OFF-THE-RECORD SECURE CHAT ROOM

Jiang Bian, Remzi Seker, Umit Topaloglu and Coskun Bayrak
Department of Computer Science, University of Arkansas at Little Rock
2801 S. University Avenue, Little Rock, Arkansas, U.S.A.

Keywords: Off-the-Record, Chat room, Instant Messaging, Security, Group Diffie-Hellman.

Abstract: Group Off-the-Record (GOTR) (Bian et al., 2007) was proposed to address the privacy protection concerns in online chat room systems. It extended the original two-party OTR protocol to support more users while preserving the same security properties. A literature survey of different Diffie-Hellman (D-H) conference key implementations will be given to justify that in an application like a chat room, the virtual server approach is truly the most efficient way to establish a private communication environment among a group of people. However, GOTR's virtual server approach raises a trustworthiness concern of the chosen chair member. Since the chair member has full control over all encryption keys, there is no constraint to prevent him / her from altering the messages while relaying them. In this paper, we present a study of the GOTR protocol and a solution to the virtual server's trustworthiness problem via employing an additional MD5 integrity check mechanism. Having such an algorithm, makes the GOTR protocol more secure, in that, it gives the other chat members an opportunity to be aware of any potential changes made by the chair member.

1 INTRODUCTION

Chat room technologies started to surface in the 1970s. Initially, they was used in Unix systems to help all the users who have logged on the same machine to communicate with each other. Then, it quickly evolved into a network based system and became much more popular with the growth of the modern Internet. The contemporary online chat room provides a real-time, text message based communication mechanism over instant messaging (IM). It is fast, flexible, expandable and is virtually cost free to users. Users tend to prefer chat rooms to other conventional communication tools like the telephone systems and electrical emails, since it is less intrusive but more interactive.

However, privacy concerns hinder the expansion of the use of chat rooms as well as Instant Messaging. Although, many studies have been conducted to provide a secure IM system, however, few of them achieve perfect secrecy, which is in high demand by the public.

2004, Borisov (Borisov et al., 2004) et. al. proposed a secure off-the-record IM system, OTR, which has addressed most of the existing security issues. It achieved perfect forward secrecy via short-lived encryption keys, which ensures that the revealing of a ephemeral key would not compromise the entire chat session. Moreover, the OTR protocol accommodates a degree of plausible deniability, so that a user could later discredit what he / she has said.

GOTR designed by Bian et. al. (Bian et al., 2007), is an extension of the two-pary OTR protocol, which makes it possible for multiple users to talk securely and off-the-record in an online chat room environment. The GOTR protocol introduced a idea of choosing a chair member to act as a virtual server and to be responsible for relaying all messages securely. The communications between the virtual server and every other user are guarded by the original OTR protocol. In this way, a secure communication chain is created.

However, the trustworthiness of the chosen virtual server remains a flaw. Since the virtual server supervises all the communications in the chat room, it is fairly easy for him / her to make changes while repackaging messages. In order to prevent the messages from being altered by the virtual server, we suggest to employ an MD5 integrity check mechanism. Having a such mechanism will give other users an opportunity to verify that the messages truly have not been corrupted (i.e. the message could have been changed by the virtual server, or due to a network failure.) during its transmission.

This paper consists of five sections. In Section 2, we present the key features of the original OTR protocol. In Section 3, we conduct a literature survey to show the general idea of conference key distribution systems, and compare their efficiency and complexity to justify the advantages of using the virtual server approach. A case study of the GOTR protocol will be presented in Section 4.2 along with the idea of employing an MD5 integrity check to prevent the trustworthiness failure of the chair member. Moreover, a automated mechanism will be suggested to deal with the membership alternation problem caused by network failures. Conclusions will be given in Section 5, followed by future work in Section 6.

2 OFF-THE-RECORD INSTANT MESSAGING

Many studies have been conducted to secure IM systems (Gaim-e, 2002) (Pidgin-Encryption, 2007) (©Secway, 2006). Most of them simply add extra authentication and encryption layers above the public IM protocols to ensure the authenticity and confidentiality. However, perfect forward secrecy and deniability are often omitted by these secure IM solutions. Nikita Borisov et. al. introduced the Off-the-Record Messaging (OTR) system (Borisov et al., 2004) at the Workshop on Privacy in the Electronic Society, 2004. The OTR protocol is a comprehensive security solution for public IM systems and it implements strong cryptographic protection. Furthermore, two key features of the OTR protocol are off-the-record communication and perfect forward secrecy protection.

