Secure Off-the-Record Messaging

Mario Di Raimondo
Dipartimento di Matematica ed Informatica
Università di Catania, Italy
diraimondo@dmi.unict.it

Rosario Gennaro
IBM T.J.Watson Research Center, USA
rosario@watson.ibm.com

Hugo Krawczyk
IBM T.J.Watson Research Center, USA
hugo@ee.technion.ac.il

ABSTRACT

At the 2004 Workshop on Privacy in the Electronic Society (WPES), Borisov, Goldberg and Brewer, presented "Off the Record Messaging" (OTR), a protocol designed to add end-to-end security and privacy to Instant Messaging protocols. An open-source implementation of OTR is available and has achieved considerable success.

In this paper we present a security analysis of OTR showing that, while the overall concept of the system is valid and attractive, the protocol suffers from security shortcomings due to the use of an insecure key-exchange protocol and other problematic design choices.

On the basis of these findings, we propose alternative designs and improvements that strengthen the security of the system and provide the originally intended features of the protocol, including deniability, in a sound and well-defined sense.

Categories and Subject Descriptors

K.4.1 [**Computer and Society**]: Public Policy Issues – *Privacy*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Authentication*; E.3 [**Data**]: Data Encryption

General Terms

Security

Keywords

Perfect forward secrecy, deniability, authentication, instant messaging

1. INTRODUCTION

The Internet has introduced new ways in which people communicate. The most common is Electronic Mail (or email) which allows people around the world to communicate in a fast and very efficient way. Due to its success, nowadays the email system is entrusted with all forms of information, including very sensitive data. As a consequence, the need to secure the email infrastructure has received plenty of attention, and solutions such as PGP [4, 27] and s/MIME [25] are widely available. Such solutions are designed to provide with the three pillars of secure communications, namely:

Confidentiality: the content of communications should remain secret: an unauthorized person should not be able to learn any private information.

Authentication: the recipient of information should have certainty about the sender of the information; no other person (or entity) should be able to impersonate the legitimate sender.

Integrity: Unauthorized changes to information transmitted between legitimate parties must be detected by the receiver.

To achieve these properties cryptographic techniques are applied, with encryption functions used to provide confidentiality, and digital signatures and message authentication codes to provide for authentication and integrity.

1.1 Instant Messaging

In addition to email, a Web communication tool that has gained immense popularity is Instant Messaging (IM). Through IM programs one can "chat" with other on-line users in a way that is more interactive, timely, and fun than through email. In addition, although it originally started as a recreational tool, IM is becoming more and more common inside corporations as an essential collaboration tool.

As it happened with electronic mail, this newer tool is going through a second phase of maturation. In particular, the use of IM technology both in private personal communications as well as in sensitive business settings requires that security protection be added to these tools.

While many of the popular IM solutions, such as MSN Messenger and Yahoo, do not provide any means of protection, others have already started adding security functionality to their IM programs. Such is the case of AOL that provides protection similar to s/MIME and Trillian's SecureIM which offers data encryption but not authentication. These initial examples indicate a growing awareness for the need to secure IM applications. Before continuing it is worth pointing out that existing security tools such as SSL and VPNs can provide protection of IM traffic between servers but are insufficient in most implementations to secure the traffic between the end points to the communication. Such an end-to-end security is required to protect personal communications as well as needed in many business scenarios.

An important step towards the establishment of widespread mechanisms for protecting end-to-end IM communications has been taken by Borisov, Goldberg and Brewer [3]. They developed a security design for IM based on the premise that not only IM requires end-to-end security but the specifics of the medium and its applications require special security considerations. In particular, they observe that well established end-to-end tools such as PGP (designed to protect email traffic) fall short of providing the security functionality required in the IM setting. Interestingly, while more demanding, the interactive character of the IM setting also presents some advantages over off-line settings such as email.

Indeed, in the case of email, when Alice wants to send an email message to Bob, she cannot engage in a security exchange with Bob (who may not be connected to the network at all), but rather Alice will use Bob's public key to encrypt the message before sending it. An adversary (Eve) can now sit on the network and collect all these encrypted emails, which at this point are useless to her. But what if at some future point Bob's private key gets compromised and exposed to Eve? This is a realistic possibility given that many of these keys are not well protected (insecure storage, easy to guess PINs or passwords, etc.). With the knowledge of Bob's private key, not only can Eve read Bob's future communications, but she can also read any of the past messages addressed to Bob that were protected under this key! (All is needed is that Eve recorded the encrypted communication.)

In an interactive settings as IM, this drawback can be avoided by using systems that provide the so called "perfect forward secrecy" (PFS) [12, 8]. In this type of systems, the exposure of a secret key (even a long-term private key) doesn't compromise the security of past transactions. The basic idea is that long term secret keys are used for the purpose of authentication only, while encryption itself is performed using short-term session keys that are erased at the end of a session. As long as these session keys are not recoverable from past transcripts and the long-term secret keys then PFS is ensured. The interactive nature of IM systems allows to design a security solution that achieves PFS.

Another issue to consider is privacy, intended not just as the desire to keep the contents of communications secret (we refer to the provision of such secrecy as confidentiality), but also the desire of keeping private the very fact that a communication between two given individuals ever took place. In particular, cryptographic means applied for securing the communications *should not leave a proof verifiable by a third party (say a judge)* that a particular conversation took place. Ideally, any party to a conversation should be able to *deny* its contents even if the peer to the communication purposely misbehaves to obtain such a proof. Obviously, this requirement conflicts strongly with the desire for authenticity: if Alice requests a proof that a certain message came from Bob, how can Bob later deny that he ever sent that message?

Again if the communication is interactive, there are solutions to this quandary. The idea is that Bob provides a proof that is convincing only for Alice and nobody else. As said, even Alice should not be able to later prove that the exchange ever took place. This kind of authentication, with a "deniability" feature as above, is called deniable authentication. The conflict between authentication and deniability features is best illustrated by digital signatures whose main feature (and the main reason to use them in some applications) is the provision of non-repudiation (namely, the ability to prove to a third party that a signer committed to a document or other piece of information). However, non-repudiation is the opposite of what deniability asks for. Yet, as we will see later, one can use signatures in a careful way in order to preserve deniability. Additional techniques for supporting deniability are provided by encryption functions and message authentication codes (MAC).

