Ceremony Design and Analysis

Carl Ellison

Microsoft Corporation, One Microsoft Way, Redmond WA 98052
cme@microsoft.com

Abstract. The concept of **ceremony** is introduced as an extension of the concept of **network protocol**, with human nodes alongside computer nodes and with communication links that include UI, human-to-human communication and transfers of physical objects that carry data. What is out-of-band to a protocol is in-band to a ceremony, and therefore subject to design and analysis using variants of the same mature techniques used for the design and analysis of protocols. Ceremonies include all protocols, as well as all applications with a user interface, all workflow and all provisioning scenarios. A secure ceremony is secure against both normal attacks and social engineering. However, some secure protocols imply ceremonies that cannot be made secure.

1 Introduction

It is common for computer professionals to disparage human users as the source of all the flaws that make an excellently designed product malfunction. Some will admit a certain amount of responsibility for this by characterizing the design of a user interface as extremely difficult, but few accept the challenge of designing systems and protocols that produce the correct results when operated by actual human users.

The issue comes up most prominently with security protocols – well designed and thoroughly reviewed – that are fielded and broken. The breaks are usually by **social engineering**. Social engineering exploits human weaknesses to bypass security, doing an end-run around a well designed security protocol. Examples of social engineering include password theft by confidence game techniques and phishing.

The concept of **ceremony**[1] extends the concept of **network protocol** by including human beings as nodes in the network. Ceremonies include all network protocols as a degenerate case, but also all applications with user interfaces and all instances of workflow. For security protocols, the out-of-band provisioning of cryptographic keys becomes a ceremony.

The ceremony concept was chosen to allow us to express distributed system designs that include human beings using terms with which a protocol designer is comfortable. In particular, a ceremony can be designed and analyzed with variants of the mature methods already in use for a network protocol, up through and including formal proofs of correctness.

[1] The term "ceremony" was coined for this purpose by Jesse Walker of Intel Corporation.

This paper is organized as follows. Section 2 defines a ceremony and describes types of analysis one might perform. Section 3 illustrates ceremony analysis with HTTPS and cryptographic e-mail examples. Section 4 introduces ceremony design. Section 5 discusses some well-designed ceremonies. Section 6 presents conclusions and section 7 outlines some possible future work in this area.

2 Definition of Ceremony

A **ceremony** is like a network protocol, some of whose nodes may be human and whose network links are not limited to traditional communications channels. As a result, all network protocols are ceremonies, but ceremonies also encompass all workflow. They also include all applications that have a UI, even if the application is not networked. Such would be a degenerate ceremony of two nodes: a computer application and a human user. Of special interest in the security community are the ceremonies for key management and cryptographic provisioning which can never be pure protocols[2] and therefore must always be ceremonies. Without ceremony definition and analysis, such processes were often not formally specified or analyzed.

A normal protocol operates among two or more nodes and each node has a state encoded as one or more state variables, a set of possible messages and a state machine. For analysis purposes, the entire distributed system can be modeled as the collection of its nodes and the channels between the nodes. The system can be analyzed for normal behavior, absent any faults, from just that model. If nodes are allowed to be reset at arbitrary times, losing their state, and if the channels are allowed to drop, garble or reorder messages, then the distributed system can be analyzed for its fault-tolerance behavior. If channels are allowed to be adversarial – injecting messages, replaying arbitrary messages, constructing messages from parts of old messages, etc. – then the system can be analyzed for security behavior in a hostile environment. If some nodes of the system are allowed to be adversarial as well, then the system can be analyzed for Byzantine security behavior. All of this analytical power is available for ceremonies.

The simple definition of ceremony above hides its complexity. The problem comes with modeling a human node. Like a computer protocol node, the human node has state and a state machine. It receives and emits messages, sometimes via a computer's user interface (UI) and sometimes by human-to-human communication, whether intermediated by computer or not. However, the human input and output of messages is often subject to error, probably because of a translation or filtering mechanism that we have not fully characterized. Similarly, the human state machine is difficult to model. We don't program humans the way we do computers, and when we try to, the attempt usually fails. We must instead learn the human state machine empirically, by observing actual human behavior.

---

[2] A cryptographic protocol requires keys on each end of the communication, for either authentication or integrity protection. Prior to provisioning there are no such keys. Therefore, secure provisioning cannot ever be a cryptographic protocol. Keys must be provisioned over a secure channel, where the security is created out-of-band of the protocol – or by a secure ceremony, where nothing is out-of-band.

3 Example Ceremony Analyses

In this section, we examine two ceremonies that have security flaws. Specifically, they allow man-in-the-middle (MITM) or impersonation attacks even though the protocols they use were carefully designed to avoid MITM attacks. We rely only on the simplest of models of human node behavior to demonstrate these flaws.

3.1 HTTPS with server authentication

HTTPS is based on the SSL protocol [7] or its successor, TLS [4]. That protocol, illustrated in Figure 1 below, is designed to prevent MITM attack.

