CRYPTOLOGIA
http://dx.doi.org/10.1080/01611194.2015.1135487

Chaum's protocol for detecting man-in-the-middle: Explanation, demonstration, and timing studies for a text-messaging scenario

Alan T. Sherman, John Seymour, Akshayraj Kore, and William Newton

ABSTRACT

This article explains, demonstrates, and evaluates Chaum's protocol for detecting a man-in-the-middle (MitM) of text-messaging network communications. MitM attacks pose serious risks to many network communications. Networks often mitigate these risks with robust protocols, such as TLS, which assume some type of public-key infrastructure that provides a mechanism for the authenticated exchange of public keys. By contrast, Chaum's protocol aims to detect a MitM with *minimal* assumptions and technology, and in particular without assuming the authenticated exchange of public keys. Chaum assumes that the eavesdropper can "sound like" the communicants but that the eavesdropper cannot fabricate sensible conversations.

Using an encryption function and one-way function, Chaum's protocol works in three phases. In Phase I, the communicants exchange their public keys. In Phase II, each communicant generates a random string. The first communicant cryptographically commits to that string, and sends the string to the other communicant after receiving the other's string. In Phase III, using any of four different "scenarios" the communicants verify that each possesses the same two strings. The protocol forces any MitM to cause the communicants to possess different pairs of strings. The text-messaging scenario is similar to a forced-latency protocol proposed by Wilcox-O'Hearn in 2003.

This article implements and experimentally demonstrates the effectiveness of the third scenario, which uses timing to detect a MitM in text-messaging. Even assuming a MitM can send messages without any network latency, the protocol forces the MitM to cause delays noticeable by the communicants. This article is the first to explain, demonstrate, and evaluate Chaum's protocol, which Chaum described only in an abandoned and nearly inscrutable patent application.

ARTICLE HISTORY
Received 22 August 2015
Accepted 28 November 2015

KEYWORDS
Chaum's man-in-the-middle protocol; cryptanalysis; forced latency; Interlock protocol; key establishment; man-in-the-middle (MitM); network security protocols; U.S. Patent Application 2006/0218636 A1; Zfone protocol

1. Introduction

In 2006, David Chaum hosted seven highly capable cryptographers[1] who attend the Crypto Conference to a fancy three-hour lunch, during which he explained his latest idea: a new protocol for detecting man-in-the-middle

CONTACT Alan T. Sherman sherman@umbc.edu Cyber Defense Lab, CSEE Department, University of Maryland, 1000 Hilltop Circle, Baltimore, MD 21250, USA.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/ucry.
[1]Gilles Brassard, Ueli Maurer, Rafail Ostrovsky, Ronald Rivest, Victor Shoup, Zooko Wilcox-O'Hearn, and Moti Yung.

(MitM) in a variety of network communications using *minimal* assumptions and technology. Chaum provided the participants in advance with diagrams and written materials, yet by the end of the affair, only one of these august guests seemed to understand the proposal. Subsequently, Chaum worked with one guest (Shoup), without avail, in an attempt to prove mathematical properties of the protocol.[2] Since then, few have even heard of the protocol, which Chaum described only in a now abandoned U.S. patent application (Chaum 2006) that is nearly inscrutable.

MitM attacks remain a significant threat to network communications. For example, in February 2013, Dyn Research (2013) detected numerous MitM attacks that redirected Internet traffic from major financial institutions, governments, network service providers, and others, from the United States, South Korea, Germany, and other countries through the Internet Service Provider GlobalOneBel in Belarus. The attack also tampered with network trace utilities, making it difficult for a victim (e.g., in Virginia) to realize that someone in Minsk could monitor his web activities.

By mid-2014, blogs reported that the Chinese computer company Lenovo was shipping new computers with the Superfish malware (The Next Web.com). Superfish implements a powerful MitM attack, defeating SSL implementations by installing a bogus self-signed certificate authority. Although this malware apparently performed "only" as adware, it had the alarming potential to do much more. These two examples illustrate the need for better protection against MitM threats, including defense strategies that go beyond current SSL implementations.

Building on prior initial work of Newton (2010) and Seymour (2013), this article explains Chaum's protocol, demonstrates its effectiveness with timing studies in a text-messaging scenario, and evaluates its utility.

Chaum's protocol proposes a mitigation to the significant threat of eavesdropping by an adversary who inserts herself between two communicants in a network. Its main distinguishing characteristic is to use minimal assumptions and technology, building on his belief that fewer assumptions tend to lead to greater security. In particular, Chaum does not assume the existence of any trusted mechanism for communicants to exchange authenticated public keys. By contrast, the widely-used SSL/TLS protocols (Dierks and Rescorla 2008; Rescorla 2001) require some type of assured public-key infrastructure (PKI) that enables communicants to exchange authenticated public keys. In comparison with the Interlock (Rivest and Shamir 1984) and Zfone (Wikipedia, "Zfone") protocols, Chaum's protocol makes fewer assumptions and protects against a stronger adversary. The text-messaging scenario we analyze is similar to a little-known forced-latency protocol proposed by Wilcox-O'Hearn (Langley 2003; Wilcox-O'Hearn 2003) in 2003 on web postings.

[2]Private communications with David Chaum.

As illustrated in our accompanying animation (Cyber Defense Lab), the protocol works in three phases. First, the communicants exchange public keys without authentication assurance. Second, each communicant generates a random string. The first communicant commits to it, and sends it to the other communicant after receiving the other's string. Third, using one of four different scenarios, the communicants check if they possess the same two strings. The protocol, by means of the string commitment, forces any MitM to cause the communicants to possess different pairs of strings. We focus on the scenario for text-messaging, which we consider the simplest and most practical. In this multi-round scenario, the communicants detect a MitM through increased message delays.

The protocol has asymmetrical security properties: If the protocol completes even one initial round successfully (with a not-too-long delay), then the communicants are certain that there is no MitM. If, however, one or more rounds yield a sufficiently long delay, then the communicants strongly suspect a MitM (but the delay might have been caused by other reasons).

The protocol is especially useful when there is no available mechanism for exchanging authenticated public keys. For example, having met Bob earlier in the day at a conference, Alice later converses with Bob via text-messaging.

We experimentally demonstrate the effectiveness of the protocol. To protect against MitM, communicants must choose what assumptions they wish to accept. For example, the assumptions underlying TLS include a trusted PKI. Chaum's protocol assumes each communicant can detect semantic irregularities in the communication and knows an upper bound on the maximum delivery time of a text message. There are situations in which Chaum's protocol adds value (e.g., when a suitable PKI is not available). Regardless, there is merit in understanding its ideas and mechanisms, as well as the necessary and sufficient assumptions to effect secure sessions. Given the difficulties previous researchers encountered with this protocol, it is non-trivial from the patent application simply to understand how it works.

Our contributions include

- The first clear and correct explanation of Chaum's protocol;
- The first implementation and demonstration of Chaum's protocol, which we carry out using a text-messaging scenario;
- Experimental timing data showing that the protocol works effectively; and
- A simplification of the protocol for text-messaging, which uses a constant wait time that does not depend on the committed string.

2. Background and previous work

Defenders against MitM threats have relied primarily on three strategies: establishing authenticated public keys, exploiting shared secrets among the communicants, and leveraging characteristics of the communication channel to detect a MitM without the exchange of authenticated public keys.

Blake-Wilson, Johnson, and Menezes (1997) surveyed a variety of key agreement protocols, and Blake-Wilson and Menezes (1999) and Johnston and Gemmell (2002) analyzed authenticated Diffie-Hellman key agreement protocols.

A variety of mechanisms exist to help communicants establish authenticated public keys, including certificates, PKI, and DNSSEC. Building on certificates, the SSL/TLS protocols (Dierks and Rescorla 2008; Resorla 2001) enable communicants to establish secure sessions with confidentiality, authentication, and integrity. In some applications, however, mechanisms for exchanging authenticated public keys are not easily available, and they add complexity.