An early example of a protocol providing support for this feature is SKEME [17] which was designed in order to provide a deniability option to IPsec's IKE protocol [13]. In last years deniable authentication has received some more formal attention in the cryptographic literature [17, 11, 23, 14, 21, 9].

1.2 Off-the-Record Messaging

The work of Borisov, Goldberg and Brewer [3] from the 2004 WPES workshop was the first to tackle the above security issues in the context of Instant Messaging. Their contribution has been both at the level of establishing requirements for security solutions in the context of IM as well as in designing (and implementing) specific mechanisms to achieve these requirements. They proposed "Off-the-Record (OTR) Messaging" as a security mechanism to adopt on top of a generic IM protocol. The goal was to obtain desirable properties such as privacy, authenticity, PFS and deniability. Their solution roughly consists of two phases: an authenticated key-exchange phase where a first session key is established and a subsequent re-keying phase where session keys are renewed.

The authors of OTR have also created an Open Source project to embed their protocol in the major IM programs. At present the project hosts a native plugin for the multi-protocol client GAIM[1] and a proxy server that virtually permits its use with any IM client. The OTR project has achieved considerable success in the Open Source community. It could be considered a first step toward a standard for IM security.

1.3 Our Contribution

The goal of this paper is to examine in depth the principles behind the design of OTR in an effort to improve its already good features. Our analysis points to some flaws in the cryptographic design of OTR, including feasible attacks against the authentication features of the key-exchange protocol. We also discuss the need to refine some of the notions used to argue about the security of the protocol. On the basis of this analysis, we propose alternative mechanisms for achieving well-founded security without sacrificing (in some cases, even improving) the performance of the protocol. Most importantly, the proposed designs are all rooted in provably secure mechanisms, both at the level of secure key-exchange protocols as well as in regards to well-defined deniability properties. The rest of the paper is devoted to presenting these issues and alternative designs for OTR. Some of the improvements proposed in this work are being considered for inclusion in a future version of the official OTR protocol.

[1]GAIM is available in different platforms like Windows, Linux and other Unix-like OS.

2. THE OTR MESSAGING PROTOCOL

In this section we describe the "Off-the-Record Messaging" (OTR) protocol from [3]. OTR was designed to provide security features for an underlying instant messaging (IM) protocol. IM protocols are characterized by a sequence of messages exchanged between two parties but not necessarily alternate, that is a party could send more than one message before he or she receives a reply. (This property matters in the re-keying process as we will see below.)

The OTR protocol uses the underlying IM protocol as a transport layer, namely, each binary message of the OTR protocol is converted into characters that are transported by the IM protocol. Upon receipt, this message is converted back to the original format. Each OTR message is marked by a special tag so it is recognizable.

The OTR protocol consists of two phases: a first phase where an authenticated key-exchange is performed obtaining a shared session key; a second phase consisting of a continuous refreshment of the session key during the exchange of IM messages.

2.1 The OTR Authenticated Key-Exchange

To establish a shared secret key the OTR protocol uses the well-known Diffie-Hellman Key-Exchange [7].

Briefly, there are some public parameters: a prime p and a generator g of a subgroup of Z_p^* of large prime order q. Alice and Bob pick two random numbers (the *secret exponents*), x and y in $\{1, \dots, q\}$, respectively, and they exchange $X = g^x$ and $Y = g^y$ (computed mod p) over a public channel. At this point both parties can compute a shared secret value that is computationally unknown (under proper assumptions) to any eavesdropper. This secret value is g^{xy} and is computed by Alice as Y^x and by Bob as X^y. The shared key K is then set to a suitable hash of g^{xy}, i.e., $K = H(g^{xy})$ [18, 10], and the exponents x, y are erased. Even though the eavesdropper sees X and Y, it is believed that computing g^{xy} from these values alone is intractable.

The basic DH protocol provides security only against passive eavesdroppers. If the adversary is capable of interfering with the messages being sent, then it is extremely easy to mount a man-in-the-middle attack. In particular it is very easy for the attacker to impersonate Bob and fool Alice into believing that she is having a private conversation with Bob.

For this reason it is essential to use an "Authenticated Key-Exchange" (AKE) protocol which provides the means, on top of the Diffie-Hellman exchange, to authenticate the parties and guarantee that the key is solely known, and uniquely bound, to the legitimate parties to the exchange. Authentication of a Diffie-Hellman exchange is often done using the public keys of the peers (though there are other mechanisms, like pre-shared keys and passwords). The basic goals are, therefore, to guarantee that honest parties are not fooled to believe in the wrong identity of a peer to an exchange, that both parties have consistent views of who the peer to the exchange is, and that only the legitimate peers may possibly know the value of the exchanged key.

In order to add authentication to the basic DH protocol, OTR uses public keys and digital signatures. Specifically, each party A in the OTR network has a pair of secret/public keys $(\mathsf{sk}_A, \mathsf{vk}_A)$ for a digital signature scheme (implemented using either DSA or RSA signatures).

In a signature scheme, the secret key sk_A is used to create valid signatures by the owner of the key such that no other party can create valid signatures. At the same time, anyone, using the public key vk_A, can verify the validity of any given signature. Note that the public key is associated to the identity of a specific party. OTR adopts a simple and non-hierarchical approach to the distribution of public keys, where each party stores the public keys of the users he communicates with. When first entered users are prompted to verify validity of the public key via fingerprint recognition, much like in SSH [26].