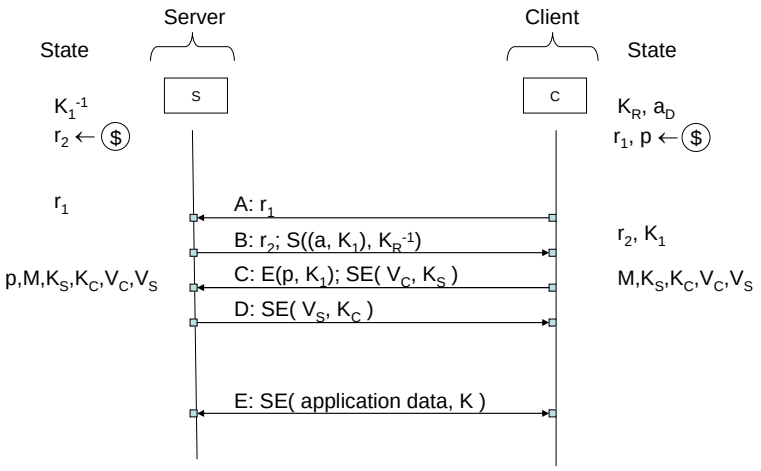

Figure 1: TLS Protocol Flow

This protocol will typically start with negotiation messages not shown in Figure 1, which assumes the most common negotiation outcome – server authentication and use of RSA public keys. The two nodes in the protocol, the client C and the server S, start with certain state information. The client knows a root key, K_R, that certifies address-key pairs, and knows an address, a_D, to which it desires to connect. The client generates two random values, r_1 and p. The server has a private key, K_1^{-1}, and for this protocol generates one random number, r_2.

Looking only at the message fields of cryptographic interest, in message A the client sends r_1 to the server. The server replies with r_2 and its certificate containing the pair (a, K_1) signed by K_R^{-1}. From this information, the client learns r_2 and K_1. It

verifies the certificate against K_R and compares the certified value a, to which K_1 is bound, to a_D. If the certificate fails to verify or if $a \neq a_D$, then the client aborts the connection. Continuing past this point establishes that the cryptographic channel terminating in K_1 is the one the client desires.

The client then uses its value, p, the pre-master secret, and the random values r_1 and r_2 to compute the master secret, M, and from M, to compute session keys for client and server, K_C and K_S, and to compute verification values, V_C and V_S. In message C, the client sends p, encrypted under K_1, followed by V_C encrypted and integrity protected by the session key(s) K_S (symmetric keys used for normal application data transmissions).

From message C, the server learns p, only if the server holds K_1^{-1} – that is, only if the server is the one that can terminate the desired cryptographic channel. From p, r_1 and r_2, the server computes M and the values derived from M. It verifies the client's message, $SE(V_C, K_S)$ and generates its own verification message, $SE(V_S, K_C)$, which it then transmits to the client as message D. When the client receives message D, it knows that it has in fact connected to the correct server, without a man in the middle. There might be other computers in the middle of the network connection, but the cryptographic channel is private between it and S.

With both client and server assured of a private connection, application data is transmitted, encrypted and integrity protected under the session keys thus computed. The application data exchange might start by having the client log in with a user name and password, since the protocol of Figure 1 does not include client authentication.

The HTTPS ceremony, in Figure 2, has 7 nodes instead of 2. It illustrates a successful MITM attack and also includes the out-of-band key management that cannot be achieved by a network protocol but that is needed by the TLS protocol of Figure 1. The details of the TLS handshake protocol are not shown explicitly in Figure 2. They proceed as shown in Figure 1.

In the HTTPS ceremony of Figure 2a, we start by provisioning the client computer, CC, with the PKI root key, K_R. In Figure 1, such provisioning happened out-of-band, but there is nothing out-of-band to a ceremony. For this provisioning we rely on a concatenation of channels. The CA delivers its root public key over a channel to human R. Human R delivers this key over a channel to the user, C. The user C delivers that key to her computer CC. All three channels need to be secure for the ceremony to be secure.

The security of the human-to-human channel in this ceremony depends on pre-existing state in the node C. This state includes both information about R (that C has authorized R's key to serve as a root key on C's computer) and a method for authenticating R over the direct R to C channel. We haven't specified here how that state was established and in particular whether C is justified in making that authorization. Neither have we analyzed the specific authentication ceremony between R and C. Both of those should be analyzed for any implemented ceremony.

The root key distribution in actual systems has many more links than shown in Figure 2a. Keys may go from a CA operator to the software vendor of the browser or operating system, from there to an OEM who packages them with a physical computer that is then sold to a customer. Each of these physical or person-to-person transactions is a valid ceremony message and subject to analysis. The reader is

encouraged to analyze selected real key distribution ceremonies, using familiar techniques from the analysis of network protocols.

