CHAPTER 7

Key Continuity Management

Key Continuity Management is a promising technique for realizing the ease of anonymous end-to-end encryption while still alerting the user to many kinds of active attacks. By eliminating reliance on trusted third parties, the cost and usability barriers of deploying cryptographic protection are dramatically reduced. This chapter introduces KCM, presents patterns for its use, and presents the results of a user test designed to test KCM's protection against likely spoofing attacks.

7.1 Key Continuity Management

Key Continuity Management (KCM) is an attractive approach for using PKI technology without third-party certification. The model was introduced with SSH [Ylo96], and formally named by Gutmann [Gut04b].

7.1.1 KCM in a nutshell

KCM is based on two observations made earlier in previous chapter. The first is the impossibility of having a top-down identification infrastructure that correctly presents globally meaningful names. The second is the observation that a man-in-the-middle attack can be detected after the first exchange between two trusting parties by observing changes in public keys.

Whereas systems based on PKI rely on trusted third-parties to certify the legitimacy of keys, KCM allows users to decide for themselves which keys to accept, which to be suspicious of, and which to reject. The idea is to flag the key as "new" to the user the first time the key is seen paired with a specific identifier (in this case, an email address). After this initial presentation, the user is then warned if the pairing between the identifier and the key changes at some point in the future.

Applications that implement KCM can ignore both X.509 distinguished names and certification chains, and instead become directly aware of the public key that each certificate contains. Alter-

natively, KCM can be used to augment PKI-based applications in cases where there is no registered trust root for a specific certificate.

When KCM is used with a client/server based system, the client remember a server's public key the first time that the key is presented. Subsequent uses of that same key the require no user intervention. Users are be notified if the server's key changed. Using the taxonomy presented in Section 6.2.6, KCM combines the Anarchy Model with an appreciation for certificate history.[1]

If programs that process certificates implement KCM, then the programs that provide certificates no longer need to be equipped with certificates that are certified by a trusted third party: instead, self-signed certificates can be automatically created when the software is run for the first time after installation. This is, in fact, what may SSH implementations currently do.

The primary advantage of KCM is that it is easy to set up. The primary disadvantage of such a system is that it offers no protection against an attacker that can presents a legitimate key to the user upon that user's first interaction with the attacker's service. In such a case, a KCM-based system will accept the attacker's key and, in fact, warn the user when the key a for the legitimate service is presented instead.

Keys used for KCM can be quite simple: it may be advantageous for key certificates to *intentionally exclude* information such as IP address, DNS name, legal names, or other information. After all, secure keys are by definition unique. If this information is not in the key, then the information can change and the underlying security system can tolerate such changes. This allows trust to transcend addressing—potentially important for mobile services. The key becomes its own identifier.

7.1.2 Justification for key continuity management

Gutmann argues that "assurance through continuity" is generally more important to users than assurance through absolute identification. Users don't really care about formal legal names, he argues: they simply want services to be consistent, so that they are the same the *next time* as they were the *last time*. The success of McDonalds, he and others argue, is not due to the quality of the restaurant's food, but its consistency. Other examples abound. "Coke is Coke no matter what shape bottle (or can) it's in, or what language the label is in....Continuity is more important than third-party attestation."[Gut04b]

Biometric practitioners have long recognized the difference between assuring continuity of identification vs. absolute identification. Nanavati *et al.* distinguish between verification systems, which answer the question *"Am I who I claim to be?"*, and identification systems, which answer the question *"Who am I?"*[NTN02, p.12] What Nanavati terms *verification systems* are essentially systems that ensure for identification continuity: they ensure that the *body* that now stands before the biometric sensor is the same as the *body* that originally registered. The distinction between bodies and people is discussed in [Gar00, p.65–p.66].

One of the most successful uses of biometrics to date has been stand-alone access control systems.

[1] KCM as described here does not handle issues such as certificate expiration, where there is a desire to have an orderly progression from one certificate to another. One way to envision such a system would be to use the old key to sign the new certificate. This would require some kind of specification or standard, but it could almost certainly be worked within the existing X.509 certificate formats.

These systems contain a database of locally stored *templates*, rather than *identities*, that are allowed access: any individual presenting the appropriate biometric (be it thumbprint or an iris) that matches a stored template is granted access. An alternative approach is to use a biometric to validate a person's identity, then to have a list of identities (e.g. names or public keys) that are allowed access. Such systems have been less successful in the marketplace because they are more difficult to manage, more brittle in operation, and have negative privacy implications.

Another justification for the use of KCM is that it is less brittle in light of modern business relationships than PKI-based alternatives employing third-party certification. For example, the Internet banking service provided by the Massachusetts-based Cambridge Trust Company is authenticated with a certificate belonging to the Metavante Corporation of Milwaukee, Wisconsin. (Figure 7-1) This out-sourcing relationship is not indicated anywhere on the Cambridge Trust web site. The only indication that a customer has is that the "Log In" button on the Cambridge Trust web site links to the URL `https://cib.ibanking-services.com/cib/login.jsp` on a web server operated by Metavante. A security-conscious Internet user might reasonably look at the Metavante web site and decide that they are being subjected to a so-called phishing attack. A KCM system, by contrast, would tell the user the first time they hit the login button that the computer was visiting a new site for the first time—matching the user's expectations. Afterwards, no warning would be generated unless the user was actually subject to an attack.

7.1.3 KCM security vs. traditional CA-based security

One of the primary criticisms of KCM is that it does not offer the same degree of security as traditional certification by trusted Certificate Authorities. This belief implicitly assumes:

- That certification by Certificate Authorities actually provides a high-degree of security and assurance.
- That it is so cheap and easy to obtain third-party certificates that legitimate businesses will in fact obtain them, rather than live with the risk of not using them.

We have observed that neither of these conditions are true. Many phishing sites have been discovered with valid SSL certificates signed by respected Certificate Authorities.[M.04, Col04] In some cases attackers have broken into web sites with valid SSL certificates and set up their attacks in unauthorized directories. But it is also possible for attackers to obtain CA certificates from organizations such as VeriSign with a certified DBA ("Doing Business As") licenses; Mazières writes that he was able to obtain a VeriSign certificate for $440 from VeriSign for a business that was not listed in the phone book and that, for all practical purposes, did not exist.[Maz00, p.16] Meanwhile, the E-Soft survey shows that many organizations running SSL servers do not feel motivated to obtain properly signed SSL certificates from respected CAs.

