-key, which is exchanged to all users joined in this group. This key is frequently regenerated and exchanged, for example when new users join or leave. This is to ensure, that users who left the channel aren't able to decrypt messages by sniffing. The channel-key can be automatically generated and exchanged in the channel or it can be a passphrase which is negotiated by the participants beforehand, in the latter case, of course without refreshment of the key.

In both cases, the actually communication is always encrypted using a symmetric algorithm. Message authentication is done using Message Authentication Code (MAC) algorithm.

File transfers are encrypted as well.

Cryptography beyond the protocol

In most cases the messaging protocol doesn't include any cryptographic methods. To allow secure communication in spite of this , several systems have been developed, that use cryptography within the boundaries of the messaging protocol. Hereby, the cryptography is not done by the protocol, but between the communicating applications. The protocol doesn't “know”, that it's doing a key-exchange or sending encrypted data. This logic is handled end-to-end by the clients.

The biggest disadvantage of this is, that not all clients support all security systems. Especially the official clients used for the popular instant messaging networks like Windows Live Messenger and ICQ/AIM don't support any cryptographic methods. This is unfortunate, as most users rely on the official chat clients and don't switch to other software just because it is supposed to be more secure.

1 http://de.wikipedia.org/w/index.php?title=SILC&oldid=53536026

2 http://www.silcnet.org/support/documentation/wp/silc_protocol.php

3 http://silcnet.org/docs/draft-riikonen-silc-ke-auth-09.txt

SecureIM

In 2001 a multi-protocol client called Trillian became popular. It enabled the use of the most common protocols at that time in one application. Version 0.72 in December 2001 introduced a new feature called SecureIM[1], which offered the possibility to communicate with other Trillian users encrypted. SecureIM only supports the OSCAR-protocol used by ICQ and AIM, while there are many other protocol supported by the client. As SecureIM was developed by the Cerulean Studios without, AOL tried to block its use, but eventually ceased attempts. The biggest advantage of SecureIM was the easy use for the end-user. If possible, Trillian applied it automatically and so the user didn't have to do anything to get encrypted messaging. While it was limited to the Trillian-client at first, SecureIM is also available for Miranda IM-client as a plugin[2].

SecureIM is also a hybrid system combining an asymmetric key-exchange and a symmetric encryption algorithm. The key-exchange is handled using a Diffie-Hellman-exchange to generate a common session-key. This key is only of 128bit length[3]. There are no protections against man-in-the-middle-attacks in place.
The actual encryption of messages is done using the Blowfish-algorithm with the session-key. Since the key is so short, the encryption can be broken easily using brute force. Therefore the system can be used for private communication, but is unsuitable for the transport of important information.

SimpLite

SimpLite is a series of proprietary IM-clients for Yahoo, MSN, ICQ/AIM and XMPP. Currently the software is only available for Windows-platforms. Versions for Linux and Mac OS X are planned. Each program is a stand-alone and allows to connect to the corresponding network. On establishing communication, it exchange keys up to 2048bit using the RSA-exchange. Once the key is negotiated the messages are encrypted using AES or Twofish[4].

Off-the-record

Off-the-record[5] (OTR) is a system that enables encrypted communication on otherwise non-encrypted messaging protocols. On the one hand it allows encrypted messages and also authenticates the chat-partner, but still offers deniability. OTR is distributed in a series of plugins for Pidgin, Miranda, Kopete and AdiumX, covering the biggest open source IM-clients[6]. OTR is most spread in the XMPP-network, but also supports cross-network encryption using

1 http://en.wikipedia.org/w/index.php?title=SecureIM&oldid=230949368

2 http://addons.miranda-im.org/details.php?action=viewfile&id=2445

3 http://www.mail-archive.com/cryptography@metzdowd.com/msg08129.html

4 http://www.secway.fr/us/products/simplite_jabber/tech.php

5 http://en.wikipedia.org/w/index.php?title=Off-the-Record_Messaging&oldid=251788296

6 http://www.cypherpunks.ca/otr/index.php#downloads

XMPP-transports to other networks. Another implementation are OTR-proxies, which intercept messages and encrypt them with OTR. However, this only supports the OSCAR-protocol at the moment.

The OTR specification is open source and can be accessed by everyone[1]. OTR comes in different versions, that developed over time. Changes are mainly in the authentication methods. Version 1 had a fingerprint signing, later versions also included automatic signatures using Message Authentication Codes. The communication begins with an unauthenticated Diffie-Hellman-key-exchange. Once the secure channel is established the authentication is performed. This authenticated key exchange AKE is similar to the SIGMA-protocol. At the end of the exchange, both participants have a common encryption key and have verified each others authentication using a MAC-protocol.