2.1 The OTR Protocol

In an off-the-record chat room, a sender has the ability to deny what he/she has said in a previous chat session. The conversations are not bound to any participant's identity. However, the messages in an ongoing conversation are still properly authenticated through using Digital Signatures and Message Authentication Codes (MAC).

There are four key security properties integrated in the OTR protocol. These properties are briefly described next.

Perfect Forward Secrecy. This has been archived through the use of short-lived encryption/decryption key. After the completion of transmitting one message, a new shared secret will be generated and used to encrypt/decrypt the next message. And the current key will be simply discarded by both sender and receiver. As a result, the revealing of current shared key will not compromise the secrecy of previous messages. Even if an attacker manages to capture the current encryption/decryption key, it will soon be invalid for the next message.

Digital Signature. The authentication issue is addressed via a digital signature system. Each user's digital signature in OTR acts as long-lived keys. The keys are solely for authentication purposes and are not used to encrypt the real messages.

MAC Code and Plausible Deniability. The MAC (Stinson, 2002) code is introduced to retain the deniability of an off-the-record conversation. Unlike digital signatures, MACs are generated and verified via the same secret key. Therefore, a MAC value does not provide the non-repudiation property offered by a digital signature.

Malleable Encryption and Forge-ability. In general, malleability is an undesirable property in encryption algorithms. However, for the OTR protocol, it is an enhancement feature that enables anyone to transform an encrypted message (m) into another valid cipher text ($m_{\prime}$) without the knowledge of the plaintext. Such Homomorphic Encryption scheme (Canetti et al., 1996) provides an intensive degree of forge-ability.

2.2 Security Weakness in the OTR Protocol

Mario Di Raimondo, et al. (Raimondo et al., 2005) pointed out three major security flaws or vulnerabilities in the OTR protocol.

First of all, OTR inherits a possible "identity misbinding" attack from the D-H protocol. Suppose that a malicious user, Eve, stands between two end-users, Alice and Bob, and she has the ability to intercept all the communications between them. If she attacks properly (Diffie and Hellman, 1976), Eve could manage to make Alice think that she is talking to Bob, but actually she is speaking to Eve. For a real life example, Eve can use this authentication flaw to mislead a customer, Alice, and a bank, Bob. One simple solution is to include identity information in the digital signature, but it will surely dismiss the deniability property.

Moreover, the revealing of an ephemeral private key could cause an impersonation attack. An attacker could use this piece of information to produce a valid session key as long as the long-term keys are not re-invoked. This attack could be defended by doing full key refreshment periodically, which ensures that the revealing of an ephemeral private key of one conversation session will not affect the next fully refreshed one.

Furthermore, the improper mechanism of revealing MAC keys weakens the secrecy of encryption keys. Since the MAC keys are generated as a one-way hash over the encryption key, the attacker can use this knowledge to mount a ”dictionary attack”, although it is probably computationally too expensive.

Consequently, Mario Di Raimondo, et al. suggested three alternate Authentication Key Exchange (AKE) algorithms, SIGMA, SKEME (Krawczyk, 1996) (i.e. which is an early voice of a protocol designed to provide deniability to IPsec’s IKE protocol.) and HMQV, and discussed both the advantage and disadvantage of using these three protocols.

This discussion resulted in a second version of the OTR protocol (Borisov et al., 2004), where:

1. fixed the identity-binding flaw (the impersonate attack vulnerability) simply by adding an additional identification message at the beginning of the conversation session.
2. No longer revealing the users’ public keys to passive eavesdroppers and this helps in privacy preserving for the internal application’s OTR messages

And, additionally, they provide a support of fragmentation OTR messages, since a lot of Instant Messaging protocols have limitation on the size of each message.

Despite the security advantages it offers, the OTR project has a missing key feature, a support of chat room conversations.