7.1.4 Examples of KCM

Many systems have been found that implement aspects of the KCM model:

- As previously noted, one of the best examples of KCM is SSH. Under normal operation SSH alerts the first time that a new service is contacted. After that first contact, however, SSH is silent unless the server's key changes.

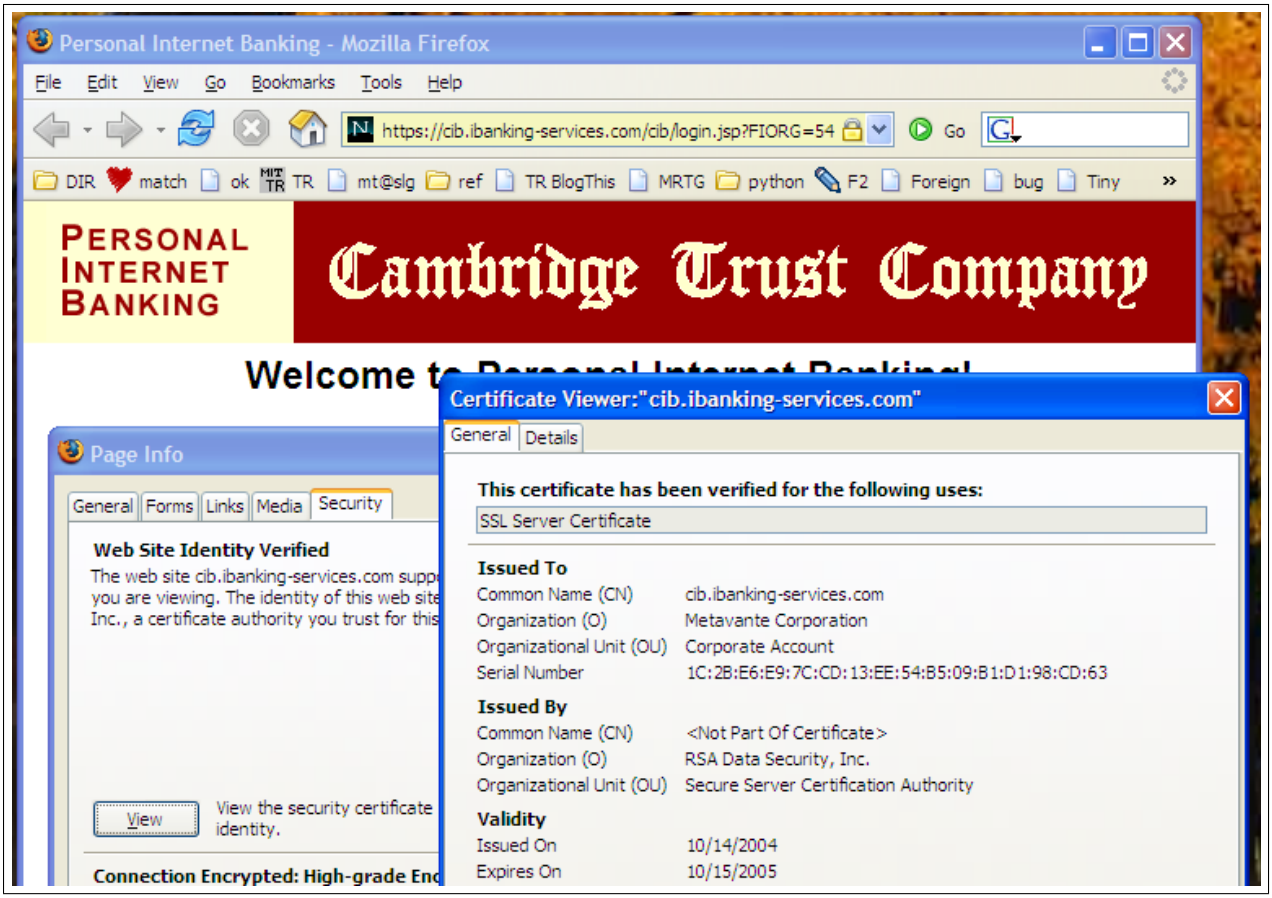


Figure 7-1: The Internet banking service provided by the Massachusetts-based Cambridge Trust Company is authenticated with a certificate belonging to the Metavante Corporation in Milwaukee, Wisconsin.

SSH keys can change for a number of reasons. For example, keys frequently change when a computer's operating system is upgraded and a new version of SSH is installed. When students return to a university in September after a long summer break and are told that the SSH key of their mail server has changed, the change can be easily explained—the sysadmins have upgraded the server over the summer. On the other hand, if a student goes to the Black Hat "hacker" convention in Las Vegas and is alerted that the SSH key of their mail server has changed, the changes probably indicate that someone at the conference is mounting a man-in-the-middle attack over the conference 802.11 Wi-Fi network.

- Current versions of some SSL clients will warn the user when a client connects to an SSL-secured server that is using an X.509 key that is either invalid or signed by an untrusted CA. The user has the choice of trusting the server or not trusting the server. At this point most SSL implementations will abandon all subsequent key checks for the web site, but some clients (such as the client in Apple Mail) will remember the SSL certificate and will alert the user if the remote site changes its certificate at some later point in time.
- Microsoft's S/MIME implementation in Outlook Express and Outlook offers a kind of manual