Lightweight "bottom-up" techniques, such as PGP fingerprints, have limited security (e.g., sometimes an adversary can easily edit the fingerprint without detection), and they can erode privacy by publicly associating identities and public keys.

Barak and Colleagues (2005) studied secure multi-party computations without authentication.

Meadows (2003) and Cremers, Lafourcade, and Nadeall (2009) surveyed a variety of tools for analyzing cryptographic protocols. Such tools include Cryptographic Protcol Shapes Analyzer (CPSA) (Doghmi, Guttman, and Thayer 2007),[3] Maude-NPA,[4] and Scyther.[5] Although none of these tools is configured to reason about time, it ought to be possible to adapt some of them to perform temporal reasoning using their ability to deal with sequences of events.

In the rest of this section, we explain how Chaum's protocol is stronger than the Interlock and Zfone protocols, which are two well-known protocols for detecting a MitM without assuming an authenticated exchange of public keys. We also review the forced-latency protocol of Wilcox-O'Hearn and the prosecution history of Chaum's patent application.

2.1. Interlock protocol

In 1984, Rivest and Shamir published the Interlock protocol for detecting a MitM, Eve. For Alice to send a message m_1 to Bob, using what she believes to be Bob's public key, Alice encrypts the message to produce a ciphertext c_1. Alice divides c_1 into two halfs c_{11} and c_{12}. Similarly, Bob encrypts his message M_1 to produce a ciphertext C_1, which Bob divides into two halfs C_{11} and C_{12}. Alice sends c_{11} to Bob. After Bob receives c_{11}, Bob sends C_{11} to Alice. After Alice receives C_{11}, Alice sends c_{12} to Bob. After Bob receives c_{12}, Bob sends C_{12} to Alice.

[3]https://hackage.haskell.org/package/cpsa
[4]maude.cs.uiuc.edu/tools/Maude-NPA/
[5]http://www.cs.ox.ac.uk/people/cas.cremers/scyther/

By sending half the ciphertext, Alice cryptographically commits to the message in a way that Eve cannot exploit. Eve cannot decrypt the ciphertext without the entire ciphertext, yet the protocol demands that Eve send a transmission to Bob before Alice will transmit the second half of her ciphertext. Consequently, Alice or Bob will detect Eve, either by detecting a semantic irregularity in the conversation or a timing delay. The communicants will detect this delay either in real time units or by an answer to a question being delayed by one transmitted message: A question in Alice's message m_1 might be answered in Bob's response M_3 rather than in Bob's immediate response M_2. As with Chaum's protocol, the Interlock protocol assumes that Eve cannot fabricate sensible conversations.

The protocol can be used in full-duplex or half-duplex modes. In full-duplex, each communicant transmits a message in each cycle. In half-duplex, the sender sends the entire message in two halfs before the recipient sends his response. When used in half-duplex, the communicants can detect a MitM only by assuming that Eve must increase the latency of the network communications, since she must receive the entire message before constructing a response.

Thus, the Interlock protocol assumes a weaker adversary than does Chaum's protocol. Chaum's protocol is robust against an adversary with zero network latency, but such a powerful adversary can defeat the Interlock protocol in half-duplex mode.

Bellovin and Merritt, and Ellison point out additional weaknesses of the Interlock protocol, some of which can be mitigated by forward latency (Wikipedia, "Interlock Proticol").

Rivest and Shamir assumed that the communicants can recognize each other's voices, but there is nothing in the Interlock protocol that demands this assumption.

2.2. Zfone protocol

Zimmermann's Zfone (Wikipedia, "Zfone") software secures Voice over Internet Protocol (VoIP) by implementing the 2011 ZRTP protocol (Zimmermann, Johnston, and Callas 2011), which provides some protection against a MitM. Without relying on any authenticated exchange of public keys, Zfone establishes an ephemeral Diffie-Hellman key pair. Zfone further protects subsequent communications between the parties by cached key material, using ideas similar to those described by Abdo et al. (2006).

In a crucial step of ZRTP, each communicant reads a displayed "short authentication string" (SAS), while the other verifies the utterance against the displayed SAS. ZRTP assumes that during this step, each communicant can recognize each other's voice, and that the adversary cannot sound like the speaker. The SAS is a hash of the public key. Vaudenay (1995) gave a basis of this protocol.

Thus, the Zfone protocol assumes a weaker adversary than does Chaum's protocol. Chaum's protocol is robust against an adversary who can sound like the communicants (e.g., by recording and playback), but such an adversary can defeat the Zfone protocol.

2.3. Forced latency

In the winter of 1999/2000, Wilcox-O'Hearn devised a forced-latency protocol, which is very similar to Chaum's text-messaging scenario.[6] Wilcox-O'Hearn described his protocol on his blog on 31 March 2003, and Langley offered another explanation of it on 6 April 2003. The idea of forced latency to detect MitM dates back to at least 1982 from the Interlock protocol.

In the forced-latency protocol, Alice sends a commitment of her message M, waits an agreed-upon time τ, and then sends M. Alice suspects a MitM if Bob's response arrives more than τ seconds after Alice sends M.

Wilcox-O'Hearn proposed his protocol in the context of preventing what he calls the "chess grandmaster attack," in which an unscrupulous chess player repeats his opponent's moves against a grandmaster in a concurrently-played game and then copies the grandmaster's replies in the original game. Recently, Ed Zieglar observed that the Wilcox-O'Hearn protocol is vulnerable to an attack that exploits small message spaces (e.g., small number of possible chess moves): Simultaneously initiate a protocol instance for each possible message.[7]

2.4. Prosecution history of Chaum's patent application

On 24 March 2006, building on his provisional application filed 24 March 2005, Chaum (2006) filed a U.S. patent application on a communications systems embodying his MitM detection protocol. On 30 December 2008, the Patent Office declared the application abandoned. Chaum had never responded to a communication dated 18 June 2008, rejecting all 14 claims both on the basis of indefiniteness and on the basis of anticipation over the 2006 patent application of Abdo and Colleagues, which is a continuation of a 2002 patent application. The examiner also rejected the last six claims, declaring, "There is no result produced by the claimed system that is useful, concrete, and tangible." (U.S. Patent Office, p. 3).

It is not surprising that an examiner found Chaum's application indefinite given that six out of seven hand-picked *extraordinary* "dining cryptographers" could not understand the proposal. A patent must be clear and understandable to a person of *ordinary* skill in the art. We suspect that the examiner, too, did not understand the application and hence could not appreciate its novel and possibly useful contributions.

[6]Private communications with Zooko Wilcox-O'Hearn.
[7]Private communications with Ed Zieglar.

It is our opinion that Abdo and Colleagues does not anticipate Chaum. Abdo dealt with a different technique: to extend trust from a prior authenticated session by saving shared secrets established during that session. Wilcox-O'Hearn, however, did anticipate some claims, though his disclosures are in a chess setting.

3. Assumptions and adversarial model

We assume no trusted third party nor any trusted mechanism for exchanging authenticated public keys.

The primary goal of the adversary is to insert herself between the communicants, relaying messages from each to the other, for the purpose of eavesdropping on the conversation.

We say that the adversary has broken the protocol if she can read or modify at least one bit of the plaintext conversation without detection. The protocol does not aim to prevent an adversary from intercepting and relaying the ciphertext without modification.

We assume a powerful adversary who can "sound like" each of the communicants, either by recording and playing back transmissions of the communicants, or by impersonating the sound of their voices. We assume, however, that the adversary cannot "think like" either communicant; that is, the adversary cannot fabricate sensible conversations without detection. Thus, we assume that each communicant can detect any "semantic irregularity" in the communications, as might be revealed, for example, by one of them asking a personal question (e.g., "What did you eat with me at dinner last Tuesday?"). The number of bits of such shared personal information is limited.