The OTR authenticated key-exchange phase requires that each party signs its Diffie-Hellman value. The public key is sent with the first message:

$$\begin{array}{ll} A \rightarrow B: & Sign_{\mathsf{sk}_A}(g^x),\ \mathsf{vk}_A \\ A \leftarrow B: & Sign_{\mathsf{sk}_B}(g^y),\ \mathsf{vk}_B \end{array}$$

If the public key vk_A is already stored by B and it is associated with the identity of A, this assures B that g^x comes from A and vice versa (in the absence of pre-stored public keys the protocol could use PK certificates). After the verification of the signatures, both parties compute their shared secret value g^{xy} and erase the DH exponents.

The main goals of the above protocol, as stated in [3], are to provide a key exchange that guarantees authentication, PFS, and deniability. As we show below, however, this AKE protocol is not a secure key-exchange protocol as it is open to authentication vulnerabilities and other attacks. Moreover, a straightforward fix of the protocol results in the loss of its deniability properties. We discuss the security shortcomings of the OTR AKE in Section 3 and propose alternative solutions in Section 4.

2.2 The key refreshment

Once the first session key is established, OTR engages in a very fine-grain key refreshment procedure. Basically for each message sent each party generates a DH exponent which will be used to generate a new DH key. Authentication is performed using the key shared in the previous message. Once the new key is established it will be used to encrypt and authenticate messages, while the previous one is erased.

Because, as we discussed, IM messages do not necessarily follow an alternate pattern between the two parties, care is necessary to make sure that the right DH exponents are "paired" by the parties to generate the new key. Basically, instead of sending a DH exponent for each message, Alice will send one only after she received a response from Bob. Key IDs are enclosed to ensure that both parties know which key is being used.

2.3 Encryption and authentication of messages

Messages exchanged during a OTR session are encrypted and authenticated. The message is first encrypted using AES in counter mode and then the resulting ciphertext is authenticated using HMAC (with hash function SHA-1) [16]. As discussed above, the new DH exponent is also included under the MAC argument in order to authenticate it.

The choice of a counter mode of operation is justified by the designers of OTR with the desire of having a *malleable* encryption scheme in order to increase deniability (see below). Recall that in a malleable encryption scheme given a ciphertext it is possible to modify it to obtain a ciphertext of a related plaintext. This is clearly easy to do with a counter (or other stream-cipher) mode of operation where flipping a bit in the ciphertext results in the flipping of the corresponding bit of plaintext.

Another mechanism used by OTR for the purpose of deniability is that after a new key has been established via the key refreshment mechanism, the previous MAC key is *revealed*. This way the MAC on previous messages will cease to have authorship value associated with Alice or Bob, since once the key is revealed anybody could create a valid message/MAC pair with that key.

The reason behind the above choices (i.e., a malleable encryption and revealing the MAC keys) is deniability. A valid ciphertext/MAC pair cannot be associated with Alice or Bob, because anybody could have created a ciphertext that decrypts correctly (by simply modifying another valid ciphertext, via the malleability of the scheme) and then computed a valid MAC on it (since old MAC keys are made public).

3. SECURITY ANALYSIS

In this section we analyze the security properties of OTR and point out some flaws and attacks, mainly to the AKE component. In Section 6 we describe further design shortcomings in other parts of the protocol.

The design of an AKE protocol is a critical task. This topic has been the subject of many works in the literature. Defining and proving security for AKE protocols is not an easy task as can be seen from the long list of protocols that have been broken and discarded along the way. Unfortunately, the OTR AKE protocol suffers of such weaknesses too. The good news is that based on the significant progress made in the cryptographic literature in last years regarding the design and analysis of AKE protocols we are able to spot the OTR's weaknesses and, at the same time, offer secure (and equally efficient) alternatives.

3.1 An authentication failure

One of the obvious goals of an AKE is to guarantee that the identities of the real participants in the protocol be known to each of the peers to the exchange and the key known only to these parties. If the protocol successfully terminates (i.e., both parties compute a shared key) then we want that both parties have computed the same key, and that this key is associated to the right identities. That is, both Alice and Bob must end with the same key K, and have consistent views of who the key was shared with.

Diffie *et al.* [8] have shown that the association, or binding, between keys and identities is trickier than one may perceive in a first place. In developing an authenticated DH protocol they showed an attack in which the attacker Eve interferes between Alice and Bob in a way that both parties end computing the same key but while Alice believes that the peer to the exchange is Bob, Bob believes that the key was exchanged with Eve. Therefore, communications protected using this key will be considered by Bob as coming from Eve and not from Alice. In [8] it is shown how this can be used by Eve to defraud a customer Alice and a bank Bob. Other consequences from such an authentication failure have been later shown in applications ranging from personal relations to military applications. Since its discovery by Diffie *et al.* this attack has become a basic test for sound designs of key-exchange protocols (the attack has been referred to by several names, including the “unknown key share (UKS)”, “source substitution” and “identity misbinding” attacks).

Unfortunately, the signed Diffie-Hellman key agreement adopted in OTR (and described in Section 2.1) is subject to this form of attack, as showed next. Here Eve (E) acting as a person-in-the-middle runs two conversations at the same time, one with Alice and one with Bob. The DH values sent from Alice to Bob are relayed by Eve to Bob but this time under Eve's name (all Eve needs to do is to sign this DH value under her own private signature key). On the other hand, the response from Bob (intended to Eve in Bob's “mind”) are relayed without change to Alice. Pictorially (the symbol $E[B]$ denotes the fact that Eve acts in this message as an interface to Bob):

$$\begin{aligned} A \rightarrow E[B]:&\ g^x,\ Sign_{\mathsf{sk}_A}(g^x),\ \mathsf{vk}_A \\ E \rightarrow B:&\ g^x,\ Sign_{\mathsf{sk}_E}(g^x),\ \mathsf{vk}_E \\ E \leftarrow B:&\ g^y,\ Sign_{\mathsf{sk}_B}(g^y),\ \mathsf{vk}_B \\ A \leftarrow E[B]:&\ g^y,\ Sign_{\mathsf{sk}_B}(g^y),\ \mathsf{vk}_B \end{aligned}$$

The established key g^{xy} is equal for both sessions, but Alice associates this key to the identity of Bob, and Bob associates it to the identity of Eve. Even though Eve doesn't know the key, she is able to forward messages between the two sessions with the result of these messages being associated to the wrong identity by Bob, with fraudulent consequences as mentioned above.