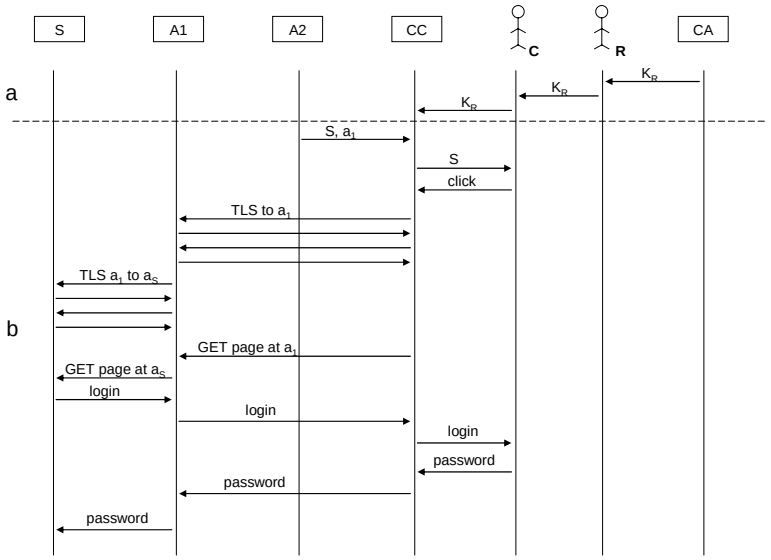

Figure 2: HTTPS Ceremony

In Figure 2b, CC starts with K_R. We assume that the servers S and A1 have private keys and TLS certificates pre-provisioned. We also assume that CC, S and A1 are capable of generating random values for use in TLS.

The attacker has two machines, A1 and A2. A2 delivers an HTML page, not secured, to CC with content intended to induce the human, C, to click further. In particular, that page offers a link to the server, S, by name "S" with associated URL, "a_1". S has address a_S, not a_1, which is the basis of this fraud and is left for the ceremony to detect. The name "S" could be a name or a logo or anything else that C might take as an identifier for the desired server, S.

The first obvious problem in this ceremony is in the message from CC to C[3]. CC offers C not the pair (S, a_1), but rather just S. In many browsers and some e-mail clients, the URL to which the user connects by clicking on S might be displayed – in the window chrome or in a pop-up dialog box. If a_1 is not displayed, then we don't need to ask about the model of the state machine of user node C. If C is not given the datum a_1, then C cannot act on that datum. However, even if C is given that datum there is no guarantee that C knows what decision to make as a result of seeing the

[3] The "message" from CC to C is displayed as part of a normal user interface (UI). However, in ceremony terms such a display is a message. Similarly, the mouse click by which the user interacts with that displayed page is, in ceremony terms, a message from C to CC.

URL, a_1. Will C have been provisioned ahead of time with the association between S and a_S? That provisioning process is not shown in Figure 2. If we add it, where is the user going to store that association and how? It will almost certainly have to be in the user's memory. Can users remember such associations? How many can they remember? Clearly, a user can remember that www.microsoft.com is associated with Microsoft Corporation, but that assumption cannot apply for all possible URLs.

To address the general problem of what to show the user in order to get a correct decision, we need research into how users make security decisions from information presented in a UI. However, just following the ceremony as defined in Figure 2, we don't need that extra research. C was given only S and decides to click on it. At that point the success of the attack is almost inevitable. CC is not in a position to detect the fraud. It was directed by the user, C, to set up a TLS connection with the server at URL a_1. It does that flawlessly and securely. If the fraud is to be detected, it must be detected by the user, C, someplace else in the ceremony.

The next (and last) opportunity for C to detect this fraud is at the login message, after both TLS sessions have been established and the requested page (a login page) has been returned. This page is marked as "secure" (in popular parlance). It has a locked padlock and maybe a change in colors in the window chrome. It also has an associated TLS certificate.

In the discussion of TLS in Figure 1, the TLS certificate was described as only a binding between the domain name address, a_1, and the server key, K_1. There is much more in a certificate. In addition to the domain name address, there is also a corporate legal name and maybe address. If the user, C, were to compare that name (and address) to the legal name and address for S, then the user could detect the fraud before entering her password and returning it to A1. This assumes the user knows the legal name and address for S. The provisioning of that information to the user needs to be part of the ceremony design. Even given that provisioning, the ceremony is flawed. The message from CC to C does not include the TLS certificate. In typical browsers that certificate is available if the user is sufficiently expert to find it and interpret its fields but it is unintelligible to the non-expert.

Even for a user well versed in X.509, the certificate might not be useful in detecting fraud. There are legitimate companies that hire web portal companies to host shopping sites or other TLS-secured sites. Those portal companies will purchase a certificate using their own name, not the name of the company whose page they are hosting. So, a certificate issued to some name other than S might not be fraudulent.

This discussion assumes that S was a textual name and the same name by which the company is registered legally. What if S is a brand name or a logo. It was a clickable entity on an HTML page and could easily have been a picture of the logo or even of a product. We do not normally have company logos in a TLS certificate much less images of products. Among other things, it is not clear that the issuer of a TLS certificate is an authority on company logos. That authority might be a trademark office, not a traditional commercial CA. We can imagine certificates issued by a logo authority that bind logos to server keys or to URLs, but that would be a change to the ceremony in Figure 2.

It is left as an exercise for the reader to secure the ceremony in Figure 2, perhaps after reading section 4.

3.2 Cryptographic E-mail Ceremonies

We consider cryptographic e-mail in four phases: (1) certification, (2) distribution of root keys, (3) receiving signed e-mail, (4) sending encrypted e-mail.