3 GROUP DIFFIE-HELLMAN AND THE CONFERENCE KEY SYSTEM

In order to create a secure communication environment over insecure channels (public Internet infrastructure), it essentially falls into a problem that how to distribute the encryption/decryption keys securely. The OTR protocol uses the most familiar Diffie-Hellman (D-H) (Diffie and Hellman, 1976) key agreement for this purpose. The D-H protocol allows two users to establish a shared secret over an unprotected channel. It is believed that the most efficient way to break the D-H protocol is to solve the underlined mathematic problem, the discrete logarithm problem. However, there is no computationally feasible approach has been found.

When a group of people demand to talk privately over an unsecure environment, a conference key distribution system (CKDS) is required. Many papers (Ingemarsson et al., 1982) (Burmester and Desmedt, 1994) (Bellare and Rogaway, 1995) (Steiner et al., 1996) (Bresson et al., 2001) (Bresson et al., 2007) have been presented to solve such a conference key distribution problem. And our goal is to find an efficient group D-H conference key system that is suitable to be used in securing chat room systems, while preserving off-the-record and perfect forward secrecy.

Basically, as a group D-H system, it has to collect contributions from each communication member (g^{r_i}), compute the conference key (centralized or independently), and distribute the common secret to each member securely.

Ingemarsson et. al. proposed a conference key system where n members needs to be arranged in a logical ring (Ingemarsson et al., 1982). During one key establishing round, one user U_i raises the intermediate key value received previously from U_{i-1}, and passes the result to the next participant, U_{i+1}. In $(n-1)$ rounds, all members will compute the same

$$K = g^{r_1r_2+r_1r_3+...+r_1r_n+r_2r_3+r_2r_4+...+r_2r_n+...+r_{n-1}r_n} \; mod\; p$$

Burmester and Desmedt (Burmester and Desmedt, 1994) proposed a cyclic based conference key distribution system and claimed that their protocol was much more efficient. This protocol runs $2n+1$ rounds, and eventually, all users will agree on a conference key

$$K \equiv g^{r_1r_2+r_2r_3+...+r_nr_1} \; mod\; p$$

Michael et. al. (Steiner et al., 1996) introduced a ”natural” extension to the two-party D-H protocol. Its key features are summarized as follows: 1). $n+1$ rounds of message exchanging; 2). constant message sizes; 3). small number of exponentiations required. Also, they have provided a solution to the problem of member addition and deletion, which ensures that the alteration of group membership will not compromise the group’s secrets.

However, in a chat room system, every message will be multicast to each user automatically by the underlined chat room protocol. So, it is hardly possible to create such a logical ring without heavy overheads. Even if a user only meant to send a intermediate key to U_i, the chat room system will automatically send a copy to every other user as well. This is a waste of effort, and since this value has no meaning to the rest of the group, each user will simply discard it. Moreover, in all three aforementioned CKDSs, a user U_i needs to raise a previously received intermediate key to the power of its own private key, r_i, and pass it to the next member or broadcast to everyone else. Exponentiation is computationally expensive in a computer system. Even in Michael et. al.'s system, which has the minimal number of exponentiations of all three protocols, still requires $5n-6$ exponentiations. Moreover, recall that in order to achieve the perfect forward secrecy, a short-lived key scheme is used in the OTR protocol. If we simply applied one of these CKDSs, it would create a tremendous amount of overhead. It would require the exchange of at least $2n-1$ messages and take $n+1$ rounds to compute a common secret, when only encrypting one small text message. Obviously, as the number of chat member increases, the bandwidth used to establish the shared channel will be hundreds of times bigger than the real message payloads.

4 GROUP OFF-THE-RECORD IM PROTOCOL

Group Off-The-Record (Bian et al., 2007) is an extension of the original bipartite OTR protocol. In the GOTR protocol, the member who initiates the GOTR chat room will be chosen to act as a virtual server. Then, the virtual server starts a two-party OTR key exchange to establish a private, off-the-record channel with each other member pair-wise. Since two chat members do not share a common secret directly, the virtual server is responsible for routing all messages from one user to another. When the virtual server receives a message, he / she will need to index through his / her key table to find the proper decryption key, decrypt the message and re-encrypt it with the key he / she shares with the designated receiver. All communications between a user and the virtual server are guarded by the two-party OTR protocol. In such a design, a secure communication chain, from one user to another, has been created.