Once the common key is shared, it is used in an AES message encryption. For every message a new key is negotiated where the previously used key is the computation basis for the next generated key. This way each message is encrypted uniquely and keys change accordingly often. This ensures forward-secrecy, because even if one message gets compromised, not the whole communication is. The Socialist Millionaires' Protocol (SMP) can be used to compare the common keys on both ends, without revealing more information than the fact that the common key is identical or not. This also protects against Man-in-the-middle attacks. OTR only works for regular two-party-chats, group chats are not supported.

One important features of OTR is the deniable authentication. Since the authentication is handled within the encrypted channel, from the outside the authenticity of the communication can not be proven. Thereby every user can later claim that this conversation has never happened. This is a matter of interest, whether or not somebody wants to be able to deny the conversation taking place later on, or whether the other one wants to have proof of the conversation. In that case, a communication with PGP-encryption and Digital Signatures is the better way.

OpenPGP

Implementations of OpenPGP offer tools to encrypt and decrypt messages manually. Of course you can copy-paste texts into the IM-client and send them. This could work for any protocol, but is not very practical.
XMPP, while natively sending unencrypted plaintext, supports defining the type of the sent data. Messages can be encrypted or digitally signed. The XMPP-protocol can be told, that encrypted data using OpenPGP[2] are being sent and the other end knows what it's receiving. Still the implementation of the encryption is handled by the applications.

To use this method of encryption the IM-client requires an interface for OpenPGP to hook in. This either has to be built-in or added as a plugin. OpenPGP, for example Gnu Privacy Guide (GPG) has to be installed on the system and will be used by the client. GPG handles the key-exchange, key-signingand encryption.

1 http://www.cypherpunks.ca/otr/Protocol-v2-3.1.0.html

2 http://xmpp.org/extensions/xep-0027.html

The biggest disadvantage of GPG-based message encryption is the complexity. For the novice user it's not trivial to set up and it requires a basic understand of how it works. It works securely, but the effort users have to invest scares a lot of people. Several instructions how to set up GPG in some common clients are attached to this paper.

Conclusion

After analyzing the most common instant-messaging protocols, it becomes obvious, that security is most-often not an important part of the specification. Either because of policy decisions, for legal reasons or a lack of interest. The problem is, that secure communication is difficult to handle for the user. Cryptography is complex and it needs to be made simple,and as transparent as possible to the user. Skype and Of-the-record-encryption are pursuing this approach, by hiding the cryptography as as much as possible from the user, but still ensuring a secure encryption mechanism working in the background.

As always, security systems are only as good as they have been analyzed. To analyze it, you require access to the specification and the implementation. SecureIM has been breached a long time ago. Skype has a rumored backdoor. Open source implementations like GnuPG and OTR can and are being reviewed for security. Therefore they are good choices for secure communication. For private conference networks, SILC could be a valid solution.

Attachments:

1. **Verschlüsseltes & anonymisiertes Instant Messaging per Jabber**
   Umfangreiche Vorstellung von OTR und GPG für das XMPP-Protokoll, Anleitung zur Einrichtung von Psi mit GPG, Pidgin mit OTR und Tor.
   http://hp.kairaven.de/jabber/index.html
2. **Howtos für verschlüsseltes Instant Messaging**
   Anleitung zum Einrichten von OTR in Pidgin, sowie OTR und GPG in Kopete
   http://freiheitblog.wordpress.com/2008/02/25/verschlusseltes-instant-messaging-teil-1-einleitung/
3. **Verschlüsselt Chatten mit Pidgin und OTR (bebildert, Windows)**
   Anleitung zum Einrichten von OTR in Pidgin
   http://datenschutz-wuerzburg.net/2008/01/19/verschlusselt-chatten-mit-pidgin-und-otrbebildert-windows/
4. **Client Konfiguration**
   Anleitungen zur GPG-Einrichtung in verschiedenen Clients
   http://jabber.rwth-aachen.de/wiki/OpenPGP#Client_konfigurieren
5. Liste von Clients mit Off-the-record-Unterstützung:
   AdiumX (Mac), Pidgin, Miranda IM, Kopete (Linux)
6. Liste von XMPP-Clients mit GPG-Unterstützung:
   Psi (Windows), Pidgin, Gajim, Kopete, Jabbin, Jbother, Centericq