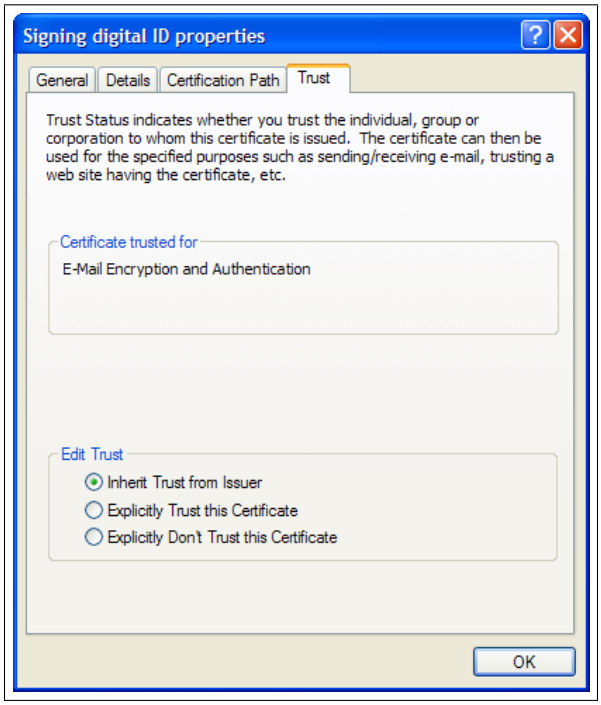


Figure 7-2: Microsoft's S/MIME implementation allows users to explicitly trust or not trust a certificate, bypassing the trust inherited from the certificate's issuer. This is a kind of manual KCM.

KCM in which users who receive S/MIME-signed mail can decide to "Explicitly Trust this Certificate," as shown in Figure 7-2. Certificates can also be explicitly untrusted, even if they are signed by a CA that is trustworthy.

- Stajano and Anderson's "Resurrecting Duckling" security model for ad-hoc networking implements a subset of KCM. In this model, child devices choose to trust the first "mother" device that they see (like a duckling imprinting on its mother). After the initial trust decision is made, the only way to change a device's trust settings is to reset the device, a process that erases all of the device's internal state (like a duckling that is killed and then resurrected).[SA99]
- The 802.11 Wi-Fi client built into MacOS 10.3 will alert the user when a new wireless network is detected and ask the user if the network should be trusted in the future. The operating system remembers the last five trusted networks in the `~/Library/Preferences/com.apple.airport.plist` file. When new networks are trusted, the older networks gracefully age out.

Based on these observations, comments by Gutmann, and the research performed in the course of this dissertation, a proposed set of Rules for Key Continuity Management appears in Figure 7-3.

7.1.5 Disadvantages of KCM

KCM is not without its disadvantages.

In a world where all certificates are actually certified by trusted third parties, the Distinguished

1. Programs that need to provide certificates should automatically generate self-signed certificate C bound to identity I when identity I is first configured.
2. The first time that a self-signed certificate C is received for an identity I, the user should be notified that a new identity has been presented.
3. On each subsequent presentation of that same self-signed certificate C for identity I, the system should indicate that the certificate has not changed. Ideally, the system should track the total number of times that the (C, I) pair has been presented and make this information available to the user in a manner that is inobtrusive but evident.
4. If the user receives a message claiming to be from identity I that is not accompanied by a certificate C, the user should be warned that the identity usually employs digital certificates, but for some reason it is not doing so.
5. If the user receives a message claiming to be from identity I that is accompanied by a new certificate C_2, the user should be warned that the certificate has changed and that an attack may be in progress.
6. The system should visually distinguish unverified KCM identities from identities that have been verified by a trusted third party.

Figure 7-3: Proposed rules for Key Continuity Management

Name on a certificate can be believed to mean something that can be certified. For example, a VeriSign Class 1 Digital ID that claims to be from `marketplace-messages@amazon.co.uk` almost certainly was issued to an individual or organization that had the ability to receive email messages sent to the `marketplace-messages@amazon.co.uk`—after all, this is what VeriSign promises in its Relying Party Agreement.

In a KCM-certified world, the only way for a user to be sure that the holder of a Digital ID has the ability to receive email at a particular address is by sending that Digital ID holder an email message and awaiting an unambiguous reply. For example, one could request a "signed return receipt" as specified in RFC 2634 [Hof99] and implemented in a compliant S/MIME client such as Outlook Express.[2]

In a world with trusted third parties, users—by definition—rely on those parties. In a KCM world the users are mostly on their own—just as SSH users are today. If your laptop tells you that the server's SSH public key has changed, the change might be because somebody has reinstalled the server's operating system. Or the change might be because you are trying to access your server from a wireless "hot spot" at the DEFCON hacker convention and somebody is trying to mount a sophisticated man-in-the-middle attack to steal your username and password. There's really no way to be sure.

[2]Unfortunately, it's also possible for users to employ unreliable techniques such as sending an email message to the address and waiting for a response. In the *Johnny 2* user test several users were spoofed by sending a question and then misinterpreting a message from the Attacker as if it were a response to their question!

7.1.6 The KCM spoofing attacks

As the above analysis illustrates, the primary risk of KCM is the spoofing attack—that is, that an attacker may convince a KCM user that he or she is someone else.

Consider the case in which Alice and Maria are two individuals that are engaged in a long-running dialogue using email clients that implement KCM. Each message that Alice and Maria exchange is digitally signed; their mail clients verify the signatures and check that the (email address, public key) pairing has not changed.

If an attacker wishes to spoof Alice—perhaps to trick Alice into sending some confidential documents to a HotMail address—that attacker can't forge Maria's digital signature because the attacker does not have Maria's private key. But there are three other specific attacks that the attacker might employ:

1. **The New Key Attack.** The attacker might send mail to Alice with Maria's `From:` address and signed by a different key. The attacker could claim that she is having computer problems—hence the new key—and ask that the confidential documents be sent to Hotmail.
2. **The New Identity Attack.** The attacker might send mail to Alice with a new key *and* a new `From:` address. The attacker could once again claim computer problems, or the attacker could simply claim that she is working from home and doesn't have access to her work computer system.
3. **The Unsigned Message Attack.** Finally, the attacker could send mail to Alice with a forged `From:` address, but this time the message could be sent without an accompanying signature.

These attacks are not unique to Key Continuity Management: in particular, both attacks #2 and #3 can be conducted with a traditional system based on keys certified by Certificate Authorities.

This chapter doesn't mean to argue that KCM is a superior authentication strategy to third-party certification *in theory*. Instead, it argues that certification with third-party party Certificate Authorities has so many barriers to its use that there are many times it is not used *in practice*. At very least, KCM is more secure than no certification at all. When faced with CAs that do not actually certify the identity of certificate holders, KCM may actually provide more security than CA-based systems, since KCM-based systems will warn when keys are changed.