A simple way to overcome the attack is to include the identity of the intended receiver in the signature. This way it becomes impossible to forward signed messages to sessions in which different identities are involved. This signed DH agreement would become:

$$\begin{aligned} A \rightarrow B:&\quad g^x,\ Sign_{\mathsf{sk}_A}(g^x, B),\ \mathsf{vk}_A \\ A \leftarrow B:&\quad g^y,\ Sign_{\mathsf{sk}_B}(g^y, A),\ \mathsf{vk}_B \end{aligned}$$

Unfortunately, by doing so the protocol lost the deniability feature that motivated it. Indeed, in the above protocol, each participant is leaving a full, non-repudiable, proof that it communicated with the other party.

3.2 A freshness-impersonation vulnerability

In addition to the above attack, the OTR key-exchange protocol suffers of an additional vulnerability that questions the resistance of the protocol to full impersonation attacks. A well-designed key-exchange protocol should have the property that the *only way* an attacker can impersonate Alice in key-exchange sessions is by compromising the long-term private key used by Alice for the purpose of authentication (indeed if such key is leaked to the attacker, the latter can assume the identity of the victim). The exposure of any other piece of information, such as session-specific values, should be of no use for the attacker to impersonate Alice in other sessions. In other words, a basic security requirement is that *the exposure of ephemeral session-specific secrets should have no bearing on the security of other sessions.*

The OTR protocol, however, does not provide such guarantee. Since the signature of Alice on the value g^x does not carry any freshness indication, then Bob has no means to verify the freshness of the signature. Hence, an attacker that finds a *single* ephemeral value x used by Alice, will be able to impersonate Alice to *any other party in the system* for as long as the public key of Alice is not revoked! All the attacker needs to do is to replay the pair g^x, $Sign_{\mathsf{sk}_A}(g^x)$. Since it knows x it can compute the session key for whatever response g^y comes from Bob.

This vulnerability to the disclosure of ephemeral information is particularly serious in the case of DH exponents since one way to handle the cost of DH exponentiations (especially in low-powered devices) is to pre-compute pairs (x, g^x) and store them in memory until they are needed. As such, these exponents may be much more vulnerable to attack than the long-term private key of a party. Even in cases where DH exponents are not pre-computed one may hope to have the long-term key better protected (e.g., in a hardware token) than temporary DH exponents generated for a specific session (and possibly stored in a shared computer from which the user may occasionally run an IM session).

Notice that if we modify the protocol to include the identity of the receiver Bob in the signature (as suggested for preventing the identity misbinding attack), impersonation of A to B is still possible if a pair (x, g^x) used by A in an exchange with B is ever learned by the attacker.

3.3 Deniability

In a deniable authentication protocol there are two seemingly conflicting requirements: Alice wants to be convinced that she is really talking to Bob; on the other hand Bob does not want Alice (or any other entity) to be able to prove to a third party that Bob said something or, in some cases, even that Bob ever talked to Alice.

Deniability can thus be defined at various levels: for example Bob may be willing to admit that he talked to Alice, as long as he can deny the content of the conversation. The strongest definition of deniability is when Bob can deny that the conversation ever took place. Usually this condition is defined by enforcing a *simulation* requirement [11]. A protocol is fully deniable if there exists an efficient algorithm (called a *simulator*) that produces transcripts indistinguishable from the real ones *without* knowing the secret keys of the parties. If such a simulator exists, then any conversation can *a posteriori* be denied as it could be the product of the simulator and not of the real party. That is, a transcript of communication shown by Alice does not constitute a proof that the transcript was generated in a conversation with Bob since Alice could have produced the transcript by simply using the simulator.

The OTR protocol uses digital signatures for authentication. Clearly, Alice cannot generate signatures in the name of Bob without knowing Bob's secret key, therefore strictly speaking the protocol cannot be fully simulated. On the other hand, since the only signatures produced by the protocol are on random values, and independent of the communication with Alice, they only constitute a proof that Bob was involved in some conversation but this cannot be traced to the specific peer. Such a level of deniability is often sufficient in practice.

However, if we consider the fix for the OTR AKE introduced in Section 3.1 where the identity of the intended receiver is included under the signature, then the signature is not on a random value anymore. This signature constitutes an undeniable evidence that Bob has talked specifically to Alice and this stronger fact could have further implications (for example, proving collusion in a crime or just a love affair). Thus the OTR AKE from [3] does not satisfactorily solve the conflict between authentication and deniability: the proposed deniable protocol does not ensure strong cryptographic authentication, and once we fix the authentication part the protocol ceases to be deniable.

In the next section we give several AKE proposals which are both secure (i.e., good at authenticating the parties) and deniable (with varying levels of deniability as discussed in Section 5.1). Their efficiency is comparable, and even better in some cases, than the original OTR protocol.

4. BUILDING A SOUND AKE FOR OTR

Fortunately, there is no need to invent a new key-exchange protocol for the purposes of OTR. Several, well-analyzed and proven secure protocols exist that fit the needs of OTR messaging. Here we discuss some of these options. In Section 5 we expand on the privacy properties of these protocols.

4.1 SIGMA

SIGMA is a signature-based authenticated DH exchange [18] adopted as the main key-exchange protocol in IKE (versions 1 and 2 [13, 15]). The protocol has been formally analyzed and proven secure in [6]; in particular, SIGMA solves the weaknesses of the OTR AKE protocol as discussed in Section 3. The protocol is outlined next.

$$\begin{array}{ll} A \rightarrow B: & g^x \\ A \leftarrow B: & g^y \\ A \rightarrow B: & \text{“}A\text{”},\ Sign_{\mathsf{sk}_A}(g^y, g^x),\ MAC_{K_m}(\text{“0”}, \text{“}A\text{”}),\ \mathsf{vk}_A \\ A \leftarrow B: & \text{“}B\text{”},\ Sign_{\mathsf{sk}_B}(g^x, g^y),\ MAC_{K_m}(\text{"1”}, \text{“}B\text{”}),\ \mathsf{vk}_B \end{array}$$

Here “A” and “B” denote the identities of Alice and Bob and the MAC key K_m is derived by hashing the value g^{xy}. The session key is derived also from g^{xy} in a (computationally) independent way from K_m (see [18] for full details and rationale).