3.2.1 Certification

The description of cryptographic e-mail ceremonies must start with a discussion of the process of getting an e-mail certificate and delivering the key by which that certificate is validated. There are three human parties to these ceremonies. Using the terms common in the PKI community, there are:

1. Alice, the relying party (RP) with computer AC,
2. Bob, the end entity (EE) with computer BC; and
3. Carol, the CA with computer CC.

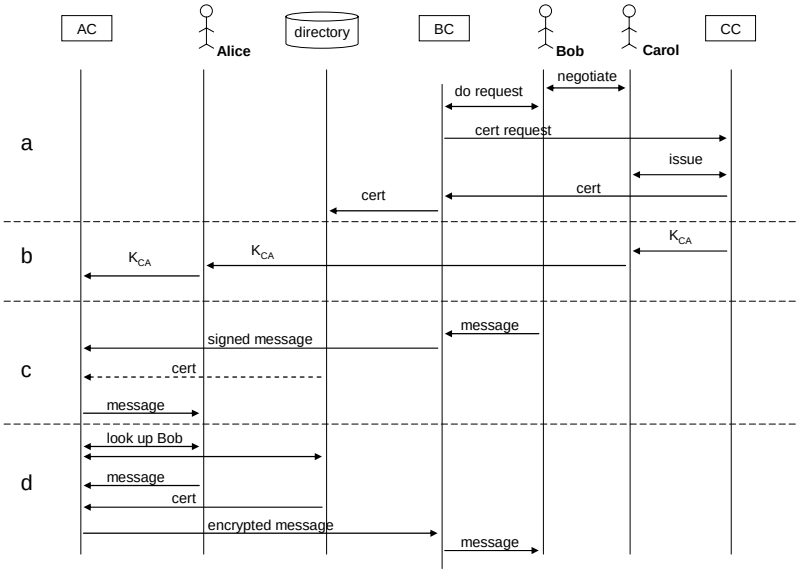

Figure 3: Cryptographic E-mail Ceremonies

The ceremony of Figure 3a is the process by which Bob gets a certificate. This must start with a human-to-human sub-ceremony by which Bob convinces Carol to issue him a certificate. We assume that Bob is successful in this negotiation but we do not attest to the security of that negotiation. The possible ceremony flaws in the negotiation between Bob and Carol are left to the reader to enumerate.

Bob then interacts with his computer, BC, to cause it to generate a certificate request to Carol's computer, CC. The protocol used will differ depending on whether Bob is asking for an X.509 S/MIME certificate pair [11], a PGP key signature [1], or some other form. We assume Bob provides his public key(s) in this request message.

There is a difference between PGP and S/MIME that may be of interest later. If Bob is asking for a PGP certificate, then he supplies both his common name and his e-mail address for that certificate: in this case, "Bob xyzzy042@hotmail.com". If Bob is asking for an S/MIME certificate pair (one for signed mail and one for encrypted mail), then he supplies his e-mail address, but Carol assigns his common name. We assume that Carol already has a certificate issued to the name "Bob", so she gives Bob the name "Bob 587".

In all cases, Carol then instructs her computer, CC, (the CA) to issue the requested certificate and send that certificate to Bob's computer (e.g., via his e-mail address).

For the sake of illustration, we assume that Bob's computer then uploads the newly received certificate to a directory.

3.2.2 Distribution of root keys

The ceremony of Figure 3b can occur before or after the issuance of Bob's certificate. This is the ceremony by which Alice receives the CA key, K_{CA}, from Carol and depends on the security of human-to-human channels. As noted in section 3.1, the ceremony actually used for distribution of root keys is more complex than is shown and the reader has been encouraged to analyze it.

3.2.3 Receipt of signed e-mail

Figure 3c shows the normal signed e-mail ceremony. Bob composes a message to Alice, asking her to do something. Bob's computer digitally signs that e-mail with Bob's private key, K_B^{-1}. Bob's computer, BC, sends that e-mail to Alice's computer, AC, possibly including Bob's S/MIME signature certificate or PGP signed key (binding Bob's common name and e-mail address to his public key, K_B). Bob's computer might also transmit an S/MIME encryption certificate (binding Bob's common name and e-mail address to his encryption public key, K_{BE}). Alternatively, Bob's computer might not transmit any certificates with the message and Alice's computer might then fetch the certificate it needs from a directory.

These details of certificate handling are irrelevant to the security of the signed message. The message has an associated cryptographic channel that is independent of the actual delivery ceremony for either the message or its associated certificate.

AC verifies the message's signature using KB, and verifies the certificate for KB, thus having established that the message came from "Bob xyzzy042@hotmail.com" or "Bob 587; xyzzy042@hotmail.com".

So far, the ceremony is well analyzed and we can assume that it is invulnerable to impersonation. The new element that still needs to be analyzed is the last message in the ceremony, from AC to Alice.

The message from AC to Alice is delivered as a body of text with header information depicting the results of the signature validation. For example, there might be an explicit error warning if the signature or the certificate (chain) did not validate. The absence of that warning constitutes communication to Alice that the cryptography was correct. But, the origin of the message must be displayed to Alice in some form. This display depends on the design and construction of the mail agent. The information it has available for display includes:

1. the common name, "Bob" or "Bob 587", from the certificate

2. the e-mail address xyzzy042@hotmail.com, from the certificate
3. the e-mail address provided in the mail header [3]
4. the common name provided in the mail header

It is common for an e-mail agent to display both message header information and information from the signature certificate, as shown for example in Figure 4 below.