7.1.7 Applying key continuity management to S/MIME

Given that support for S/MIME is broadly deployed, it would seem that the only barrier to its general use for securing email is the difficulty that users have in obtaining certificates. (Other problems remain if these certificates are to be used for signing contracts.)

One approach that would get S/MIME certificates into more hands would be for the existing CAs that give away free e-mail only certificates to work with the S/MIME client vendors so that the these certificates could be automatically obtained when a new email address was configured into a mail client. Currently, the only established CA that issues free e-mail only certificates reports that it is not interested in creating such a system,[Ing05] but this could change.

As an alternative to Digital IDs issued by a trusted third party, S/MIME clients could implement a

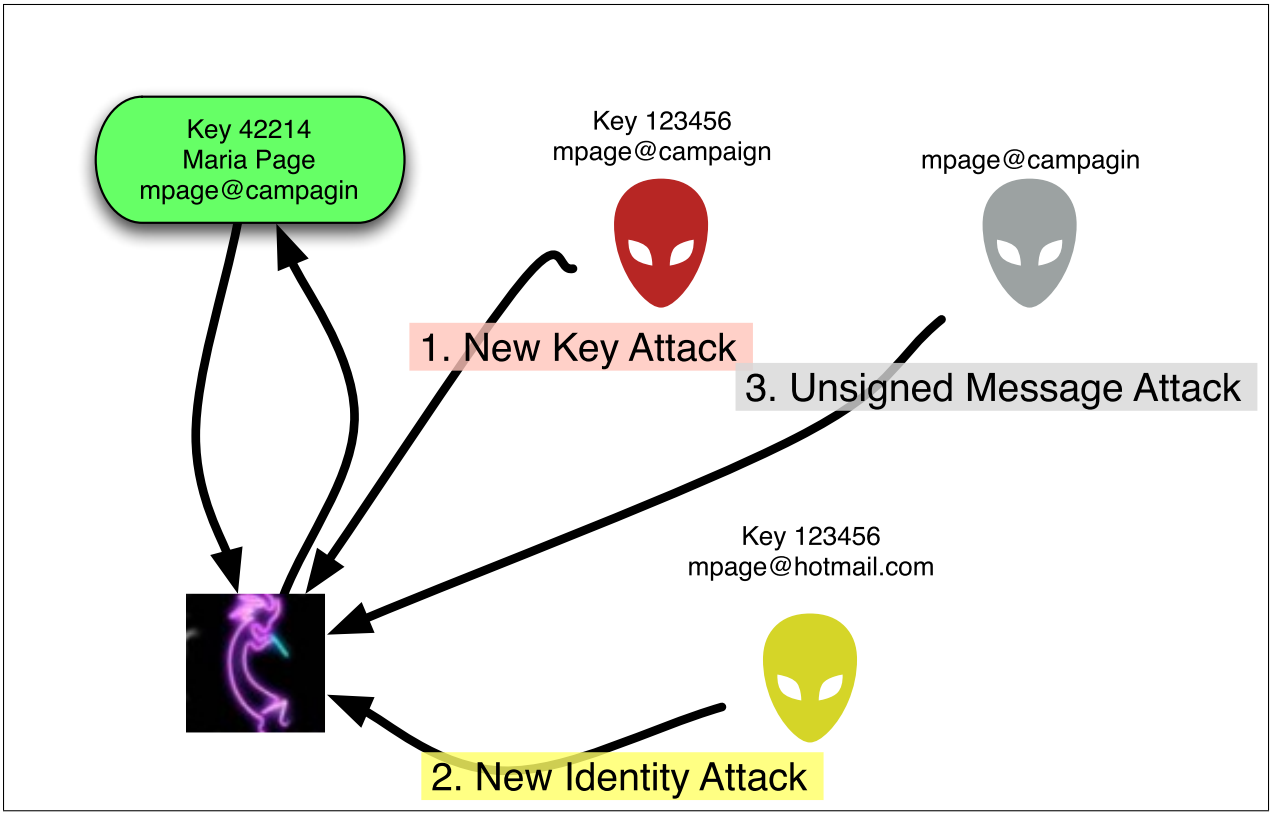


Figure 7-4: Three spoofing attacks possible against a user (lower left) employing Key Continuity Management to certify a series of communications with Maria Page (`mpage@campaign`).

form of KCM. Indeed, much of what is required for a functioning KCM system is already present in the S/MIME standard and clients: every S/MIME message is supposed to include all signing certificates, and S/MIME clients are supposed to incorporate certificates from every received message into the message store.

Full support for Key Continuity in the S/MIME environment would require the following:

- S/MIME clients would need to automatically generate a new certificate for every new mail identity *when that identity is first used.*
- If the same identity is to used to send email from multiple clients, the clients would need to distribute the certificate and the corresponding private key.
- S/MIME clients would need to *notify* the user the first time that a message from an email address is certified by a private key that matches an accompanying self-signed Digital ID.
- S/MIME clients would need to *track* how many times each (certificate,email address) pair had been received and present this information to the user.
- S/MIME clients would need to *alert* the user if the self-signed Digital ID for a particular email address changed.
- S/MIME clients would need to *alert* the user if an email address that normally used digital

signatures sent a message that was not digitally signed.

- S/MIME clients would need to *distinguish* between identities that are certified with self-signed certificates and those which are certified through external authorities—the theory being that an external authority would become more trustworthy if it was observed to sign many certificates with which the user is in communication.

Much of this could be done today without modification to S/MIME clients, solely through the use of a mediating proxy and a specially created "permissive" certification hierarchy. The design of such a system is presented in Appendix D.

7.2 Patterns for Improving Message Security

Based on the analysis of user desires and expectations for secure messaging, the analysis of software capabilities, and the consideration of PKI's history, this thesis proposes the following patterns for improving the security of today's email systems without negatively impacting the usability of those systems. In many cases, applying these patterns will simultaneously increase usability and security.