The use of the MAC value in the protocol serves two purposes: it prevents the identity misbinding attack and, at the same time, it provides a deniable exchange by avoiding signing the peer's identity.

The 4-message SIGMA protocol as described above is called SIGMA-R. A 3-message variant, called SIGMA-I, can be obtained by inverting the order of the 3rd and 4th messages. This has significance in the form of identity protection that the protocol may provide if these last two messages are encrypted (see Section 5.2). As we will see in Section 5.1, SIGMA-R has better deniability properties than SIGMA-I.

Note that SIGMA requires more messages than the two in the original OTR protocol. If the use of a 2-message protocol is considered important, the next two subsections present such protocols at the expense of a weaker from of PFS.

4.2 SKEME

As noted earlier, the SKEME protocol [17] constitutes an early example of a protocol designed as both a secure AKE as well as to support deniable communications. The original SKEME protocol has several flavors. Here we are interested in the one that implements an authenticated DH exchange using public-key encryption as a means of authentication. The original protocol involves three messages and was formally analyzed in [5]; it can be directly used in the context of OTR messaging to provide all the desirable security properties discussed earlier, including PFS and deniability[2].

[2]The option of using SKEME is suggested at the end of the OTR paper [3] as an alternative AKE which would avoid the deniability issues created by the use of digital signatures. As we see here, the use of SKEME also addresses the other more significant security problems present in the OTR AKE.

Here, we describe a 2-message variant of SKEME in case that one wants to preserve the number of messages in the original OTR AKE. As we will see this provides with the same array of features of a 3-message SKEME but with a limited form of PFS.

$$\begin{array}{ll} A \rightarrow B: & \text{“}A\text{”}, g^x,\ Enc_B(n_A) \\ A \leftarrow B: & \text{“}B\text{”}, g^y,\ Enc_A(n_B) \end{array}$$

In this description “A” and “B” stand for the identities of A and B, respectively. The symbol Enc_A (resp. Enc_B) means encryption under the public key of A (resp. B). The values n_A, n_B are random nonces chosen by A, B, respectively. The session key is computed as: $K = PRF_{n_A}(g^{xy}) \oplus PRF_{n_B}(g^{xy})$ (PRF is a pseudorandom function such as HMAC or AES). This differs from the original 3-message SKEME in that the MAC values exchanged as part of the explicit authentication of the original protocol are now removed. Instead, the exchange is only *implicitly* authenticated through the session key computation that uses a prf keyed via the secret nonces.

This protocol can be shown to be secure in the key exchange security model of [5] *except* that the protocol does not provide explicit key confirmation (which, as shown in [5], is not a necessary property for guaranteeing secure communications) and it offers a weaker form of PFS. Indeed, as shown in [19], no 2-message protocol with implicit authentication can provide full perfect forward secrecy. Instead, what the protocol guarantees is the following weaker property: if during the key exchange execution the attacker did not inject its own messages into the communication then full PFS is guaranteed for the resultant session key. On the other hand, if, for example, the message from Bob to Alice was actually chosen by Eve, then Eve may learn the session key if she later finds Alice's private key. What is important is that as long as Eve does not learn this private key then she gets no information on the session key. If this weakening of PFS is to be prevented then one has to go back to the original 3-message SKEME protocol [17] (which can also be run as a 2-message protocol where the third message is piggy-backed to the first message from Alice to Bob in the session).

Note that in the first message of the protocol Alice uses Bob's public key, thus it is assumed that Alice has acquired Bob's public key before the initiation of the session. This is the typical case assumed in [3]; in other cases Alice will have to learn this key by some other means. Note, that Bob does not need to know Alice's public key in advance since she can include her public key in the first message (of course, this should be complemented by some way for Bob to check for the authenticity of Alice's public key, e.g. via a certificate).

SKEME supports deniability by completely avoiding the use of signatures and their possible non-repudiation implications. Instead, the authentication happens through the ability of the parties to decrypt the nonce sent by the peer.

4.3 HMQV

The MQV protocol is a well-known protocol developed by Law, Menezes, Qu, Solinas and Vanstone [20]. It is the most efficient and versatile of all known authenticated DH protocols (and consequently widely standardized). Unfortunately, the MQV protocol suffers of several design weaknesses. Recently, [19] presents a fix to the protocol which preserves its original outstanding performance while at the same time delivering its “promised security” in a formally provable way. The protocol is relevant to the applications in this paper due to its simplicity, performance and provable security including deniability (the latter property has been recently formalized and proven in [9]).

The HMQV protocol has a 2- and 3-message variants that are relevant to our context. Below we describe the 2-message variant that, as in the case of 2-message SKEME, provides limited PFS. To move to the 3-message protocol (which adds explicit key confirmation via MAC values) one either needs the additional message in the AKE protocol or to piggy-back this third message to the first session message sent from Alice to Bob.

In the HMQV setting each party has a static DH value as its public key; for example, Alice will have as its public key a value $\bar{A} = g^a$ where g generates a group of prime order q and a is a secret value in $\{1, \ldots, q\}$ known only to Alice. Similarly, Bob's public key will be $\bar{B} = g^b$ for a secret value b. The 2-message HMQV exchange consists of a simple (unauthenticated) DH exchange, namely,

$$\begin{array}{ll} A \rightarrow B: & \text{“}A\text{”},\ X = g^x \\ A \leftarrow B: & \text{“}B\text{”},\ Y = g^y \end{array}$$

The authentication is provided via the session key computation which is computed by A as $K = H((Y\bar{B}^e)^{x+da})$ and by B as $K = H((X\bar{A}^d)^{y+eb})$, where $d = h(X, B)$ and $e = h(Y, A)$. The function $h(\cdot)$ is a hash function that outputs $|q|/2$ bits and $H(\cdot)$ is a hash function that outputs values of the length of desired key (in the OTR case, 128 bits). Both functions can be computed using the same underlying hash function with suitable truncation. The key computation only costs 1.5 exponentiations (since exponentiation to the power of e or d involves only $|q|/2$ multiplications). An additional exponentiation is needed to compute the ephemeral DH value g^x, g^y.[3]

Note how the lack of any authentication information transmitted in the protocol makes HMQV well suited for deniable applications such as OTR (see next section).