We assume each communicant has a random number generator and computationally-secure standard cryptographic primitives, including a one-way function, encryption function, hash function, and pseudorandom number generator.

We assume the adversary has powerful network capabilities including the ability to send and receive network communications without any network latency. The protocol aims to be secure even when the adversary's network capabilities greatly exceed those of the communicants.

Scenario 3 assumes each communicant can observe the time at which he or she sends or receives any message. This scenario also assumes each communicant knows a typical maximum time required to send a message and receive a response, including the time to construct the response.

Some of the scenarios assume further that the communicants share a common public string, such as a book or database of jokes, also known to the adversary. Some also assume that the communicants can embed a short string (e.g., an index into the common public string) in a semantically meaningful utterance in such a way that the adversary cannot modify the embedded string without detection.

4. The Chaum protocol

Chaum's protocol works in three phases: (I) Exchange public keys without authentication; (II) commit to a string; x and exchange string, and (III) verify if the communicants possess the same strings. Each communicant will possess the same pair of strings if and only if there is no eavesdropper in the middle.

Separately, as we summarize in the appendix, Chaum additionally proposes three ways for communicants to expand their "web of trust" by leveraging their trust in the credentials (public keys) of common friends.

We shall assume Alice (A) is communicating with Bob (B), possibly in the presence of eavesdropper Eve (E).

We assume that Alice and Bob must follow the protocol, but Eve may do whatever she wishes. If either communicant detects non-compliance with the protocol, that communicant terminates the protocol. Similarly, if either communicant detects any semantic irregularity in the conversation, that communicant terminates the protocol and assumes that Eve is present.

Two of Eve's strategies are (1) block all communications between Alice and Bob, and instead hold separate parallel conversations with each. The assumption that Eve cannot fabricate sensible conversations ensures that the communicants will detect this strategy, but no technical aspect of the protocol prevents this strategy. (2) Sit in the middle between Alice and Bob, relaying all messages from each to the other. In this second strategy, the committed string forces Eve to establish separate string pairs with Alice than with Bob. In Phase III, each scenario enables Alice and Bob to detect that they possess different string pairs.

Throughout, we explain the protocol with the help of diagrams showing what happens without Eve, and what might typically happen with Eve present.

In Phase I, Alice and Bob exchange public keys and establish a session key. All subsequent communications between Alice and Bob are encrypted with this session key. Because the effectiveness of the protocol does not depend on this layer of encryption, for simplicity and clarity, we omit it from all of our protocol diagrams. Use of this session key forces Eve, if present, to be present initially and throughout the conversation. Also, it protects the conversation from anyone who does not know the session key.

Also for simplicity and clarity, we omit other standard details from our protocol diagrams, such as the presence of unique sequence numbers, timestamps, and protocol instantiation numbers cryptographically bound to each communication.

4.1. Notation

Let $E_k(m)$ denote the encryption of message m under key k; let f be a one-way function; and let h be a cryptographic hash function. Let $||$ denote string concatenation. Let $\text{trunc}_{128}()$ mean truncate to 128 bits.

Let p_A, p_B, p_E denote the public keys of Alice, Bob, and Eve, respectively, and let s_A, s_B, s_E denote the corresponding private keys.

Our notation, which differs from Chaum's, relates to notation in Chaum's patent application as follows. Chaum introduces the symbols C and D, which represent the communication devices (e.g., computers) of Alice and Bob, respectively. Chaum denotes the public keys of Alice, Bob, and Eve as q^c, q^d, and q^f, respectively, appealing to a context involving discrete logarithms. He denotes the corresponding private keys as c, d, and f. In the patent application, r^{cd} denotes a session key used by Alice and Bob. Chaum writes the expression $y = x \oplus x'$ to denote a general combining operator $\oplus$, which is not necessarily exclusive-or.

4.2. Phase I: Exchange public keys

Alice and Bob exchange public keys using any unauthenticated key exchange protocol, such as the Diffie-Hellman protocol (Rescorla 1999). They also establish a session key, used to encrypt all subsequent communications between Alice and Bob.

After Phase I, there are two cases to consider: Eve was present, and Eve was not present. If Eve was not present, then Alice and Bob exchanged their correct public keys, and they established a session key known only to them.

If Eve was present, then unknown to Alice and Bob, Alice and Eve exchanged public keys p_A, p_E and established a session key k_1, while Eve and Bob exchanged public keys p_E, p_B and established a session key k_2. Alice thinks p_E is possibly Bob's public key, and Bob thinks p_E is possibly Alice's public key.

Only in Phase III will Alice and Bob learn which case is true.

4.3. Phase II: Commit to string x

Alice generates a random string x_A and shares it with Bob, and Bob generates a random string x_B and shares it with Alice. Importantly, Alice commits to x_A by sending $f(x_A, p_A, p_B)$ before Bob shares x_B. Without committing to x_B, Bob sends x_B to Alice. After receiving x_B, Alice shares x_A.

Let $\hat{x}_A$ denote the string Bob receives purportedly from Alice, and let $\hat{x}_B$ denote the string Alice receives purportedly from Bob. If Eve is present, the string x_A sent by Alice might not be the string $\hat{x}_A$ received by Bob.

Alice computes the string pair $y_A = x_A||\hat{x}_B$, and Bob computes the string pair $y_B = \hat{x}_A||x_B$.

Figures 1 and 2 show this process in detail. Before computing y_B, Bob verifies the correctness of the received commitment $f(x_A, p_A, p_B)$ by recomputing it, using his knowledge of $\hat{x}_A, p_A, p_B$. Bob terminates the protocol if this verification fails.

If Eve is not present, then $y_A = y_B$. If Eve is present, the protocol with its commitment forces Eve to choose her value x_E to be sent to Bob before learning what value x_A Alice chose. As a result, at the end of Phase II, Alice and Eve compute the string pair y_A, and Eve and Bob compute the string pair y_B, with $y_A \neq y_B$.

One could skip Phase II and simply take y_A and y_B to be $p_A||p_B$, as understood by Alice and Bob, respectively. The benefit of Phase II is to establish fresh values y_A, y_B for each session.

4.4. Phase III: Scenarios to detect if strings differ

In Phase III, Alice and Bob detect if Eve is present by determining if the strings y_A and y_B they computed in Phase II differ. To do so, they carry out one of four different "scenarios," which exploit various aspects of the communications channel (e.g., timing, jitter) or which exploit a common public string. The appendix summarizes how each of the four scenarios work.

Each scenario works not by keeping the strings y_A and y_B secret from Eve (Eve knows them), but by enabling Alice and Bob to detect when these strings differ, even with Eve is in the middle.

We focus on Scenario 3, a text-messaging scenario involving timing, because we consider it the simplest and most practical of the scenarios from the patent application. Each scenario implements a "ping-pong" protocol comprising a sequence of "rounds," where each round consists of a sequence

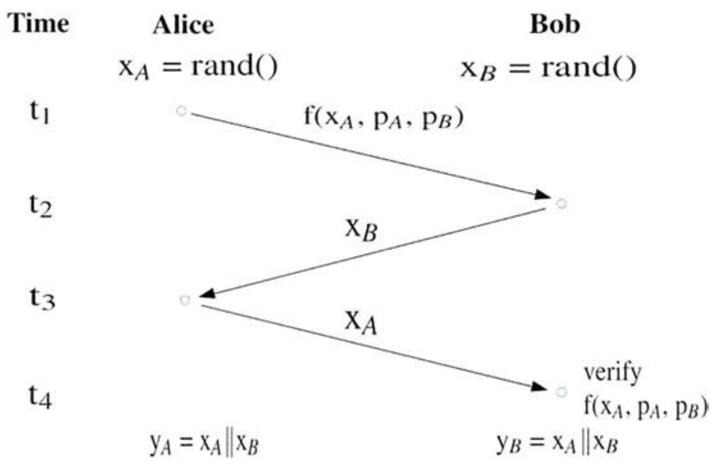

Figure 1. Phase II of Chaum's protocol, without Eve present. Alice first commits to her random string x_A but waits to receive Bob's x_B before Alice sends her x_A to Bob.