- **LEVERAGE EXISTING IDENTIFICATION** (page 330)
  Use biometric, PKI, and other strong identification systems to authenticate pre-existing relationships, rather than creating new ones. For example, both AOL and E*TRADE recently decided to allow their users to purchase RSA Security's SecurID 2-factor authentication tokens for use with their services.[Sec04][Sec05a] In both of these cases, the services are using SecurID to validate pre-existing relationships, rather than to certify new ones.
- **EMAIL-BASED IDENTIFICATION AND AUTHENTICATION** (page 331)
  The ability to read email at a pre-arranged email address can be used as a weak authentication. This approach, called Email-Based Identification and Authentication, is now being widely used for password recovery[Gar03a] and proof-of-identity in some anti-spam systems. It can be leveraged for recovery of passwords from desktop applications (which would send email to remote systems to unlock passwords), for distributing private keys to bootstrap PKI, and many other purposes.
- **SEND S/MIME-SIGNED EMAIL** (page 332)
  As discussed in Chapter 6, the vast majority of Internet users who are not using Webmail systems now have the ability to receive and validate mail that is signed with S/MIME signatures, provided that those signatures are made with a certificate from Thawte or VeriSign. Organizations sending bulk do-not-reply email should send it signed with S/MIME signatures.
- **CREATE KEYS WHEN NEEDED** (page 333)
  Software should automatically create keys and self-signed certificates when installed, rather than waiting for users to manually perform these operations. Although SSH[Ylo96] now performs this function, today's email systems do not. Without the easy availability of self-signed S/MIME certificates, there has been no incentive for software vendors to develop easy tools for working with this certificates.
- **MIGRATE AND BACKUP KEYS** (page 337)
  When keys are created, they need to be migrated to every machine that might need use of them. Keys also need to be backed up. For example, if a person reads email on multiple

computers, each of those computers needs to have access to the S/MIME private key that is needed to unseal any signed messages. Currently this is a difficult and error-prone manual process. It can and needs to be automated

- **TRACK RECEIVED KEYS** (page 335)
  In order to make use of self-signed certificates, it is necessary for the computer to track the recipient's history with the certificate and to make that history understandable to the user. The theory is that a self-signed certificate is not very trustworthy on the first day that it is seen, but that it becomes more trustworthy with extended use. Software needs to be able to be able to distinguish those conditions to the user.
- **TRACK RECIPIENTS** (page 336)
  Likewise, it is important for client software to understand the difference between a correspondent who has the ability to send and receive S/MIME-encoded mail, one that has the ability to send signed but not to receive sealed, and the ability to use other message security systems such as PGP. Currently this kind of tracking must be performed by the users of mail security software. This functionality needs to be moved into the software itself.
- **KEY CONTINUITY MANAGEMENT** (page 334)
  When an X.509 certificate is received that is not signed by a trusted CA, the certificate's trust settings needs to be directly managed by the client software using the certificate history as guidance.
- **DISTINGUISH INTERNAL SENDERS** (page 338)
  Visually distinguish between mail sent from within an email system with mail sent from outside the system that has the same `From:` address as internal senders. This is pattern codifies the practice described in Section 5.5.2

7.3 Testing KCM with *Johnny 2*

The study described in this chapter, *Johnny 2*, is based on a radical reinterpretation of Whitten and Tygar's *Johnny* results. It is possible that the usability problems uncovered in the *Johnny* user study were not driven by the PGP 5.0 program itself, nor by the lack of training offered within the program, nor by PGP's key certification process, but by the underlying key certification model used by PGP. *Johnny 2* seeks to determine whether or not the usability barriers can be overcome by replacing third-party certification with Key Continuity Management.

Whitten and Tygar uncovered many usability failings in PGP 5.0. Among these failings were the program's use of two incompatible public key encryption algorithms (RSA and El Gamal), the use of a nonsensical feather to denote signing, and the lack of user-accessible logs that detailed the use of third-party key servers.

But while the usability failings found in PGP 5.0 can certainly explain the failure of PGP 5.0 in the marketplace, these failings can't explain the similar failure of every other secure messaging system that implements public key cryptography. Such a failure can be explained by a common usability failure in the underlying certification model used by these systems: Before Alice can send Bob a piece of sealed email, Bob needs to first create a public key and get that key to Alice. Furthermore, Bob needs to either convey the key directly to Alice, so that she knows that it really came from Bob,

or needs to somehow certify the key so that Alice will trust it. This problem can be called the *Public Key Deadlock*.

7.3.1 *Johnny 2*

This project started as an attempt to replicate the setting of Whitten and Tygar's original *Johnny* study, but replacing PGP with a system that implements KCM. The goal was to test the hypothesis that users could complete the same task using software based on the KCM model with a higher success rate than observed in *Johnny* using the traditional model.

Johnny 2 needed to demonstrate that relatively naïve users with KCM could defend themselves against a variety of relatively sophisticated of spoofing attacks. To do this, *Johnny 2* needed to answer two separate but related questions:

1. Can Key Continuity Management make the task of sending and receiving secure email easier for untrained users?
2. Are the warnings that can be provided by a Key Continuity Management system sufficient to allow users to guard against spoofing attempts by third parties?

During the course of the user study, it became apparent that the *Johnny 2* study could also answer a number of other important questions:

- Do users who have been selected specifically so that they profess no knowledge of public key cryptography know, nevertheless, what it means to "encrypt" and "sign" a message?
- If users can encrypt email messages simply by clicking a button that says "Encrypt," will they click that button when they are sending information that has been designated by their boss as being confidential?
- If users can sign email messages simply by clicking a button that says "Sign," would they click that button?
- If users are faced with a situation in which they want to send a message with encryption but can't because they do not have a key for their intended recipient, what will they do?

Whitten and Tygar interpreted their *Johnny* results as an indication that security software has specific usability problems that make it different from non-security software. As such, the authors reasoned, security software must be developed with special care and using special techniques.

Although it may be possible to use safe staging and metaphor tailoring to teach untrained users the ins-and-outs of key certification, these techniques may be necessary for the sending and receiving of secure email if the underlying trust model can be revisited.