5. PRIVACY CONSIDERATIONS

This section discusses some of the important privacy considerations related to the choice of protocols suggested in the previous section. We first consider the deniability properties of these protocols, and then discuss identity protection.

5.1 Deniability Properties

As discussed in the introduction and Section 3.3, an important privacy property that we would like our protocols to provide is “deniability”, namely, the property that no information generated by the protocol (during the KE part and subsequent data exchange) could be used to later prove to a third party (which we call a judge) that the communication, or its specific contents, took place. Deniability, however, is not an absolute term. It comes in different flavors and strengths depending on the assumptions that one makes about the behavior of the participants in the protocol and

[3]The exponentiation $X = g^x$ (and $Y = g^y$) can be performed off-line. In particular, in HMQV the eventual exposure of an ephemeral value x (resp. y) does not compromise the security of other sessions provided that A checks that $Y\bar{B}^e$ (and B that $X\bar{A}^e$) is of prime order q. Interestingly, this check also helps in achieving deniability.

even on the behavior of the judge. It also depends on the type of information that the protocol allows to deny. In the best case, nothing can be proven to a third party: neither the contents of a conversation nor the mere existence of the communication (i.e., the fact that party A talked to party B). In other cases it may suffice that the protocol leaves no provable trail for the contents of a conversation though it does allow to prove that a communication between A and B took place.

Dealing with the whole range of deniability flavors is beyond the scope of this paper; a detailed and formal treatment of the subject is presented in [9]. Here we will discuss informally (and consequently somewhat inaccurately) some of the deniability properties of the key exchange protocols described in the previous section.

The three protocols have the most basic and important deniability property, namely, if both peers are honest at the time of the run of the KE protocol then they leave no provable trail of the communication. To see the significance of this deniability property consider the protocol discussed at the end of Section 3.1 in which parties sign the peer's identity. Clearly, this protocol leaves an undeniable (signed) proof that B talked to A (and viceversa); moreover, not only can A prove to a third party that B talked to her, but this proof is available to any eavesdropper (at least for the non-encrypted portion of the communication). In contrast, in the protocols we propose as long as A follows the protocol then neither she or an eavesdropper will be able to prove later the contents or existence of the communication to a judge.

What happens, however, if one of the peers to the exchange (say A) purposely deviates from the honest run of the protocol in order to later be able to prove something about the communication to a judge J. The first thing to note is that in most cases such a proof of communication can be provided from A to J if A and J actively collaborate at the time the communication is happening, in particular during the run of the KE protocol. In this case J can dictate some protocol values to A that will convince J that she is receiving information authenticated by Bob. (This is easy to achieve with the SIGMA and SKEME protocols[4] but not necessarily with the HMQV protocol where convincing J may require that A reveals her private key to J). Notice, however, that even in this case J may not be able to convince another party about what he witnessed.[5]

The more interesting question is to what extent can a party (say A) deviate from the honest run of the protocol in a way that it will *later* allow A to prove something about the communication to a judge J with whom A was not collaborating during the run of the protocol. We prove in [9] that, under suitable cryptographic assumptions, protocols SKEME and HMQV provide deniability in this case.

The case of SIGMA is more complex due to the signing of the peer's DH value. Consider the 3-message protocol (SIGMA-I). The responder B signs the pair (g^x, g^y) before even knowing who he is talking to. Therefore, the specific value of g^x (chosen by the initiator and signed by B) will say nothing about who B communicated with. It could have been sent by anyone in the network, in particular by someone with whom B is not willing to talk to at all. However, in the case of A (the initiator) the signature on the pair (g^y, g^x) is generated by A after she checked who the sender of g^y is. This can allow B to "frame" A by choosing a special value of g^y (e.g. $y = SIG_B$("I talked to A on [date]")). Note that this is not sufficient proof that A really talked to B since g^y could have been sent to A by some other party C (who A knows and who received y from B), or the pair (g^y, g^x) could have been generated by A acting as a responder in another run of the protocol (in which case A signed g^y without verifying who the sender was). Yet, in some attack scenarios a carefully-chosen value of Y may be used as some form of evidence against A (e.g., a value y computed as the signature of B on the pair $(``A", g^x)$ may be used to prove that A acted as the initiator). In addition, a large scale attack directed to frame many parties (say, a large web site – e.g., one whose visitors/customers may prefer not to be identified – performing this attack on all its customers) may also be difficult to deny.

We will omit a more detailed discussion here and will only note that SIGMA-R is more robust to the above attacks: since the first to sign in SIGMA-R is the initiator A. Hence, any choice of g^y by a dishonest responder B (even if g^y depends on g^x) has no consequence in framing A (who signs before even knowing, or verifying, who she is talking to). Hence any signature of party A on a pair (g^y, g^x) can always be claimed to have been produced as initiator and hence it carries no proof of whom the peer to the exchange was (it is important that signatures produced by the parties as initiators and responders are indistinguishable). This property of SIGMA-R may be particularly relevant to scenarios such as instant messaging where end points are individuals acting both as initiators and responders of key exchange executions.