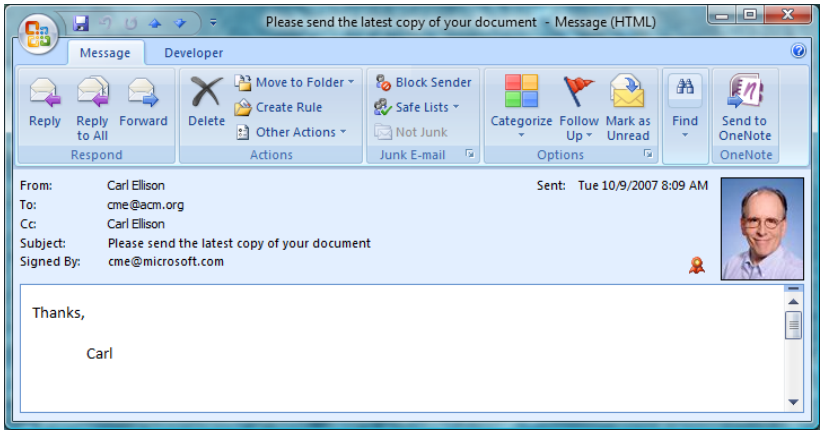

Figure 4: Sample signed message

In this message, the From: line is taken from the mail header. The Signed By: line is taken from an S/MIME certificate. The gold and red seal icon indicates that the signature was verified correctly. The picture of the sender is supplied by the personal contact list as indexed by the message header From: address.

How Alice evaluates the message from Bob is a topic needing empirical study. Her process will involve comparing what is displayed for her to what she has in her memory. She could compare:

1. the From: address
2. the picture
3. the Signed By: address
4. text in the message itself (phraseology, word choice, …)
5. etc.

Our job, analyzing the security of this ceremony, is to predict the probability of an incorrect choice when the message was actually sent by an imposter, Mallet, claiming to be Bob. Mallet supplies the From: address and the message text. For e-mail agents that show a picture based on the From: address, Mallet's message would be displayed with Bob's picture. The Signed By: address would show Mallet's e-mail address, rather than Bob's[4]. Mallet knows that Bob has address xyzzy042@hotmail.com, so

[4] Microsoft Outlook, from which this image was captured, warns the user if the Signed By: address differs from the From: address – so this potential ceremony attack is avoided. Some other mail agent might not give that warning and therefore might allow this attack.

Mallet could get the e-mail address xyzy042@hotmail.com, and that would be the address shown in the Signed By: line.

If Alice were a computer (a robot, perhaps), the fact that the Signed By: address differed in one character from the correct one would make the header incorrect and she/it would detect fraud.

What would a real human do?

Has the node, Alice, been provisioned securely with Bob's e-mail address in the first place? That provisioning was not included in the ceremony. If she has never seen the address xyzzy042@hotmail.com or has not successfully associated that address with Bob in her memory, then the work Mallet did to get the address xyzy042@hotmail.com was unnecessary. Alice the robot might reject the message as unverifiable if she/it did not have the Signed By address in memory. Based on observed e-mail behavior among real users, we predict that Alice the human will act on incomplete information. In fact, there is an aphorism praising the ability to act quickly, on incomplete information: "he who hesitates is lost". One never hears: "he who fails to hesitate is lost".

This behavior is subject to verification by well controlled experiment, but we predict that if Alice does not have Bob's e-mail address in her memory, she will accept Mallet's signed message as coming from Bob. All the other information displayed in the header calls Bob to mind. She has no referent for the Signed By: address, so she ignores it. She might ignore it even if she does know Bob's e-mail address, given that all the other evidence presented to her is internally consistent and calls Bob to her mind. The fact that the one inconsistent datum happens to be the only secured one does not raise an alarm for Alice.

This process by human beings of doing fuzzy comparisons – ignoring comparisons that cannot be completed because information is lacking or because so much other evidence suggests a different outcome – suggests a Boolean expression evaluation algorithm that does not use logical AND to connect the various component comparisons but rather uses some weighted threshold function and that stops the evaluation process as soon as the threshold is satisfied. Again, we need research to discover this algorithm definitively, but this process has been observed in practice and has been documented under the name "The John Wilson Problem" [6]. We also have anecdotal evidence that the weights in this presumed threshold function are not uniform – that, for example, a picture carries more weight than a text string. We also know that information that confirms an expectation or belief is accepted readily. So, if Alice were expecting a message from Bob when Mallet's impersonation of Bob arrived, she might be more inclined to accept it than she otherwise would.

If the threshold fuzzy comparison algorithm above is correct, then the construction of a proper message from AC to Alice would require us to eliminate any information that has not been secured. Extra information (like the From: address) serves to pollute that presumed threshold function and reduce the security of the ceremony.

3.2.4 The Directory Dream

X.500 [2, 13] is a technical specification for a tree-structured directory. In the late 1980s, while it was being formalized, the assumption was that there would be a global X.500 directory as soon as it could be deployed – surely in a matter of a year at most.