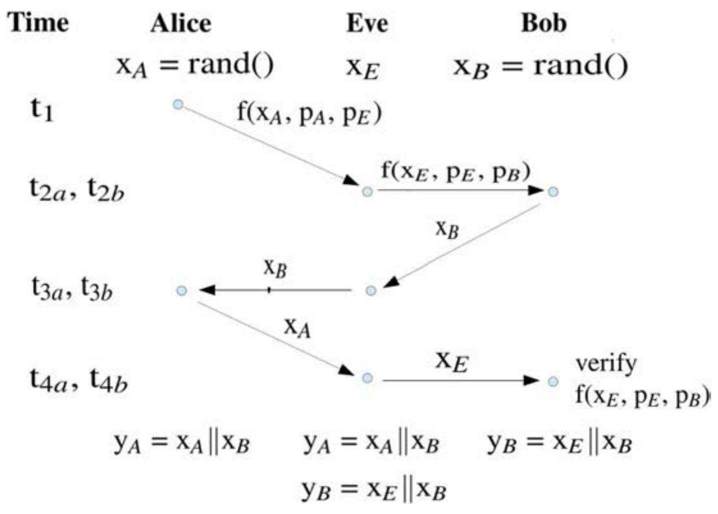

Figure 2. Phase II of Chaum's protocol, with Eve present. Eve must commit to a string x_E before learning Alice's x_A. As a result, Alice and Bob end up with different string pairs $y_A \neq y_B$.

of transmissions (we call them "pings" and"pongs") alternately sent from one party to the other.

Throughout the conversation, and especially at the beginning, each communicant monitors the conversation for any possible semantic irregularity (typically this monitoring exploits informal shared secrets); if any irregularity is detected, the communicant terminates the protocol suspecting a MitM.

5. Scenario 3: Text-messaging

In this scenario, Alice measures the elapsed time from sending a message to Bob to receipt of his response. A cryptographic commitment combined with a mandatory wait by Bob forces a MitM to double this time, even assuming a powerful adversary with no network latency. We now explain how the scenario works and why it works correctly. We also propose a simplification based on constant wait times.

This scenario requires Alice and Bob beforehand to have chosen some agreed-upon wait time, which we assume is also known by Eve. A simple policy is to use a common default wait time that depends only on the network technology. This time is agreed upon securely in advance and is not negotiated in the presence of Eve.

5.1. How Scenario 3 works

At the end of Phase II, Alice computed a string y_A, and Bob computed a string y_B. Alice and Bob wish to determine if $y_A \neq y_B$, indicating presence of a MitM.

As illustrated in Figure 3, Alice begins the first round by selecting a round key k at random. She salts this key with her y_A to produce the salted key k'

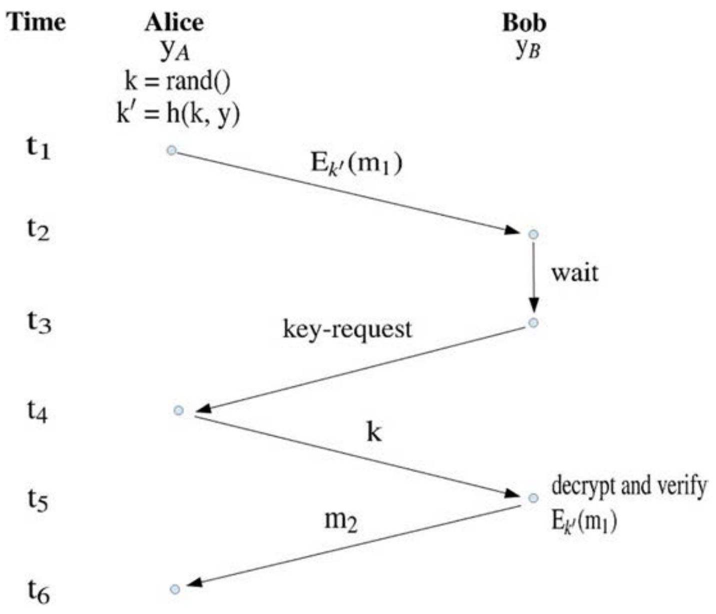

Figure 3. Scenario 3, without Eve present. Alice commits to message m_1 by encrypting it with k', a randomly chosen key k salted with her string pair y_A. Bob waits and then sends a key request. Alice sends the unsalted key, after which Bob sends his reply message m_2.

(e.g., $k' = \text{trunc}_{128}(h(k, y_A))$). The details of the salting technique are unimportant, except that it should be free of extreme algebraic weakness.

Next, Alice sends Bob the commitment $E_{k'}(m)$, where m is the next text block of the conversation. Upon receipt, Bob waits a pre-agreed upon time (e.g., a constant amount) and then sends Alice a key request.

Upon receipt of the key request, Alice first checks that the elapsed time is neither too long nor too short. Less than the expected wait time is too short; twice the expected wait time is too long. If the elapsed time is out of bounds, then Alice suspects a MitM. If the elapsed time is within bounds, then Alice sends the *unsalted* key k to Bob.

Using k and the salt y_B, as known to Bob, Bob calculates k' and deciphers the message. Each communicant always checks the received message for semantic irregularity, terminating the conversation if any is found.

Bob then constructs his response and sends it to Alice. There are multiple variations of how the protocol continues. One choice is for Bob, in his next message to Alice, to begin the next round of the scenario, playing the role of Alice. Regardless, Alice or Bob selects a new round key for each round.

Another variation, which we implement in our prototype, is to "dovetail" the current round with the next round. In this variation, Bob's last message of the current round begins the next round, with Bob then playing the role of Alice from the previous round. Thus, the first "ping" of the next round overlaps with the last "pong" of the current round.

In this dovetailing variation, upon receipt of Bob's last message from the current round, Alice will be unable to decipher it immediately. She must wait to receive the round key chosen by Bob. Upon receipt of this key, Alice will be able to decipher the message and check it for semantic consistency. With dovetailing, if Eve is present, Alice will notice a tripling of the expected wait time.

5.2. Why Scenario 3 works correctly

When Eve is present, it is convenient to refer to the round key selected by Alice as $k = k_A$, to distinguish it from a round key k_E chosen by Eve.

A MitM must, in some way, relay messages between the communicants. Although Eve knows y_A and y_B, which will be different when she is present, she does not initially know k_A, and hence she does not initially know its salted version k'_A. Two of her choices are relay $E_{k'_A}(m)$ from Alice to Bob without modification, or first determine k_A and use it to decrypt $E_{k'_A}(m)$ and then relay a re-encryption $E_{k'_E}(m)$, where $k'_E = \text{trunc}_{128}(h(k_E, y_B))$ is the salted key using salt y_B of another key k_E chosen by Eve. Both strategies fail.

If Eve relays $E_{k'_A}(m)$ without modification, Bob will detect the MitM because Bob is expecting Alice to use salt y_B, but Alice used y_A. Importantly, the security of E and h prevent Eve from finding another key that Bob could use with salt y_B to decipher the unmodified relayed message. Furthermore, Eve cannot modify the plaintext without detection because doing so would create a semantic irregularity.

If Eve first determines k_A, she must wait the time expected by Alice. She can try to cheat by waiting slightly less (hoping network latency will cause the key request not to arrive too soon), but if she waits substantially less, Alice will detect that Eve's key-request arrived too soon (in less than the expected wait time). Since Bob will eventually wait the expected amount of time, Eve will have caused the elapsed time for Bob's response to reach Alice to have at least doubled. Alice will notice this increased latency (Figure 4).