5.2 Identity Protection

As clearly stated in the conclusion of the OTR paper [3], anonymity or *identity protection*, was not a design choice for OTR. However, we believe that a protocol whose goal is to achieve a high level of privacy, as is the case of OTR, should strive to provide identity protection to the largest possible degree. Specifically, by identity protection (see [17, 18, 1]) we refer to the hiding of (logical) identities from attackers in the network (not from the peers to the exchange). Such logical identities may be names of individuals, companies, an email address, a public key or its certificate, etc. In contrast, physical addresses, as those required for routing, will inevitably be exposed; but in many cases these routing addresses will not be bound to logical identities in which case protecting the latter is of significance. Indeed, hiding who am I talking to may not be less significant than hiding the content of a conversation (especially considering that many times it is the information of who I am talking to that will trigger the interest in eavesdropping into my conversation, or just be the reason to throw me to jail).

[4]For example, in the SKEME protocol J will encrypt a nonce n_A that only she knows under B's public key, and give the resultant ciphertext to A who sends it to B. If B is willing to talk to A then he will send an encrypted nonce n_B to A, and will compute a session key derived from n_A, n_B and g^{xy}. Assuming that J also chose x and that A can prove that the ciphertext received from B contained the value n_B (this is possible for "committing encryption" schemes such as RSA) then J will be convinced.

[5]Needless to say, we limit ourselves to "algorithmic" proofs of communication, and ignore other means that courts may accept as evidence such as physical tapping (or just the word of a gentleman...).

Some key-exchange protocols such as the one originally used in OTR do not provide identity protection. However, both the SIGMA and SKEME protocols can provide such protection while enjoying all other desired security features, including deniability. For achieving identity protection in the case of SIGMA the parties use the values g^x, g^y to compute a key with which they encrypt the identities (including public keys and certificates if sent) and the signatures. In this case, SIGMA provides protection of identities for both parties against eavesdroppers. In addition, SIGMA-R provides protection of the responder's identity against active attacks while SIGMA-I does it for the initiator (see [18]). SKEME can also provide identity protection by simply including the identities under the exchanged ciphertexts (see [17]).

In the case of HMQV, the protocol does not support identity protection since each party needs to know the peer's PK before it can compute the session key. If one is willing to pay with an extra exponentiation and extra message, then one could use the 3-message variant of HMQV while encrypting the second and third flow with a key *carefully* derived from g^{xy} (computing g^{xy} is where the extra exponentiation is needed). We omit any discussion and details of such key derivation.

One possible objection to the use of identity-hiding protocols is that in IM identities are anyhow transmitted in the clear as part of the addressing system. While this is the case for many such protocols, it is not an essential (or unavoidable) need. One could envision protocols that will use session identifiers, or other forms of temporary "aliases", to conceal such identities (at least from network attackers if not from the IM server). If such measures are implemented, one must be careful not to have the key-exchange protocol itself reveal the (logical) identities. In this case, identity-protecting key-exchange protocols become very desirable.

6. FURTHER COMMENTS ON OTR

This section discusses some further technical aspects of the design of OTR [3] and points to some weaknesses that should be removed from the protocol.

6.1 Repudiability of the symmetric encryption

Two central design choices in OTR are the use of a malleable encryption scheme (such as a stream cipher) to encrypt messages, and the release of MAC keys after each key refresh (see Section 2.3). Both steps are introduced as a means to enhance the repudiability of messages by the participants. It is not clear, however, how much is gained by these mechanisms. The basic idea is that if the attacker can produce a ciphertext and its authentication by itself then it cannot convince a third party that certain information was transmitted between Alice and Bob. Note, however, that if the key-exchange protocol in use is deniable then there is no way to tie a communication to the value of a key. So even if an attacker Eve claims to have learned a key used between Alice and Bob (and even if we assume that it really learned the actual session key), there is no way that Eve can convince that this is the real value of the session key. Eve could have as likely made up the key and generated the alleged communication.

One could say that if the measures of malleable encryption and disclosure of MAC keys do not help they do not cause harm either. We disagree. First, stream cipher encryption has some risks that make them less appealing in some scenarios, especially in the need of managing counters with great care to avoid re-use of counter values or other portions of a pseudorandom stream. Second, it is best to leave freedom for the choice of encryption functions and modes as available and desired by an application. Third, revealing the MAC keys does introduce timing and synchronization issues needed to prevent a too-early disclosure. While this is possible as claimed in [3] this results in added complexity to the system.

While the above considerations may be seen as subjective to some extent, in the next subsection we illustrate the danger of adding non-standard security techniques. If not carefully designed, such techniques (while appealing to the eye) may interact badly with other components of the system.

6.2 Unnecessary weakening of encryption keys

The choice to reveal MAC keys in OTR interacts badly with another design element (also motivated by deniability issues) in a way that weakens the secrecy of encryption keys used by the protocol, and consequently weakens the secrecy of the encrypted information.

In OTR the MAC keys are computed as a one-way hash of the encryption key: $K_{mac} = H(K_{enc})$, where the encryption key is obtained by hashing the result of the DH key exchange. Once again the justification for this peculiar design choice is repudiability: should the attacker find the decryption key, she would not only be able to read messages but also to MAC them. As observed in Section 6.1, the justification for such steps is dubious, One objection is that once K_{mac} is revealed the attacker has an easy way to test for the value of any encryption key (without the need for known or plausible plaintext). In the extreme case that this key is used as a one-time pad, the attacker could use this knowledge to mount a "dictionary attack" on the plaintext which is infeasible otherwise.

6.3 On the Key Refreshing

As recalled in Section 2.2, the OTR protocol performs a DH key exchange *per message* and that such exchange is authenticated with the previous key. The goal is to obtain a fine-grain perfect forward secrecy mechanism in which learning the encryption key at one point will not allow the adversary to learn even a *single* past message. We note however, that if the adversary learns the current ephemeral key[6], future messages may be completely compromised. Indeed even if the new encryption key is not computable from the old one (being the result of a fresh DH exchange), the adversary can however impersonate the parties because the old key is used for authentication. In particular the attacker can hijack the session and learn/modify all future messages.