Any day now, the dream went, we will be able to look up the e-mail address of that old friend of ours that we've lost track of.

The global X.500 directory has never been realized and probably never will be but the dream lives on in the form of slightly less global directories of user information.

Microsoft offers corporate customers a Global Address List (GAL) [10] in Active Directory (AD) [9]. Using the GAL, one can select any e-mail recipient by common name rather than e-mail address and can view various other attributes about each person – such as telephone extension, mail stop, etc. The user object in AD can hold a user's S/MIME encryption certificate, so that an e-mail sender can send encrypted e-mail to any user registered in the GAL and capable of receiving S/MIME e-mail.

Shortly after PGP [1] was deployed, several global PGP key directories were made available[5]. Eventually, pgp.com hosted an official key directory[6]. The PGP application was made configurable to automatically consult a key server when looking up the public key for an e-mail address, when that key is not found in the local cache of keys.

None of these designs and implementations included ceremony analysis. They concentrated on the computer network protocols and associated data structures. The result is directories of so many people that name collisions occur – if not exact collisions, then collisions after the filtering done by the human fuzzy comparison algorithm. These directories are like a large city phone book. You can select a subset of the listed entities, based on some search characteristics, but you are left with the task of discovering your intended one individual from within that subset through information not in the directory. This can lead to security flaws if the fuzzy threshold algorithm is a correct model of human behavior because under that model the human will not do the extra work to securely establish a single entry as the intended one.

3.2.5 Transmission of encrypted e-mail

The ceremony of Figure 3d directly suffers the directory access fuzzy comparison problem. The ceremony starts with Alice consulting the directory looking for Bob's e-mail address. If the directory is small enough (e.g., Alice's own personal directory), then there may be no collisions when she looks up Bob, but as the directory gets large, collisions become inevitable under the fuzzy comparison algorithm.

This was the original observation that became known as the "John Wilson Problem" [6]. In that environment, there was a corporate directory of 70,000 entries. The IT department was very careful to make sure that all common names in that directory were unique. Because more than just the common name was displayed to a user looking for an e-mail addressee (information such as phone number and mail stop), these display lines were not only unique but sparse IDs. Yet, John Wilson (one of 8) kept getting mail that should have gone to one of the other John Wilsons. The only way to explain that repeating problem is that when the human users did fuzzy comparisons that produced a collision, instead of halting the e-mail process and looking for a way to resolve the ambiguity, they picked one of the colliding candidates and sent the mail to that person. These e-mails were not spam. They were legitimate corporate e-mail. Some of them contained very confidential information.

---

[5] http://pgp.mit.edu/ http://pgp.nic.ad.jp/ http://pgp.openpkg.org/

[6] http://keyserver.pgp.com/vkd/GetWelcomeScreen.event

Given that apparent behavior of Alice in the ceremony of Figure 3d, the probability of error (defined as delivering the e-mail plaintext to someone who should not see it) increases as the probability of collision does – and that increases with the size of the directory. The dream of a world-wide directory of all the e-mail recipients in the world would under this logic be a security flaw. Even with small directories of a few thousand, the probability of information leakage through improper addressee selection is much higher than the probability of leakage through cryptanalysis[7].

After Alice has selected the wrong Bob from the directory, there is one more chance for detection of the problem. The ceremony of Figure 3d consults another directory (shown as the same one, in the figure) to retrieve Bob's encryption certificate[8]. If that directory's set of encryption certificates is large enough to have the wrong Bob's certificate on file, then it provides no block to the error and the sensitive plaintext is encrypted for and delivered to the wrong Bob. It is only if that directory is small enough that there are likely to be no certificates for the incorrect Bobs of the world that this step in the ceremony could possibly block Alice's mistake.

That sentence calls it Alice's mistake, but the mistake is really on the part of the ceremony designer. Alice is just behaving like a human being rather than a robot. She is not to blame. We just haven't designed the e-mail ceremony to be secure.

4 Ceremony Design

> ``Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)" [8]

At least three things set ceremonies apart from network protocols:

1. There is nothing out-of-band to a ceremony. A ceremony design is more complete than a protocol design, but the designer does not have the luxury of just assuming that magic happens out of band to make the ceremony succeed.
2. Some of the "network connections" in a ceremony can be human-computer (via UI), human-human (face-to-face, phone calls, etc.) or physical (sneaker net (including use of a smartcard or USB token), purchase of a computer with software and data pre-installed …).

[7] The John Wilson referred to here once found that 1% of the e-mail in his inbox for that day had been intended for one of the other John Wilsons. This is only one sample, but is so large compared to cryptographic error rates that it implies that this vulnerability dominates.

[8] Alternatively, Alice could be using Identity Based Encryption, letting Bob consult an IBE server instead of Alice's consulting a directory for a certificate.

3. Human nodes have different capabilities from computer nodes and must be designed appropriately, especially in view of the fuzzy comparison model. Their tamper resistance and ability to keep secrets are also quite different.