5.3. Constant vs. variable wait time

For the agreed upon wait time, we recommend using a constant wait time. This wait time should be carefully selected to be large in comparison with typically experienced network latencies, also considering the time it takes for the recipient to construct a response. A doubling of this time should be clearly noticeable to the communicants. A wait time could be chosen based on a typical maximum delay for the type of network being used.

Our suggestion is a simplification and security improvement over Chaum's proposal. Without giving details, Chaum originally proposed that variable wait times be determined from the y values. For example, one might first choose a small number (e.g., 2 or 4) of wait times. Then, one might use y as a seed into a

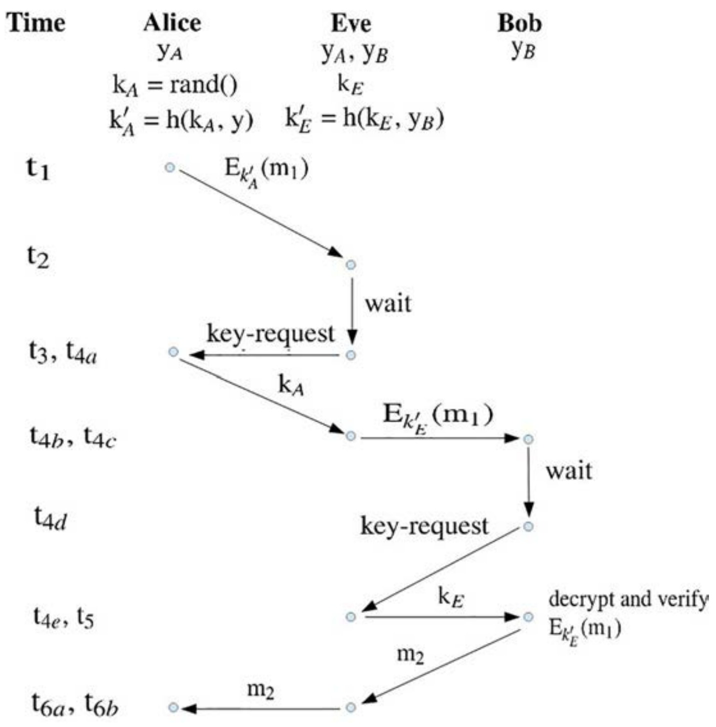

Figure 4. Scenario 3, with Eve present. Alice encrypts message m_1 using k'_A, a randomly selected key k_A salted with her string pair y_A. Because Bob is expecting Alice to use the different salt y_B, Eve must first obtain k_A before relaying m_1 to Bob. Doing so forces Eve to double the expected wait time observed by Alice before Alice receives Bob's reply message m_2. Alice will terminate the protocol if the key request arrives too soon.

pseudorandom number generator to produce a sequence of values $r_1, r_2, \ldots$. For each round i, use r_i to select a wait time for that round, selecting from the small chosen set of possible wait times. Observed wait times significantly different from those expected would be evidence of a MitM.

Originally, Chaum felt that variable wait times offered another check on whether $y_A = y_B$.[8] However, such an additional check is unnecessary. Furthermore, as realized by Chaum, with variable wait times, not all time observations will yield useful evidence (i.e., if Alice is expecting a long wait time, and Bob is waiting a short time, then Eve can delay her responses to Alice without detection).

6. Experimental evaluation of Scenario 3

To evaluate how well Chaum's protocol with Scenario 3 works, we implemented it and collected timing data, with and without a MitM. We now describe our implementation and experimental methods and summarize the results of

[8]Private communications with David Chaum.

our timing studies. Our results show that the protocol with Scenario 3 is realizable and effective.

6.1. Purpose

The main purpose of our experimental work is to provide an initial evaluation of the feasibility and effectiveness of Chaum's protocol with Scenario 3. We also wish to confirm that our foregoing discussion has addressed all crucial issues and to uncover any possible neglected pragmatic concerns.

6.2. Implementation

In our demonstration version, two subjects can engage in a text-messaging conversation by typing messages into windows on separate laptop machines connected through a server on our university's network. When Eve is present, her code runs on Alice's laptop.

When a communicant clicks the send button, a software client transfers a text file from the sender's machine to the recipient's machine. For convenience, and invisible to the communicants, we implemented this file transfer using Dropbox (Drago et al. 2012), with clients for Alice and Bob sharing a Dropbox folder on their separate machines.

We choose to use Dropbox because it provides a simple means for implementing text-messaging communications through a real network, and because it exemplifies a large-latency communication system (i.e., Dropbox is slow). For comparison, we also tested the system using a faster file-transfer technique: Sharing a single machine, clients for Alice and Bob simply moved files within the same file system. With Dropbox, typical times to transfer a file from one machine to the other were approximately 60–180 seconds, compared with approximately 200 milliseconds for our faster alternative.

We implemented Chaum's protocol in JAVA using the javax.crypto API (Oracle, "Java Cryptography"; Oracle, "Pakage Javax.crypto"). We used AES encryption with 128-bit keys and message blocks, and SHA-3 hash which produces a 160-bit output. We computed the one-way function as SHA-3.

We carried our timing studies using a HP-Pavilion dv6 laptop with an Intel (R) Core (TM) i7 2.20 GHz processor using 4.00 GB RAM running under 64-bit Windows 7.

Our implementation permits human subjects to play the roles of Alice and Bob. The goal of our implementation, however, is to demonstrate feasibility and to collect timing data, not to provide polished and secure code.

6.3. Methods

We collected timing data on several executions of Scenario 3, with four "dovetailed" rounds per conversation (see Section 5.1), creating four observed wait

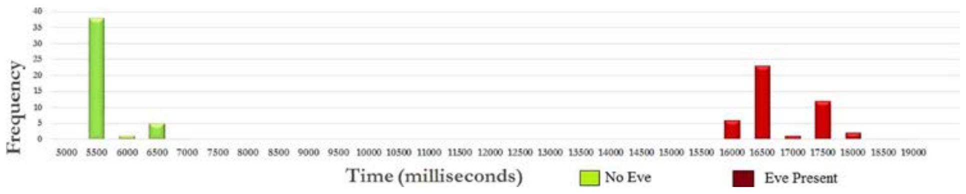

Figure 5. Histogram of elapsed wait times for Scenario 3, with and without Eve, with simulated communicants.

times per protocol instance. We did so separately with and without Eve present. When Eve was present, we followed the typical pattern of behavior for Eve described in Figure 4.

We collected timing data both with real subjects carrying on a conversation with text-messaging and with a basic computer simulation of the packet exchanges in such a conversation (without meaningful semantic content). In this simulation, each text message comprised 50 characters.

For our "fast" network, we set the required wait time to be 5,000 milliseconds. We do not report data from our "slow" network because our subjects found it annoyingly slow.

6.4. Timing study results

Figures 5 and 6 show the observed wait times in executions of a four-round protocol, with and without a MitM. Figure 5 shows wait times from 22 instances of a computer simulation of a conversation, using our "fast" network (Eve was present in 11 instances). Figure 6 shows waits times from ten instances with human communicants also using our "fast" network (Eve was present in five instances).