7.3.2 Deconstructing the Johnny scenario and findings

Care was taken to replicate as much of the *Johnny* experiment as possible to allow the results in *Johnny 2* to be compared directly with those of *Johnny*. In this way, it was hoped that any differences in the results could be attributed to differences in the underlying key certification model, rather than to differences in experimental setup or methodology.

The scenario in Whitten's original *Johnny* paper was interesting and straightforward: the experimental participant has shown up for work the first day as a volunteer at a political campaign that is trying to get a candidate elected to some state-wide office in Pennsylvania. The volunteer has been assigned the role of Campaign Coordinator and is responsible for sending the candidate's schedule to members of the campaign team. Quoting from Whitten's initial briefing document:

> "It is very important that the plan updates be kept secret from everyone other than the members of the campaign team, and also that the team members can be sure that the updates they receive haven't been forged. In order to ensure this, you and the other team members will need to use PGP to encrypt and digitally sign your email messages."[WT98, p.38]

According to [WT98, p.26], to succeed at the task of sending signed and encrypted email to the members of the campaign team, participants in the study needed to accomplish the following steps:

- Generate a key pair of their own.
- Make their public key available to the campaign team members, either by sending it to the key server, or by emailing it to them directly.
- Get the campaign team member's public keys, either by fetching them from the key server or by sending email directly to the team members to request their public keys
- Encrypt the secret message using the team members' public keys, sign it using their own private key, and send it.

Participants who completed these four steps within a 90-minute time limit were considered to have successfully completed the *Johnny* experiment. The job of the users participants was apparently complicated by the fact that these four steps were never explicitly stated: participants had to figure out the steps on their own by hunting around through the PGP interface and documentation.

Participants that accomplished these four tasks were sent email with additional tasks:

- Decrypt a signed and encrypted message from the campaign manager Maria Page.
- Make a backup copy of private and public keys.
- Create a key revocation certificate with the private key, so that the keys can be revoked at a later point in time even if the private key is lost.[WT98, p.27]

Considering that Whitten and Tygar excluded participants who had prior knowledge of public key cryptography, it is surprising that *any* of the participants were able to complete these tasks!

Table 7.1 presents a summary of the individual user tests from Whitten's *Johnny* experiment.

Key Certification in *Johnny*

One problem with the *Johnny* scenario is that none of the keys created or used by the experiment participants were ever certified in a manner that would protect against an active man-in-the-middle attack. Specifically:

Task	Success Rate	Succeeding Johnny Participants
Successfully generated a key pair	92%	P1, P2, P3, P4, P6, P7, P8, P9, P10, P11, P12
Obtained the public keys of other team members	50%	P3[a], P6, P8[b], P9[c], P10[d], P11[e], P12
Sent mail that was signed with their own key and encrypted with the other campaign members' key	25%	P6, P9, P12
Decrypted and read Maria's reply	33%	P6, P8, P9, P12
Backed up their keys	42%	P1, P4[f], P6, P8, P10
User created a revocation certificate	8%	P6
User that completed all tasks	1	P6[g]

[a] "With some prompting from the test monitor posing as Maria."
[b] After receiving "three successively stronger hints from the test monitor posing as Maria
[c] "after two fairly explicit prompts from the test monitor posing as Maria."
[d] "After prompting"
[e] But P11 didn't trust the keys; see text.
[f] Backup was in the same folder as the original key ring
[g] Scores P11's arguably correct decision not to trust the other campaign worker keys as a failure. See discussion in main text.

Table 7.1: A summary of results from Whitten's Johnny experiment; drawn from information presented in [WT98].

1. Participants were not provided with PGP fingerprints of the campaign workers' keys.
2. Participants were not allowed to call the fictional campaign workers to read a key fingerprint over the phone.
3. Participants were not provided with certified photographs of the campaign workers, and then given the ability to compare these photographs with photographs that had been digitally signed by the key owners as being authentic representations of the key owners. (This is a feature that exists in the current version of PGP but did not exist in PGP 5.0; nevertheless, this form of certification could have been performed manually using PGP 5.0, for example, by signing PostScript files that display a person's photograph and key ID when they are printed.)
4. Participants were not given campaign worker keys on a trusted floppy disk.

One of the *Johnny* participants seemed aware of this problem. P11 "didn't successfully send signed and encrypted email because she was afraid to trust the keys she got from the key server.... Too afraid of making a mistake to trust the keys that she got from the key server, alarmed by the default "untrusted" key properties, didn't appear to notice that the keys were all signed by Maria."[WT98, p.35] As a result, P11 is scored in Table 7.1 as not having successfully completed all tasks. But in fact, P11 may be the only participant who successfully completed all tasks: given the *Johnny* scenario, all of the other participants may have fallen prey to an elaborate spoof attack conducted by the opposing campaign.

The *Johnny* scenario seems to implicitly assume that both the campaign email and the campaign keys stored on the PGP key server are secure. If so, then the cryptography provided by PGP is

protecting against an adversary who can conduct passive eavesdropping of the campaign's Internet connection, but it does not protect against an attacker who can create their own PGP key, upload it to the keyserver, and then use that key to spoof the user playing the role of the Campaign Coordinator. This is not an adversary that was commonly seen in 1997, and it is not one that is commonly seen today. Today's adversaries find it much easier to upload keys to a key server, which can be done fro manywhere in the world, than to eavesdrop on the communications between two computers on a local area network.

7.3.3 CoPilot: A realization of key continuity management

Had Whitten and Tygar used a system like Stream (see Appendix D on page 413) for *Johnny*, it is likely that they would have seen radically different results:

- Whereas PGP 5.0 required that users manually create their keys, Stream automatically creates public/private key pairs as necessary and distributes the user's public key certificate to email correspondents on every mail message that it sends.[3] Thus all users in the study would have been able to create successfully generate a key pair, because the software would have done this for them automatically.
- Whereas PGP 5.0 required that users explicitly choose to sign and encrypt outgoing email, Stream automatically encrypts and signs all outgoing email whenever it has a key for the intended recipient. Although Stream did not automatically consult the PGP key servers to look up keys for email addresses when such keys were not in the user's local keyring, such an obvious additional feature could have been implemented if there was need.[4] Since the keys for the campaign team members were all uploaded to the PGP key server in 1998 (and are still there to this day, in fact—see Figure 7-5), all users in the study would have been able to obtain the keys for the campaign members and send them mail that was signed with the Campaign Coordinator's key and encrypted for the appropriate recipient.
- Since Stream automatically decrypts encrypted mail that it receives and verifies the signatures, all users in the study would have been able to decrypt and read the email that Maria Page sent them.
- Although Stream did not create backup certificates or revocation certificates, this procedure could have been easily automated as part of the automatic key creation process. If it had been automated, all users in the study would have been able to perform the function—and they would have done it without prompting by Maria.