Those differences aside, the design process for a ceremony is the same as the process for a network protocol. Each node in the ceremony has:

1. state, held in the node's memory in one or more locations
2. secrets, protected by tamper resistance and subject to access control
3. a state machine
4. input messages that are parsed and sometimes pre-processed, including
    a. messages from other nodes
    b. events (like a timer or, within a human node, a "desire")
5. for each (input message, state) pair:
    a. output messages
    b. changes in state
6. service response times and communication bandwidth
7. probability of processing errors
8. probability of node death or loss of memory

State machines need to be specified for every node in the ceremony. One of the nodes will, from its home state or from some external event (like a timer or a desire), initiate a message cascade. As with protocols, we can look for unrealizable conditions (a reply that is generated before the message that initiates it), deadlock (nodes waiting for messages from each other), race conditions, etc. We can do normal performance analyses. We can analyze a ceremony for fault/disaster tolerance. We can do normal security analyses of the ceremony and of secrets stored in each node's memory. We can run simulations and evaluate them against a set of expectations. We can do formal modeling and attempt to prove properties of the modeled ceremony. All of the techniques that we use for network protocol design should be available for ceremony design.

4.1 Modeling the Human Nodes

The major effort yet to be accomplished in the field of ceremony design and analysis is the modeling of the memory, state machines and processing performed by human nodes. The definitive research in this area should be conducted by experimental psychologists or cognitive scientists. However, as was shown by the example of section 3.1, it is not always necessary to have done that research. The HTTPS ceremony example assumed that a human node's state machine is as easy to specify and describe as that of a computer node. The flaw in the ceremony stood out because information necessary for the decision that the (human) node was asked to make was withheld from the node. No node, whether computer or human, could be expected to make a correct decision if not given the data needed for that decision.

Lacking the results of the definitive scientific research mentioned above, we can use lessons learned from *post mortem* analyses of successful attacks or other

ceremony errors. The cryptographic e-mail analysis of section 3.2 is based on such data. We can also do experiments testing actual human users operating within candidate ceremonies.

When we do those experiments, we can model the human node as a probabilistic state machine – one whose state variable has a probability distribution rather than a definite value and whose state transitions are changes in that probability distribution. Such a state machine would have a probability distribution of output messages as well. That kind of model is natural for the design and analysis of fault/disaster tolerant distributed systems, but should produce interesting results in other analyses of protocols and ceremonies.

A more normal model would be that of a deterministic, desired state machine with a probability of error. As long as the probability is low (e.g., in some cryptographic protocols: 2^{-128}), we can treat it as 0. If the probability is high (e.g., 0.01), we know we can't use the deterministic model.

If we need deterministic behavior when the error rate is too high, then we need to modify the ceremony around that node. This might involve a re-design of the UI (changing a single input message) or it might involve adding rounds of messages that test independent conditions in an attempt to reduce the error probability to the product of the individual error probabilities. Of course, multiple UI rounds will cause significant user annoyance and will probably be an inacceptable solution. A best approach would be to find a design that keeps the human from making any decision with a high error rate, while still achieving the properties that the ceremony needs.

To achieve deterministic behavior, we might consider installing a program, for example by way of a user's manual. In the 1950s this apparently succeeded, but starting at least in the 1970s one heard cries of "RTFM" and by the 1980s software was starting to be delivered with no user's manual – with only on-line help and reliance on some general user familiarity with the UI paradigm.

With some high security ceremonies, the US Department of Defense and certain corporations use legal contracts that mandate user behavior, especially in the handling of classified information. Presumably these contracts are more effective forms of programming of human nodes than a user's manual, but we would need to study the error rate of the application of such programs.

4.2 Meaningful IDs

One of the characteristics of a human node, as noted in section 3.2.3, is that humans process visual information and make comparisons with some algorithm unlike what we would envision programming. The model of human behavior used in section 3.2.3 is that of a fuzzy comparison.

Designing a ceremony message from computer to human (a UI screen) on which the human needs to make a security decision requires presenting the information that is needed for that decision, and doing so in a way that the human fuzzy comparison process won't invalidate. In some of these cases, the human is making a decision based on the identity of a source (the sender of an e-mail, the author of some software or the source of some web page, etc.). For this purpose, the human needs to be

presented with an ID for that source. We capture the requirements of that ID in the term, **meaningful ID.**

Definition: *A **meaningful ID** for use by some human being is an identifier that calls to that human user's mind the correct identified entity.*

We don't have a full model for the human ID comparison algorithm, but we can list some possible requirements for meaningful IDs based on what we do know.

1. Calling a correct entity to the human's mind implies that the human being has a body of memories about the correct entity. If there are no such memories, then no ID can be a meaningful ID.
2. What works to call those memories to one observer's mind may not work for another observer. Therefore, a meaningful ID is in general a function of both the identified entity and the person looking at it.
3. The meaningful ID needs to attract attention or be presented without distracting clutter, so that it has the opportunity to be perceived.
4. When a security decision is to be made, the meaningful ID(s) must be derived in a secure manner and competing IDs that an attacker could introduce at-will should not be displayed.

Observation #2 above makes it extremely difficult to build a global directory of meaningful IDs or to issue ID certificates that would be useful globally. Most likely, meaningful IDs will have to be held in a personal directory where the ID is chosen by (perhaps composed by) the person who owns that directory.