The GOTR's virtual server approach, avoids the trap of dealing with the complexity of generating a conference key among many users, and instead uses a router based concept: a trusted member is chosen as a router to relay all messages, and is responsible for managing all the encryption/decryption keys. As we can see, the GOTR protocol still creates overheads such as sending encrypted messages to a user who would not be able to read it, however, it reduces the total number of exponentiations used in computing the shared keys. According to the D-H protocol, for each new key, all the users except the virtual server needs to do the exponentiation twice (i.e. once to compute its public key, g^{r_i}, the other one is used to compute the shared key between he / she and the virtual server.), while only the virtual server needs to do $2(n-1)$ exponentiations. However, the total number is still much less than any existing conference key distribution systems discussed above.

4.1 A Case Study of the GOTR Protocol

Let us suppose there are three users, Alice, Bob and Carol, who want to have an off-the-record chat room conference in a secure manner. This scenario can be seen in Figure 1:

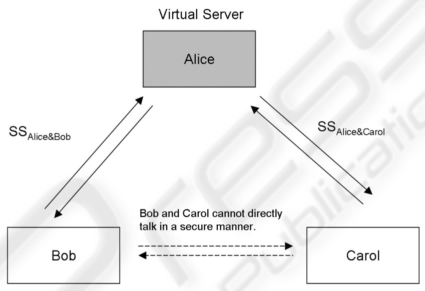

Figure 1: GOTR communication scheme.

Alice and Bob as well as Alice and Carol would establish the two-party OTR communication channel (i.e. in other words, Alice and Bob share a common secret $SS_{Alice\&Bob}$ ($SS_{Alice\&Carol}$ for Alice and Carol) and follow the OTR protocol for further conversations. Therefore, the channel between them is secure and off the record.). Since Bob and Carol do not have shared key, they cannot talk to each other directly. They have to go through the virtual server, Alice. Bob first sends Alice the message encrypted with key $SS_{Alice\&Bob}$; Alice deciphers it using $SS_{Alice\&Bob}$ and repacks it with key $SS_{Alice\&Carol}$; then she relays the new message to Carol. Now Carol has no problem reading this message. Alice, in this configuration, acts like an interpreter.

In the proposed system, there remains a security vulnerability that needs to be addressed: there is always a chance that a virtual server can be attacked. Since the virtual server is privileged to fully control all the encryption keys, it would be relatively easy for an attacker to change the content of a message and broadcast to other chat room members. Therefore, we suggest creating a mechanism for the users to verify the integrity of the messages received by the virtual server. The other flaw, which exists in the previous GOTR product, is that in the case of a virtual server encountering a network failure, all the subsequent conversations in this chat room will be unreadable since the rest of the users are not sharing any common secrets (i.e. shared encryption keys). Therefore, in this paper, we purpose an improved version of the GOTR protocol, which solves these two problems and provides some other enhancements to the GOTR Pidgin plug-in.

Moreover, a network failure of the virtual server raises another issue. Since the users do not have a shared key other than with the virtual server, if the virtual server drops off, the communication chain cannot be established. In this paper, we propose a solution to the virtual server's honesty problem using MD5 hash functions and provide a mechanism to deal with the problem of virtual server network failure.

4.2 GOTR Enhancement: MD5 Integrity Check and Network Failure Handling

4.2.1 MD5 Integrity Check

Even though the GOTR communication scheme seems to be a good solution for the security needs described before, it still has a security flaw. Since there exists a centralized controller (the Virtual Server), a failed or hacked controller will cause the GOTR system to fail. Therefore, the degree of our security mode is coupled with the trustworthiness or the security level of the virtual server. The previous version of GOTR does not provide an integrity check for the messages relayed over the virtual server. Ideally there should be a mechanism to warn Carol that a message has been modified during its transmission.

Consider the previous GOTR example, in which Alice, Bob, and Carol were having a GOTR chat room conference. Alice has been dedicated as the virtual server. The two way communications between Alice—Bob and Alice—Carol are guarded by the original OTR protocol. But OTR does not provide an assurance that Alice, the virtual server, did not change the content of any message during the repacking process.