In other words, had Whitten and Tygar used a system such as Stream that automated all key management and cryptography functions for *Johnny*, it is quite likely that 12 out of 12 of their subjects would have been able to complete all tasks. This is because the only tasks that *Johnny* tested are those tasks that were (or could be) automated by Stream.

The primary objection to systems such as Stream is that they automate key handling, but at the cost of leaving the user more vulnerable to a variety of man-in-the-middle and spoofing attacks. Thus, these systems violate Whitten's "Rules for making security invisible," which basically states

[3] Recall that the S/MIME encryption standard also aggressively distributes the user's S/MIME certificate.

[4] Indeed, the MailCrypt PGP plug-in for GNU Emacs implements this functionality.[Bud02]

Public Key Server – Index “wanton.trust ”

```
Type bits /keyID    Date       User ID
pub  1024D/57D7649C 1998/07/09 campaign coordinator2 <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/506B10DD 1998/07/09 campaign coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/C431A239 1998/06/26 *** KEY REVOKED ***
                               alma <alma@wanton.trust.cs.cmu.edu>
pub  1024D/6DE02262 1998/06/24 *** KEY REVOKED ***
                               Campaign Coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/2F815D2C 1998/06/23 *** KEY REVOKED ***
                               Campaign Coordinator <ccord@wanton.trust.cs.cmu.edu>
pub  1024D/F0DBC67F 1998/06/23 *** KEY REVOKED ***
                               Campaign Coordinator <coord@wanton.trust.cs.cmu.edu>
pub  1024D/591C3F74 1998/06/18 *** KEY REVOKED ***
                               ccoord <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/FA90DCC2 1998/06/17 *** KEY REVOKED ***
                               ccoord <ccoord@wanton.trust.cs.cmu.edu>
pub  1024D/EE0E3657 1998/06/17 *** KEY REVOKED ***
                               Campaign Coordinator <ccoord@wanton.trust.cs.cmu.edu>
pub   768D/4BAC8697 1998/05/29 Paul Butler <butler@wanton.trust.cs.cmu.edu>
pub  1024D/87B70A3D 1998/05/29 Maria Page <mpage@wanton.trust.cs.cmu.edu>
pub  1024R/F618116D 1998/05/28 Ben Donnelly <bend@wanton.trust.cs.cmu.edu>
pub  1024D/B49B8110 1998/05/28 Sarah Carson <carson@wanton.trust.cs.cmu.edu>
pub  1024D/AEB041ED 1998/05/28 Dana McIntyre <dmi@wanton.trust.cs.cmu.edu>
```

Figure 7-5: The PGP keys for the *Johnny* study campaign team members were uploaded in May 1998 and were still present on the key servers on January 15, 2005, as shown by this search of the keyserver at http://pgpkeys.mit.edu/ for the string “wanton.trust.”

that security systems should not be invisibly automated if there is a chance that the systems will sometimes make mistakes.[Whi04a, p.9]

Gutmann and others have suggested that the most users could achieve perfectly serviceable security if they simply had a system that automatically warned them when the keys of their correspondents changed. This is the essence of his Key Continuity Management proposal.

Johnny 2 tests this suggestion with a second-generation secure messaging proxy called CoPilot and a modification to the *Johnny* scenario that incorporates a series of spoofing attacks that are specifically designed to exploit weaknesses in the KCM model. CoPilot was designed and implemented as a “Wizard-of-Oz” prototype for the *Johnny 2* study. The term “Wizard-of-Oz” is used to indicate that users were tested on a prototype that works with the assistance of the experimenter working “behind the curtain.” This follows the example that Whitten set with Lime, a system that she designed and tested for her dissertation, without implementing in its entirety.

CoPilot Design
CoPilot is designed to be realized as a plug-in for programs such as Eudora, Outlook, or Outlook Express. Copilot could also be implemented as a combination POP and SMTP proxy, in a manner similar to Stream. The specific technique of implementation doesn’t matter, as long as CoPilot is able to act as a filter on all incoming and outgoing messages, and as long as CoPilot has a trusted channel through which it can communicate with the user.

For the purpose of the *Johnny 2* study, CoPilot's message engine is implemented as an outgoing message filter that processed messages as they were sent by the experimenter. CoPilot's user interface was implemented as an HTML frame around the message with a JavaScript-enabled button that could change the content of the CoPilot message.

CoPilot implements Key Continuity Management using a small set of rules:

- When any message containing a S/MIME certificate is received, that certificate is added to the certificate store. (S/MIME clients like Outlook Express are do this automatically, but CoPilot needs to track dependencies between certificates.)
- The first time that CoPilot receives a digitally signed message from a particular email address, that message is flagged with a *yellow* border.
- If subsequent digitally signed messages are received from that address, those messages are flagged with a *green* border. CoPilot will tell the user how many previous email messages have been received that were signed with the same certificate.
- If subsequent digitally signed message is received from that address that is signed with a different key, the message is flagged with a *red* border. The user can elect to trust such a key by clicking a button in the user interface. The user can change his or her mind by clicking the button a second time.
- If CoPilot receives an unsigned message from an email address for which it usually receives signed messages, the unsigned message is displayed with a *gray* border.
- If CoPilot receives an unsigned message from an email address that it has never previously seen, the message is displayed with a *white* border. Once the majority of email that is received by a user is signed, this option could be eliminated and all unsigned mail could be displayed with a gray border.