6.5. Analysis

For the computer-simulated conversation (Figure 5), there is a clean separation between when Eve is present or not present. Due to the dovetailing, the wait time triples when Eve is present. With human subjects (Figure 6),

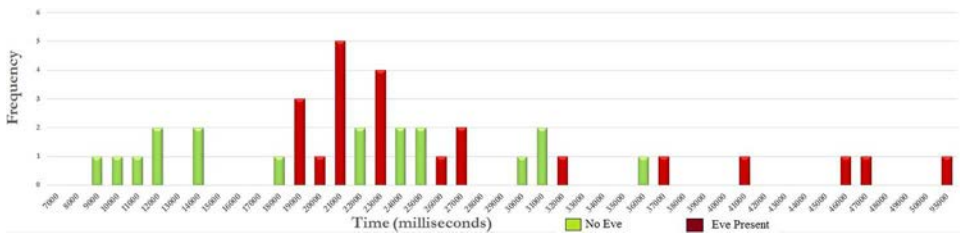

Figure 6. Histogram of elapsed wait times for Scenario 3, with and without Eve, with human communicants.

these two cases are less cleanly separated, due to significant variation in the time subjects took to construct responses. Fast response times prove that Eve is not present, and slow response times are inconclusive.

These results suggest that the communicants should endeavour to respond quickly during the initial part of each conversation, until they reach assurance that Eve is not present.

7. Discussion

We now discuss several issues, observations, and open problems related to Chaum's protocol, including when the protocol is useful and the decision-making process for deciding if a MitM is present.

7.1. Contexts when protocol is useful

The protocol is useful for conversations when the communicants have not established authenticated public keys or other shared secrets, and there is no mechanism for obtaining their authenticated public keys. Once the communicants establish authenticated public keys or shared secrets, other more traditional and robust protocols could be applied.

In the future, it may be easier for more people to exchange public keys assuredly. For example, one could include a QR code of one's public key on a business card. Yet, today, many people still find it a challenge to exchange authenticated public keys.

Using the protocol for first-time conversations might sometimes be attractive because the communicants might not have yet established authenticated keys. On the other hand, using the protocol for first-time conversations is problematic because the communicants might not have established required common knowledge or latency characteristics of the communications channel.

In principle, after one trusted communication, the parties could establish authenticated public keys. Even so, there still can be value in the Chaum protocol, for example, for environments which lack the infrastructure to establish and store authenticated public keys, or as a layer of protection against an attack that tampered with the authenticated public keys.

7.2. Issues and observations

As does the Interlock (Rivest and Shamir 1984) protocol, Chaum's protocol leverages *cryptographic commitment* to detect a MitM in a variety of scenarios. Chaum's protocol improves on Interlock by resisting an adversary with zero network latency. It improves on the Zfone (Wikipedia, "Zfone") protocol by resisting an adversary who can impersonate the voices of the communicants. Although unusual delays can cause the protocol to suspect a MitM is present when one is not, the protocol will always detect a MitM when present.

The main limitation of Chaum's protocol is its assumption, also held by Interlock, that the communicants can detect any semantic irregularity in the conversation. This assumption places an undue burden on the communicants and violates the security principle that network security protocols should not depend on the judgements of the communicants. Another limitation is that the mandatory wait periods necessarily increase communication latency.

Furthermore, the communications must use a network for which they have an accurate estimate of the maximum likely network latency. The larger this maximum latency, the more the protocol will increase communication latency. On the other hand, because the protocol must only be run during the initial few rounds of the conversation, any increase in communication latency would be limited.

Alice and Bob might begin each session with some personal talk about some personal experiences that they shared in common that only they would know. Only after they verify that there is no MitM, should they start the sensitive part of the conversation. If a MitM was not present initially, she could not insert herself later in the session; furthermore, future sessions with those public keys would also be assured.

In Scenario 3, the dependence on timing limits the utility of the protocol to synchronous conversations where both parties are texting on-line in a fashion where the time for each response is fairly predictable. Scenario 3 does not work for asynchronous text-messaging where one or both parties might wait long and unpredictable amounts of time before responding.

The patent application does not clearly explain how the communicants should agree on the expected wait times. Chaum assumes that the communicants typically have a history of communicating with each other, or with other people on similar networks, from which the wait time could be tuned and the observed elapsed times could be interpreted. This assumption might not always apply, and it raises the possibility that Eve might interfere with communications that influence the agreed upon wait time.

Some of the scenarios assume a "common public string model" of security in which the participants share a common string also known to the adversary. For example, in Scenario 1, the communicants refer to a book or database of jokes. In Scenario 3, the pre-agreed upon wait time(s) might also be viewed in this context.

Scenario 1 also assumes that the communicants can embed information, at least in small amounts, into a semantically-sensible conversation. (e.g., referring to a joke or punch line). This assumption is similar in spirit to the stronger assumption in Zfone that each communicant can utter a short string in a way that the adversary cannot impersonate. These are powerful assumptions: If the communicants could so embed their public keys (or hash fingerprints of these keys) into the conversation with integrity, then the communicants could exchange authenticated public keys. Yet, some people may prefer to accept these assumptions over those underlying SSL/TLS.

Because the protocol can falsely suspect a MitM when none is present, an adversary might be able to cause various mischief, including denial of service, by simply causing an increase in network latency, for example by sending large video files. Before an actual attack, the adversary might provoke a series of false alarms, hoping that subsequently during the real attack, the communicants will have become insensitive to a true alarm. Similarly, in the "Bayesian" decision-making interpretation (see Section 7.3), it might be confusing to the communicants what to do when the threat indicator light shines "amber" (caution).

Although the patent application does not do so, it must be specified what should be done in each possible error state, including if a packet is lost in transit or if the session abends. In Scenario 3, any round with a lost packet should be viewed as unsuccessful and suspicious and should not be interpreted as evidence of the absence of a MitM. If the session abends, the participants should restart the protocol from the beginning. Failure to handle all such error states properly can create vulnerabilities.

Because of the security need to view lost packets with suspicion, the protocol as proposed has significant limitations for use in unreliable communications, including text-messaging with poor reception.

In practice, the communicants will likely share only a limited amount of private information needed to establish semantic consistency (e.g., "What happened at dinner last night?"). Consequently, an issue arises if this private shared information is exhausted. An attacker might try to force the communicants to exhaust their shared information by causing them to restart the protocol repeatedly, for example, under the guise of lost packets.

7.3. The decision-making process

Chaum conceived of Phase III not necessarily as a *"categorical"* process, where after a short fixed number of communications each party would reach a certain conclusion. Rather, he envisioned an interpretive *"Bayesian"* process: Each communicant starts with an *a priori* belief in the presence of Eve, and updates that belief during the conversation in light of evidence derived from the scenario. Each communicant interprets the evidence in the context of his or her expectations of certain properties of the communication channel and communicants, perhaps based in part on prior history of similar communications between the parties. For decision making, each session is independent.

Concretely, there might be a colored light whose color signals the current degree of belief in the presence of a MitM. This light might start out amber (caution), and during the conversation the color might change to green (safe) or red (suspected MitM).[9]

[9]Ibid.

In Scenario 3, a single round without undue delay is convincing evidence of no MitM. A long wait time, however, could be caused by a MitM or by unusual network latency. Upon observing a long wait for each of several rounds, taking a categorical view, Alice would either restart the protocol suspecting unusual network latency, or terminate the protocol suspecting a MitM. Taking a more flexible view, Alice might continue cautiously updating her belief in the possibility of a MitM.

While the Bayesian decision-making interpretation of the protocol offers some attractive flexibility, and although it might reduce the number of false alarms, we dislike its complexity and its need to be applied in more rounds. We prefer the simpler categorical interpretation, which can be carried out in a small number of rounds.

7.4. Open problems

Open problems include formally stating and proving the security properties of the protocol in a precise mathematical fashion. One difficulty in doing so is precisely characterizing the vaguely stated assumptions, for example, that communicants can detect semantic irregularities, or that they can embed a short string into semantic content.

Another approach is to analyze the protocol with formal tools for analyzing cryptographic protocols, though most such tools are not well configured to analyze forced timing properties.

We focused on the text-messaging scenario. It would be interesting to explore the other scenarios, too.

It would also be interesting to perform a careful usability study of the protocol. Such a study might compare Chaum's protocol with SSL/TLS and a baseline text-messaging communications session without any protections.