Redundancy check mechanisms are well-known and often used in network applications for the purposes of error detection and correction. Redundancy checks, although slightly increases the size of each message, can provide an integrity check for the whole message. If only error detection is required, the checksum algorithm is obviously the most widely used and simplest method. It works by adding up basic components of data, typically the asserted bits, and sending the result along with the original message. Anyone who receives this message plus its checksum can later perform the same operation and compare the resulting value with the checksum. The receiver can conclude that the message has not been corrupted during its transmission. Some other redundancy check methods include parity checks, check digits, and cyclic redundancy check (CRC), etc. For instance, a CRC function often takes a stream of variable length data as input and produces a fixed size code as output. Compared to checksum, CRC provides more precise error detection while maintaining a plausible execution time. Basically, any kind of (cryptographic) hash function can be used to produce a message digest, which can be used to verify the message's integrity (W.W.Peterson and D.T.Brown, 1961).

In the proposed protocol, since Bob (Carol) encrypts messages with keys that are only known by him (her) and Alice, it is not possible for Carol (Bob) to decrypt these messages. Therefore, we need the help from the virtual server, Alice. Following our GOTR protocol, Alice decrypts the message originated from Bob and re-encrypts it with the key ($SS_{Alice\&Carol}$) shared between her and Carol. Under such a system, there is no way we can prevent Alice from altering the messages. However, we can provide a verification mechanism, which will give Carol a chance to check the integrity of messages. In order to assure the integrity of the message, Bob will produce a hash value H_b of the message and send it to Carol in a straight line. After receiving both the hash value from Bob and the repacked message from Alice, Carol can check whether the message has been altered or not. She simply just needs to reproduce the hash value upon the received message and compare the result with the one from Bob.

The aforementioned integrity check method has been implemented in the GOTR protocol. In this experimental work, a 128-bit MD5 algorithm (Rivest, 1992) has been employed, and it is possible to use any existing one-way hash function. By using a hash algorithm that has a higher collision rate, it may allow a malicious user to be able to alter the content of a message, but still produce the same message digest. MD5 takes an arbitrary number of characters as input but produces a fixed length output. This property assures a minimal amount of overhead to each message. Moreover, MD5 functions are widely available as libraries in almost all the major programming languages such as C, C++, Java, PHP, etc. The OTR library uses libgcrypt (Koch, 2005) to provide the basic cryptographic functionalities and it also has MD5 algorithm support. Therefore, no extra programming effort will be required to provide MD5 services.

The tests and mathematical proofs have shown that the probability of retrieving the plaintext from its hash value is negligible (Bishop, 2002). Hence, it is not an infringement of security requirements to trans-mit the hash values for messages through unsecured channels.

Since the proposed protocol uses existing internet infrastructure and requires time to encrypt the message, there will be latency. It is not possible to estimate the message and hash arrival order. Different arrival times of the hash and message might cause a false positive. In order to resolve the issue, a message identification (`?ID?`) is attached to each message. In the GOTR protocol, the unique ID is the UNIX (or POSIX) time when the message was created. UNIX time is the number of seconds elapsed since midnight UTC of January 1, 1970, not counting the leap seconds. Collisions rarely occur if we extend the timestamp to include microseconds.

From the application point of view, every time Bob wants to communicate, he will send out the encrypted message along with a message digest (i.e. hash value generated by the MD5 algorithm) and the unique identifier (i.e. timestamp). In chat room protocols, messages will be broadcast to everyone. Hence, Alice has no way to prevent Carol from receiving the message digest directly from Bob. Later, Carol could use this information to verify this message's integrity, when she receives the re-encrypted copy from Alice. In detail, she needs to perform the following steps: 1) decrypt the message from Alice and save the unique ID (UID); 2) calculate the MD5 value over the plaintext message; 3) look up the MD5 value associated with this UID from all the MD5s she gathered from Bob; 4) compare these two MD5s and make the conclusion. If they are the same, Alice has not changed the message originated by Bob; otherwise, Alice has corrupted it. This approach will cause more overhead network traffic. However, an automated way of checking the integrity of messages would have more benefits than drawbacks.

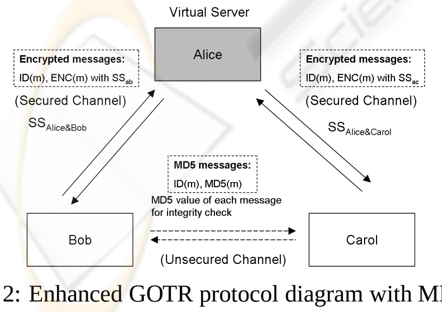

Figure 2: Enhanced GOTR protocol diagram with MD5 integrity check.