More in detail, if in the OTR protocol, Eve finds the key K_{i-1} used for the last message $i-1$, then she can run the next i^{th} message DH key exchange with Alice pretending to be Bob, since she knows how to authenticate. Similarly, she can perform the i^{th} message DH key exchange with Bob pretending to be Alice. Now all the messages exchanged by Alice and Bob go through Eve, who can of course learn them and also modify them at will.

[6]This is plausible if the user's computer has been compromised in some way.

Thus the value of performing a DH key exchange with each message where authentication depends on the previous shared key is of limited value. This is even more so given the computational cost of a DH exchange ([3] discounts the effect of such computation, however it may be considerable in low-powered devices). In any case, whether the computational cost of a DH exchange is substantial or negligible, we believe that the better use of this power is in running periodic, fully fresh, DH exchanges authenticated using the long-term private authentication keys of the parties. In this case, as long as a party's private key is not disclosed to the attacker, the party can be confident of retaining (or regaining) full control of its sessions with each such full-fledge DH exchange.

Thus we suggest that OTR will enjoy better overall security by running the AKE protocol at regular intervals. If a finer-grain refreshing mechanism is desired for forward-secrecy purposes, then a lighter, yet powerful, mechanism can be employed, such as deriving new keys (possibly on a per-message basis, if so desired) by one-way hashing the previous key. This ensures that a break at a given point in time will affect future communications but not past ones. Most importantly, the affected future communications are only those carried before the next full-fledge run of the AKE. Once this protocol is performed, the attacker looses its ability to read and modify the communication!

7. REFERENCES

[1] M. Abadi, Private Authentication, *Proc. of the 2002 Workshop on Privacy Enhancing Technologies (PET 2002)*, Springer-Verlag, pp. 27–40, 2003.

[2] M. Bellare, R. Canetti and H. Krawczyk, A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols, *Proc. of 30th Symposium on Theory of Computing (STOC)*, ACM, pp. 419–428, 1998.

[3] N. Borisov, I. Goldberg and E. Brewer, Off-the-Record Communication, or, Why Not To Use PGP, *Proc. of the 2004 ACM Workshop on Privacy in the Electronic Society*, ACM Press, pp. 77–84, October 2004.

[4] J. Callas, L. Donnerhacke, H. Finney and R. Thayer, OpenPGP message format, RFC2440, November 1998.

[5] R. Canetti and H. Krawczyk, Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels, *Advances in Cryptology – Proc. of EUROCRYPT '01, LNCS 2045*, Springer-Verlag, pp. 453–474, 2001.

[6] R. Canetti and H. Krawczyk, Security Analysis of IKE's Signature-based Key-Exchange Protocol, *Advances in Cryptology – Proc. of CRYPTO '02, LNCS 2442*, Springer-Verlag, pp. 143–161, 2002.

[7] W. Diffie and M.E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. 22, n. 6, pp. 644–654, 1976.

[8] W. Diffie, P. Van Oorschot and M. Wiener, Authentication and Authenticate Key Exchange, *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.

[9] M. Di Raimondo, R. Gennaro and H. Krawczyk, Deniable authentication and Plaintext Awareness, *Manuscript*, August 2005.

[10] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk and T. Rabin, Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes, *Advances in Cryptology – Proc. of CRYPTO '04, LNCS 3152*, Springer-Verlag, pp. 399–424, 2004.

[11] C. Dwork, M. Naor and A. Sahai, Concurrent Zero-Knowledge, *Proc. of* 30th *Symposium on Theory of Computing (STOC)*, ACM, pp. 409–418, 1998.

[12] C.G. Guenther, An identity-based key-exchange protocol, *Advances in Cryptology – Proc. of EUROCRYPT '89, LNCS 434*, Springer-Verlag, pp. 29–37, 1989.

[13] D. Harkins and D. Carrel, ed., The Internet Key Exchange (IKE), *RFC 2409*, Nov. 1998.

[14] J. Katz, Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications, *Advances in Cryptology – Proc. of EUROCRYPT '03, LNCS 2656*, Springer-Verlag, pp. 211–228, 2003.

[15] C. Kaufman, ed., Internet Key Exchange (IKEv2) Protocol, draft-ietf-ipsec-ikev2-17.txt, September 2004 (pending RFC).

[16] H. Krawczyk, M. Bellare and R. Canetti, HMAC: Keyed-hashing for message authentication, RFC2104, February 1997.

[17] H. Krawczyk. SKEME: a versatile secure key exchange mechanism for Internet. *1996 IEEE Symposium on Network and Distributed System Security (SNDSS '96).*

[18] H. Krawczyk, SIGMA: The 'SiGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols, *Advances in Cryptology – Proc. of CRYPTO '03, LNCS 2729*, Springer-Verlag, pp. 400–425, 2003. Available at `http://www.ee.technion.ac.il/~hugo/sigma.html`

[19] H. Krawczyk, HMQV: A High-Performance Secure Diffie-Hellman Protocol, *Advances in Cryptology – Proc. of CRYPTO '05, LNCS*, Springer-Verlag, 2005. Available at `http://eprint.iacr.org/2005/176`.

[20] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, An efficient Protocol for Authenticated Key Agreement, *Designs, Codes and Cryptography, 28, 119-134, 2003.*

[21] W. Mao and K.G. Paterson, On the plausible deniability feature of Internet protocols, *Manuscript.*

[22] A. Menezes, P. Van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[23] R. Pass, On Deniability in the Common Reference String and Random Oracle Model, *Advances in Cryptology – Proc. of CRYPTO '03, LNCS 2729*, Springer-Verlag, pp. 316–337, 2003.

[24] Adrian Perrig, Ran Canetti, J.D. Tygar and Dawn Song, Efficient Authentication and Signing of Multicast Streams over Lossy Channels, *2000 IEEE Symposium on Security and Privacy.*

[25] Editor B. Ramsdell, S/MIME version 3 message specification, RFC2633, June 1999.

[26] T. Ylonen, SSH secure login connections over the Internet, 6th *USENIX Security Symposium*, pp. 37–42, San Jose, CA, July 1996.

[27] P. Zimmerman, The Official PGP User's Guide, MIT Press, 1995.