8. Conclusion

We explained Chaum's protocol, which detects a MitM using minimal assumptions and technology. We implemented Scenario 3 of the protocol for text-messaging and through timing studies demonstrated that it works effectively. When it is used, for simplicity, we recommend that it be implemented in its "categorical" style of decision making and to use constant wait times.

The protocol is especially useful when the communicants have no available method for exchanging authenticated public keys. In comparison with the Interlock and Zfone protocols, it protects against more powerful adversaries. A nice feature is that it always detects Eve when she is present.

It is possible, however, for the protocol to detect Eve falsely, for example, if there are unusual network delays. Also, through mandatory waiting periods, the protocol necessarily increases communication latency. Although the protocol aims to use minimal assumptions, it in fact requires several strong assumptions: Each communicant must detect any fabricated conversation; the network latency is always or mostly below a specified maximum; and (for some scenarios) each communicant can embed information into the semantic content of a conversation.

Chaum's protocol illustrates a tension between two desirable principles for building secure systems: minimal assumptions and ease-of-use. There is considerable merit to the principle that fewer assumptions tend to result in greater security. In particular, with fewer assumptions, there are fewer assumptions that might be violated. Chaum's contribution is a new protocol for detecting MitM with minimal assumptions. Secure systems, however, require some assumed security foundation, and a more substantial foundation can result in protocols that are easier to use. Protocols that are easier to use are more likely to be used and used correctly. Unfortunately, while simpler in some respects than SSL/TLS, Chaum's protocol, as does the Interlock protocol, has an element that makes it harder to use: The communicants must detect any semantic irregularity in the conversation.

It is a serious limitation of any protocol to require the communicants to make any judgements. We feel the protocol is simple enough for many communicants to carry it out, but we distrust many of them to detect semantic irregularity assuredly. We strongly prefer protocols that are fully automatic and that yield high-assurance categorical results, even if they require stronger assumptions, provided the assumptions are feasible. For this reason, we prefer detecting a MitM with the robust, albeit somewhat complex, SSL/TLS protocols and to provide security environments that include a trustworthy public-key infrastructure, or some other suitable mechanism for assuredly exchanging authenticated public keys.

Improving on the Interlock protocol and exploiting the power of cryptographic commitments, Chaum's protocol leverages informal shared secrets to enable the communicants to detect a MitM, after exchanging public keys in an unauthenticated fashion. These informal shared secrets (i.e., common knowledge among the communicants) permit the communicants to detect semantic irregularities. Some people might prefer to accept the assumptions underlying Chaum's protocol rather than those underlying SSL/TLS. Our prototype demonstrates that the protocol works effectively, even when the adversary can send and receive messages with zero network latency.

We hope that our explication and informal analysis of Chaum's protocol will help others appreciate and learn from its novel and interesting facets.

Appendix: The scenarios and common friends

We briefly summarize how each of four scenarios works to enable Alice and Bob to determine if their string pairs y and y' are different, indicating a MitM. We also briefly summarize how to leverage trust through common friends.

A.1. Four scenarios

A.1.1. Scenario 1 (live audio/visual, common reference)

As described in paragraphs 0048–0049 of the patent application (Chaum 2006), each communicant uses his or her y value as an index into a pre-agreed upon public database (e.g., of jokes). Alice speaks the joke question, and Bob speaks the punch line. Each of these utterances is embedded into a conversation with semantic content. Each party verifies that the punch line matches the question.

There are many ways in which this idea can be embodied. One way, suggested by Chaum, is as follows.[10] Alice picks a random key k and salts it with y (e.g., $k_s = \text{trunc}_{128}(h(k, y))$. Alice sends to Bob the joke question m encrypted as $E_{k_s}(m)$. Bob then sends to Alice the punch line. After Alice receives the punch line, she sends Bob the unsalted key k, from which he can compute k_s and decrypt the question. Each side verifies that the punch line matches the question.

In another embodiment, each communicant interprets his or her y as the sequence of binary choices in some pre-agreed upon two-party game. With overwhelming probability, the outcome of the game will be the same if and only if their strings match.

A.1.2. Scenario 2 (live audio/visual, directional flow)

As described in paragraphs 0050–0054 of the patent application (Chaum 2009), each communicant derives from his or her string y four numbers, y_1, y_2, y_3, y_4, which they interpret as time intervals. The communicants then use these time intervals to regulate the directional flow of their communications.

During time interval y_1, only Bob speaks to Alice. During interval y_2, the communications are bidirectional. During interval y_3, only Alice speaks to Bob, and during interval y_4, neither party speaks. Throughout intervals y_1–y_3, each communicant engages in a semantically meaningful conversation.

If the y strings held by Alice and Bob differ, then a MitM will be unable to prevent the communicants from noticing either a deviation from the expected pattern of directional flow, unnatural delays in the conversation, or semantic irregularities.

A.1.3. Scenario 3 (text-messaging, timing)

As described in paragraphs 0055–0059 and Figure 4 of the patent application (Chaum 2006), Alice picks a key k at random and salts it with her y to produce the salted key k_s. She sends a message encrypted with k_s. After waiting a specified (we recommend constant) time, Bob sends Alice a key request, in response to which Alice sends the unsalted key k.

[10]Ibid.

As explained in Section 5, this scenario forces Eve to notably increase the time Alice observes from sending a message to Bob and receiving his response. This notable increase in time is not the result of any network latency caused by Eve. Before relaying Alice's message to Bob, Eve must first send a key request to Alice. Bob will always wait to send his key request, and if Eve fails to wait long enough, Alice will notice that she received a key request too soon.

This scenario proceeds in a sequence of rounds, where in each round Alice and Bob reverse their roles as sender and recipient. In one interpretation, these rounds overlap in the sense that the last "pong" of Bob to Alice of the current round is also the first "ping" of Bob to Alice in the next round.

A.1.4. Scenario 4 (live audio/video chat, jitter)

As described in paragraphs 0060–0062 of the patent application (Chaum 2006), this scenario is a variation of Scenario 3 in which there is a constant stream of packets. Unlike Scenario 3, in Scenario 4, the order of messages is strictly sequential and without overlapping rounds. When a MitM is present, the communicants will notice a jitter or delay in the communications.

Occasionally, Alice will "mark" a packet by encrypting it using a salted key as described in Scenario 3. Alice will then include the unsalted key in the payload of the next packet.

Thus, even if Eve has zero network latency, she is forced to delay the stream of marked packets by one packet. The assumption is that this delay will be sufficient to cause a detectable jitter.

As with Scenario 3, Scenario 4 proceeds in a sequence of rounds, where in each round Alice and Bob reverse their roles as sender and recipient.

A.2. Common friends

We briefly summarize three methods, which we shall call Techniques i–iii, that enable communicants to leverage their trusted credentials of others in a bottom-up web of trust, exploiting common "friends" and common "friends of friends." For more details, see (Chaum 2006) and (Newton 2010).

A.2.1. Technique i (common friend)

If Alice and Bob each possess the trusted credentials (public key) of a known common friend, they can leverage this credential to establish trust in each other's public keys. Bob sends Alice a list of credentials for possible common friends. Alice returns a commitment of these credentials and marks them using her private key. Bob then proves his identity to Alice by marking one of Alice's credentials and checking the committed credentials. Finally, Alice repeats the entire process playing the role of Bob. Failure of any proof detects Eve.

A.2.2. Technique ii (anonymous common friend)

This technique is Similar to Technique i, but Alice and Bob mask their authenticators with random numbers to avoid revealing the identity of the common friend, both to each other and to Eve.

A.2.3. Technique iii (common friend of friend)

This technique leverages the situation where Alice trusts the credentials of a Friend-1; Bob trusts the credential of a Friend-2; and Friend-1 shares a common trusted friend with Friend 2. First, Friend-1 and Friend-2 prove to each other that their credentials are fresh. Second, using Technique i, Friend-1 and Alice re-establish trust. Similarly, Bob and Friend-2 re-establish trust. Third, using Technique i, Alice and Bob establish trust.