As shown in Figure 2, the communication channels between the virtual server and the other chat room members are secure and off-the-record. However, the messages exchanged between Bob and Carol, are not encrypted and vulnerable to attack. Although we cannot transfer confidential messages through this unprotected channel, we can still use it for transferring message digests. When Bob wants to talk to Carol in the GOTR chat room, he would send the encrypted contents through the private channel (i.e. which passes through the virtual server) along with a unique identifier:

Message 1: Bob → Alice, Carol:

```
?RECV?Alice@hotmail.com?ENDRECV?
+ ?SEND?Bob@hotmail.com?ENDSEND?
+ ?UID?unix_timestamp?ENDUID?
+ <Enc Msg with SSAlice&Bob>
```

Carol will discard this message since it is not readable for her. Meanwhile, Bob computes the message's MD5 digest and sends it to Carol in a straight line along with the UID too for verification.

Message 2: Bob → Carol, Alice:

```
?RECV?MD5_value?ENDRECV?
+ ?UID?unix_timestamp?ENDUID?
```

Alice, the virtual server, will repack Bob's message with a proper key and pass it on.

Message 3: Alice → Carol, Bob:

```
?RECV?Carol@hotmail.com?ENDRECV?
+ ?SEND?Bob@hotmail.com?ENDSEND?
+ ?UID?unix_timestamp?ENDUID?
+ <Enc Msg with SSAlice&Carol>
```

Eventually, Carol gets both the relayed message from Alice and the message digest from Bob, and then pairs them according to the UID. She computes the MD5 value on the message from Alice and compares it with the one she received from Bob. These two values are supposed to be the same because they are calculated upon the same message with the same predefined hash function. They will not match, only if Alice has changed the message or the network package has been corrupted. No matter the reason, Carol should not trust this message. She can instead, open a new OTR channel and verify this message with Bob. Now that the problem of assuring the message's integrity is addressed, the solution for handling the virtual server's drop-off will be discussed next.

4.2.2 Network Failure Handler

When a user is using an IM service, assume that the MSN server and the user encounter a network problem, the MSN service will be cut off and close all the active sessions it has established with the user. When a user reconnects to the MSN server at a later time, the previous chat sessions will not be restored. There are two possible scenarios regarding network failure during a GOTR chat room session:

1. One or more chat room members other than the virtual server lose their connections. In this case, the remaining users can still talk securely and off-the-record. However, when those disconnected users recover and rejoin the chat room, the subsequent conversations would not be secure anymore. Continuing with our previous example, suppose Carol had a system error and restarted her computer. After she rejoins the GOTR chat room, the connection between Alice and Bob are still secure but the one between Alice and Carol has no more protection. As a part of the exception handling system, a GOTR system should recreate the private channel between Alice and Carol automatically in order to watch over the entire chat room. In our GOTR Pidgin plug-in, upon receiving the event message that Carol has rejoined the chat room, Alice, as the virtual server, will start over the entire GOTR chat room session. Even though this is not necessary for Bob, a periodic key refreshment will make the GOTR session more secure.
2. In case of the virtual server drops from the GOTR session (e.g. the virtual server loses its network connection), all private channels established for that session will be lost. The other chat room members will be talking in an unprotected environment. In this case, the GOTR system would randomly pick a user from the existing members and make him/her the new virtual server. Then the GOTR chat session will be restarted. The conversation tunnels between the new virtual server and the rest of chat users will be private again. By doing so, a new secure and off-the-record chat session will be created. If the prior virtual server reconnects and rejoins the chat room, the GOTR system will treat him/her as a new joining member and react the same way as in the first scenario

4.3 Performance Evaluation of GOTR Protocol

One question would be how many users a GOTR chat room could hold. Although there is no practical limit on the number of users who might join a GOTR session, more users will cause more overhead and network traffic, which eventually will cause noticeable delays. In addition, since the GOTR protocol uses a virtual server to route all messages, increasing number of users will increase the burden on the virtual server. Hence, with increased number of users, problems may arise not only because of the limitation of the network bandwidth but also the computational resources of the virtual server. We have not performed any formal performance test such as comparing transit time of each message, counting how many extra packets are caused by the GOTR protocol etc. We have tested our Pidgin GOTR plug-in with up to ten users in a single chat room and there was no noticeable delay. The number of users in a GOTR chat room session is limited by the number of users allowed in an MSN chat room session. However, having extra network bandwidth and computational power will provide faster off-the-record chat room service.