IDs that fail in general as meaningful IDs include a social security number (because the relying party (RP) is not likely to know it), a common name (because in any sizeable population the probability of collision is too high for security purposes), a GUID, a Windows SID, an e-mail address.

IDs that might in general be meaningful include a nickname chosen by the RP for the identified entity or a picture of the person (especially one taken by the RP).

By the definition above, however, an ID is meaningful if it works with a low enough error rate and not if it doesn't. That means that what is meaningful for one person might not be meaningful for another.

5 Some Better Ceremonies

The examples of section 3 concentrated on flaws in the ceremonies implied by existing, well-analyzed security protocols. There are examples of ceremonies not designed with the term "ceremony" in mind but with good results.

Cryptographic modules sold to banks for ATM card PIN cryptography include very well designed key management ceremonies. These are based on physical tokens such as smartcards. The human carriers of those smartcards are instructed in their care, but the ceremony is designed such that an error on one human's part is not catastrophic. The ceremony requires agreement by multiple parties before a human action is acted upon. Many other designs use physical tokens this way.

Some systems achieve a security result without hardware tokens through ceremonies using human strengths. The Clipper phone of the early 1990s used human voice recognition (both natural and superior to computer methods) to authenticate the party on the other end of the line and used a display of meaningless hex digits (computed from the hash of the keys used to set up the connection) to verify that there was no man in the middle. Each of the two parties was led through the process of reading half of those digits to the other party. The more digits one read, the lower the probability that an attacker could succeed with a MITM attack.

6 Conclusions

Security analysis of a node in a ceremony is the same for both human and computer nodes. We have to ask the question: "With what probability will an attacker be able to fool the node into making an incorrect decision?" For a computer node and a cryptographic protocol, that probability might be the probability of the attacker's guessing a high entropy secret – e.g., a value like 2^{-128}. For a human node in a ceremony, the probability of an incorrect single decision might be more like 0.01, given current UI designs.

For ceremonies that were implied rather than designed (as in the examples of section 3), simple ceremony analyses can yield results. As ceremony design becomes commonplace, the ceremony analyses will need to become more sophisticated.

There are two areas where current *ad hoc* ceremonies tend to be weak:

1. When portions of the ceremony were considered out-of-band, they are often not specified by the designer and therefore left open for arbitrary people to satisfy in arbitrary ways. Some if not most of these processes typically turn out to be weaknesses, especially in security ceremonies.
2. The human nodes in a ceremony are often given decisions to make that require super-human abilities and knowledge. The human sometimes plays the role of the *deus ex machina* that makes the ceremony hang together. Recognition of the peculiar behavior of human nodes (limited memory, probabilistic memory accesses, fuzzy comparisons, etc.) leads to discovery of flaws during ceremony analysis and could lead to radically different ceremony designs.

7 Future Work

Ceremony design and analysis is a relatively new field with much unexplored territory. There have been some early attempts at ceremony design [5, 12], but many more such attempts are needed.

The largest body of unexplored territory in Ceremony Analysis, from what we can see today, is the construction of a more sophisticated model of human node behavior, backed up by well-designed scientific research.

Parallel with that activity, it is possible to do scientifically valid user studies to populate a probabilistic state machine model for human nodes. This kind of research

can be conducted by computer scientists using the physical facilities already in place for usability studies. Such empirical studies are useful for testing alternative fully designed ceremonies. They may or may not give insight into the more general user modeling problem.

References

1. Callas, Donnerhacke, Finney, Thayer, "OpenPGP Message Format", IETF RFC2440, November 1998. http://www.ietf.org/rfc/rfc2440.txt

2. Chadwick, D W "Understanding X.500 - The Directory" (1994, 1996). http://sec.cs.kent.ac.uk/x500book/

3. David Crocker, "Standard for the Format of ARPA Internet Text Messages", IETF RFC822, August 1982. http://www.ietf.org/rfc/rfc0822.txt

4. T. Dierks, C. Allen, "The TLS Protocol", RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt

5. Dohrmann, Ellison, ``Public-key Support for Collaborative Groups", 1st Annual PKI Research Workshop, April 2002. http://www.cs.dartmouth.edu/~pki02/Dohrmann/

6. Carl Ellison, ``Improvements on Conventional PKI Wisdom", 1st Annual PKI Research Workshop, April 2002. http://www.cs.dartmouth.edu/~pki02/Ellison/

7. A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

8. Charlie Kaufman, Radia Perlman, Mike Speciner, *Network Security / PRIVATE Communication in a PUBLIC World*, Prentice Hall 2002; p.237

9. Microsoft Corporation, "Directories", http://msdn2.microsoft.com/en-us/library/aa139677.aspx

10. Microsoft Corporation, "Global Address List attribute", http://msdn2.microsoft.com/en-us/library/ms675720.aspx

11. Ramsdell, "S/MIME Version 3 Message Specification", IETF RFC2633, June 1999. http://www.ietf.org/rfc/rfc2633.txt

12. UPnP Security Working Group, "UPnP™ Security Ceremonies Design Document", October 3, 2003. http://upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf

13. "The Directory: Overview of concepts, models and services", ISO/IEC 9594-1.