About the authors

Alan T. Sherman is a professor of computer science at the University of Maryland, Baltimore County (UMBC) in the CSEE Department and Director of UMBC's Center for Information Security and Assurance. His main research interest is high-integrity voting systems. He has carried out research in election systems, algorithm design, cryptanalysis, theoretical foundations for cryptography, applications of cryptography, cloud forensics, and cybersecurity education. Dr. Sherman is also a private consultant performing security analyses. Sherman earned the PhD degree in computer science at MIT in 1987 studying under Ronald L. Rivest (see www.csee.umbc.edu/~sherman).

John Seymour is a PhD student of computer science at the University of Maryland, Baltimore County (UMBC) where he performs research at the intersection of machine learning and information security under the supervision of Dr. Charles Nicholas. His tentative PhD dissertation topic is quantifying value in open source malware datasets. In 2014, he completed his master's thesis, titled, "Quantum Classification of Malware," which was later presented at DEFCON 23. He currently performs machine learning research, quantum computing research, and work on the DARPA STAC project at CyberPoint International, LLC, located in Baltimore.

Akshayraj Kore works as an Android programmer for the startup Apio Systems in Virginia. He is a member of the lead Android team which builds situational awareness apps for driver safety and driver behavior improvement. Raj is also an International Grandmaster, playing for UMBC's chess team for two years while pursuing his master's in Computer Science at UMBC. His main research interests are cryptography, programming languages, and algorithms.

William Newton is a software development project manager at Booz Allen Hamilton. He received his bachelor's in 2004 and master's in 2010 from the University of Maryland, Baltimore County (UMBC). His main research interests are wireless security and related vulnerabilities, unique steganography techniques, and detecting and analyzing man-in-the-middle attacks.

Acknowledgments

We thank David Chaum and Zooko Wilcox-O'Hearn for fruitful discussions. Thanks also to Jonathan Katz, Neal McBurnett, and members of the UMBC Cyber Defense Lab (including Edward Birrane, Josiah Dykstra, Russ Fink, Christopher Vatcher, Michael Oehler, and Ed Zieglar) for useful feedback. William E. Byrd helped guide Kore's software implementation.

Funding

Alan T. Sherman was supported in part by the National Science Foundation under SFS grant 1241576 and supplement, and by the Department of Defense under IASP grants H98230-10-1-0359, H98230-11-1-0473, and H98230-12-1-0454.

References

Abdo, A. Y., Overton, A. J., Garms, J., and Parsons, J. E. Jr. Automatic re-authentication, U.S. Patent Application Publication 2006/0117106 A1: (1 June 2006) 1–13.

Barak, B., Canetti, R., Lindell, Y., Pass, R., and Rabin, T. 2005. Secure computation without authentication. Advances in Cryptology: Proceedings of Crypto 2005. (ed. Shoup, V.), Vol. LNCS 3621. Springer-Verlag.

Blake-Wilson, S., and Menezes, A. 1999. Authenticated Diffie-Hellman key agreement protocols. Selected Areas in Cryptography (SAC) '98. In: Tavares, S., and Meijer, H., (ed.) Vol. LNCS 1556. Springer-Verlag.

Blake-Wilson, S., Johnson, D., and Menezes, A. 1997. Key agreement protocols and their security analysis. Cryptography and Coding: 6th IMA International Conference Cirencester. In: Darnell, M., (ed.) Vol. LNCS 1355, Springer-Verlag.

Chaum, D. 2006. Distributed communication security systems, U.S. Patent Application Publication 2006/0218636 A1: (September 28), 1–19.

Cremers, C. J. F. Pascal Lafourcade, and Philippe Nadeau 2009. Comparing state spaces in automatic security protocol analysis. Formal to Practical Security. In: Cortier, V., et al., (ed.), vol. LNCS 5458, Springer-Verlag.

Cyber Defense Lab Animation of Chaum's protocol for detecting a man-in-the-middle http://youtu.be/SKQQiPtmmJk (accessed 21 February 2015).

Dierks, T. and Rescorla, E. (August 2008). The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, Internet Engineering Task Force 1–104.

Doghmi, S. F., Guttman, J. D., and Thayer, F. J., March 2007. Searching for shapes in cryptographic protocols. International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). In: Grumberg and Huth, (ed.) vol. LNCS 4424, Springer-Verlag, Extended version at http://eprint.iacr.org/2006/435.

Drago, I., Mellia, M., Munafò, M. M., Sperotto, A., Sadre, R., and Pras, A. 2012. Inside Dropbox: Understanding personal cloud storage services, In: *Proceedings of the 2012 ACM Conference on Internet Measurement (IMC '12).* Boston, MA ACM.

Dyn Research. 2013. The new threat: Targeted Internet traffic misdirection, http://research.dyn.com/2013/11/mitm-internet-hijacking/ (accessed 25 January 2015).

Johnston, A. M., Gemmell, P. S. 2002. Authenticated key exchange provably secure against man-in-the-middle attack. *Journal of Cryptology* 15(2):139–148.

Langley, A. (April 6 2003). ImperialViolet: Pability Python, http://www.imperialviolet.org/2003/04/06/capability-python.html (accessed 22 February 2015) (April 6, 2003).

Meadows, C. 2003. Formal methods for cryptographic protocol analysis: Emerging issues and trends. IEEE Journal on Selected Areas in Communications 44–54.

Newton, W. (December 2010). Chaum's protocol for detecting man-in-the middle: Explanation and discussion, MS Thesis, CSEE Dept., University of Maryland, Baltimore County, pp. 1–63.

Oracle, Java cryptography architecture standard algorithm name documentation for Java platform standard edition 7. http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html (accessed 1-27-15).

Oracle, Package javax.crypto, http://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html (accessed 1 January 2015).

Rescorla, E. 2001. SSL and TLS: Designing and building secure systems. Boston, MA: Addison-Wesley.

Rescorla, E. (June 1999). Diffie-Hellman Key Agreement Method, RFC 2631, Internet Engineering Task Force pp. 1–13.

Rivest, R. L., Shamir, A. (April 1984). How to expose an eavesdropper, *Communications of the ACM* 27(4):393–395.

Seymour, J. (January 30, 2013). Implementation and evaluation of various man-in-the-middle detection protocols: Detecting an eavesdropper of instant messages using minimal assumptions, Course paper, CMSC-644 Information Assurance, CSEE Dept., University of Maryland, Baltimore County, pp. 1–9.

TheNextWeb.com, Lenovo caught installing adware on new computers, http://thenextweb.com/insider/2015/02/19/lenovo-caught-installing-adware-new-computers/ (accessed 21 February 2015).

U.S. Patent Office. Prosecution history for U.S. Patent Application 11/388,520 by David Chaum, pp. 1–48.

Vaudenay, S. 1995. Secure communications over insecure channels based on short authenticated strings. Advances in Cryptology: CRYPTO 2005. In: Shoup, V., (ed.) vol. LNCS 3621, Springer-Verlag.

Wikipedia, Interlock Protocol, http://en.wikipedia.ord/wiki/Interlock_protocol (accessed 24 January 2015).

Wikipedia, Zfone, https://en.wikipedia.org/wiki/Zfone (accessed 12 April 2016).

Wilcox-O'Hearn, Z. (March 31, 2003). Defense against middleperson attacks. https://web.archive.org/web/20030403000153/ http://zooko.com/defense_against_middleperson_attacks.html (accessed 22 February 2015).

Zimmermann, P., Johnston, A., Ed., Avaya, Callas, J. (April 2011). ZRTP: Media path key agreement for unicast secure RTP, RFC 6189, Internet Engineering Task Force, pp. 1–115.