5 CONCLUSIONS

An off-the-record secure chat room is a much-needed product for the public and the business community. Although the GOTR protocol has improved the original two-party OTR protocol, it was far from perfect. Therefore, through our continued work that is presented in this paper, the virtual servers trustworthiness problem was addressed by employing a MD5 integrity check mechanism. This scheme provides the remaining chat members the ability to identify any message alterations if ever made by the virtual server. Although, the MD5 integrity checking mechanism would increase the overhead of each message, with the initial tests, the network delays are not noticeable. Furthermore, a few enhancements have been suggested to deal with network failures. No matter who is disconnected from a GOTR chat room, a full key refreshing process is required to secure the whole communication chain. Moreover, as discussed, periodically refreshing the private channels of the entire chat room session makes the GOTR protocol more secure.

6 FUTURE WORK

The protection of chat history files would be a nice feature to include. Currently, there is no way to prevent a user from saving the chat histories. Efficient ways of attaching list of permissions to each chat history file would be beneficial. The mechanism could be implemented such that every member in a GOTR chat room would have a synchronized chat history file with a unified access control list attached. Also, the identity-pertaining information associated with each message saved in the chat history should be sanitized, so it will not compromise the off-the-record property of the GOTR protocol.

ACKNOWLEDGEMENTS

This work was funded, in part, by grants from the National Science Foundation #0619069 and #0701890 and Department of Defense #H98230-07-C-0403.

REFERENCES

Bellare, M. and Rogaway, P. (1995). Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA. ACM.

Bian, J., Seker, R., and Topaloglu, U. (2007). Off-the-record instant messaging for group conversation. In *2007 IEEE International Conf. on Information Reuse and Integration*, Las Vegas, NV, USA.

Bishop, M. (2002). *Computer Security: Art and Science*. Addison Wesley Professional.

Borisov, N., Goldberg, I., and Brewer, E. (2004). Off-the-record communication, or, why not to use pgp. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84, New York, NY, USA. ACM Press.

Bresson, E., Chevassut, O., and Pointcheval, D. (2007). Provably secure authenticated group diffie-hellman key exchange. *ACM Trans. Inf. Syst. Secur.*, 10(3):10.

Bresson, E., Chevassut, O., Pointcheval, D., and Quisquater, J.-J. (2001). Provably authenticated group diffie-hellman key exchange. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264, New York, NY, USA. ACM.

Burmester, M. and Desmedt, Y. (1994). A secure and efficient conference key distribution system (extended abstract). In *EUROCRYPT*, pages 275–286.

Canetti, R., Dwork, C., Naor, M., and Ostrovsky, R. (1996). Deniable encryption. Cryptology ePrint Archive, Report 1996/002. http://eprint.iacr.org/.

Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654.

Gaim-e (2002). Gaim-e, encryption plug-in for gaim. http://gaim-e.sourceforge.net/.

Ingemarsson, I., Tang, D. T., and Wong, C. (1982). A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5).

Koch, W. (2005). Libgcrypt - cryptographic library. http://directory.fsf.org/security/libgcrypt.html.

Krawczyk, H. (1996). Skeme: a versatile secure key exchange mechanism for internet. *sndss*, 00:114.

Pidgin-Encryption (2007). Pidgin-encryption. http://pidgin-encryption.sourceforge.net/.

Raimondo, M. D., Gennaro, R., and Krawczyk, H. (2005). Secure off-the-record messaging. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 81–89, New York, NY, USA. ACM Press.

Rivest, R. (1992). The md5 message-digest algorithm. Technical Report RFC 1321, MIT Laboratory for Computer Science and RSA Data Security, Inc.

©Secway (2006). Simppro: Instant messengers, instant security. http://www.secway.fr/us/products/simppro/.

Steiner, M., Tsudik, G., and Waidner, M. (1996). Diffie-hellman key distribution extended to group communication. In *CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security*, pages 31–37, New York, NY, USA. ACM Press.

Stinson, D. R. (2002). *Cryptography Theory and Practice, Second Edition*. CRC Press, Inc.

W.W.Peterson and D.T.Brown (1961). Cyclic codes for error detection. In *Proceedings of the IRE*.