CoPilot's color codes are summarized in Table 7.2 on the next page. These colors are similar to those used by Cranor for the P3P Privacy Bird, which used green to indicate that a web site matches a user's preferences, yellow to indicate that a web site does not have a P3P policy, red to indicate that the site does not match the policy, and gray to indicate that the tool is turned off.[CAG02]

Although it might appear that an unsigned message should be a *red* condition, there are many instances in which legitimate email is sent by agents that do not have possession of the necessary private key. For example, Microsoft's "Outlook Web Access" will validate S/MIME signatures, but has no provisions to allow a sender to digitally sign outgoing messages. The author reads and responds to email using SnapperMail on a PalmOS-based wireless phone and an IMAP server; unfortunately, none of the mail clients that run on PalmOS have S/MIME support. SnapperMail's maker revealed that there are no plans to add support for S/MIME because of the added licensing fees for an S/MIME support library that could run on the Palm, and because of the lack of user demand.[Nic05]

If S/MIME support were extended to webmail systems and handheld devices, and if both private keys were automatically migrated between these systems, CoPilot's *gray* color could be eliminated and replaced with *red*. (Key migration is discussed in Appendix D on page 413.) In these circumstances, the system would properly give a strong warning if a user who had previously used

Frame Color	CoPilot Displayed Text	CoPilot Meaning
Yellow	This message is yellow because it is the first signed message that you have received from this email address.	A Yellow Border will appear around an email message the first time a particular Digital ID is used with an email address.
Green	This message is green because you have received `$COUNT` messages from this Digital ID.	A Green Border will appear around an email message each successive time that a particular Digital ID is used with an email address.
Red	This message is red because email from `$QFROM` was previously sent using different Digital #`$SN_OLD`. This message was sent using Digital ID #`$SN_NEW`	A Red Border will appear around an email message if the Digital ID used with that email address changes. This might indicate that the sender has moved to a different computer, or that someone else is trying to impersonate the sender.
Gray	This message is gray because it was not sent using a Digital ID and the sender of this message usually uses Digital IDs.	A Gray Border indicates that no Digital ID was used to send the message. The sender might have forgotten or have a computer problem. Alternatively, the message might be sent by someone else who is trying to impersonate the sender.

Table 7.2: The color codes displayed by the CoPilot program. The descriptions in this table are the same as those that are supplied to users in the **Color+Briefing** group, as described in Section 7.3.6. The text displayed in the right-hand column is representative text that is displayed with the message from the user test.

CoPilot differs from Stream in several important ways:

- Whereas Stream supported the PGP encryption standard,[a] CoPilot supports the S/MIME standard.
- Whereas Stream was written in C++ and used GPG as an encryption engine, CoPilot is largely written in python and uses OpenSSL as its encryption engine.
- Whereas Stream was operational and used by the author for several months, CoPilot only functions well enough to generate messages for the *Johnny 2* user test.
- Whereas Stream could send email messages to the user but otherwise had no user interface, CoPilot's design includes a rich user interface that gives users some control over trust management.

[a] Erik Nordlander spent a semester adapting Stream to work with S/MIME, but the work was not completed.

Table 7.3: CoPilot vs. Stream

certificates to send signed messages suddenly stopped.

The name "CoPilot" comes from the idea that CoPilot functions as a kind of security expert who watches over the user's shoulder and understands how to perform a variety of security-relevant tasks. CoPilot maintains a database of email addresses and certificates—implementing the *Track Received Keys* pattern—and displays the information in context to the user.

7.3.4 *Johnny 2*: Giving the *Johnny* scenario teeth

In the *Johnny 2* scenario, the fictional campaign team has decided to equip its computers with CoPilot. Ben Donnelly, the campaign's IT coordinator, has loaded S/MIME certificates for some

but not all of the campaign members into the address book of the computer being used by the Campaign Coordinator. *Johnny 2* thus tests the key continuation model and, hopefully, controls for other variables.

Johnny 2 clarifies and further develops both the political campaign and the Attacker. To make the task seem more realistic, the individual campaign members are given roles and backstories, as shown in Table 7.4.

The instructions that the participants received for the task consisted of three paragraphs of text that appeared in the human subject consent form and a single page entitled "Initial Task Description." In retrospect, hiding the task inside the human subject consent form may have been a mistake—a mistake that Whitten did not make. Many of the people who participated in the study as subjects appeared to be serial human subjects who participate in many studies on the MIT campus and have been apparently conditioned to ignore the verbiage contained in the consent forms. This problem was avoided by reading every word of the consent form to the subjects. Nevertheless, by placing the text inside the consent form, the attack and the security precautions were not at the forefront of the subject's mind when the experiment began. This helped in the establishment of a realistic test scenario.

The Attacker is an affiliate with the opposing campaign who is determined to use trickery to obtain the schedule, but is not willing or able to break into the candidate's email server or call the candidate's ISP and have the password on the candidate's email server reset. Instead, the Attacker tries to trick the experimental subject (playing the role of the Campaign Coordinator) into revealing the candidate's secret campaign schedule.

Equipped with knowledge of the campaign's personnel (perhaps obtained by previously calling up the campaign and getting a list of the workers), the Attacker sends a series of three email messages to the subject posing as members of the Campaign. In each of these email messages the Attacker tries to get the subject to send the candidate's schedule to one of several Hotmail accounts. In the scenario, the Attacker has previously created these accounts with names that are similar to members of the campaign. This employs an approach that is outlined by Mitnick.[MS02] For the experiment, the messages constitute an escalating attack that can be used to gauge the effectiveness of various defenses offered by CoPilot.

As it happens, the Attacker's messages are sent on the subject's first day on the job. And what is the subject doing this first day? The subject is sending out copies of the Candidate's coveted schedule to all of the members of the campaign team, each message sent at the request of Campaign Manager Maria Page. Due to a variety of circumstances, no other member of the Campaign team is in the office.

Not content with simple trickery, the Attacker attempts to maximize his chances of success by jamming the Campaign's telephone lines. Such attacks are actually quite easy to do, and have in fact been carried out in the past during actual political campaigns—a similar attack was carried out against New Hampshire's state Democratic party in an attempt to counteract the party's get-out-the-vote effort on Election Day 2002.[Sch04b]

Thus, *Johnny 2* is an experiment that is similar to *Johnny*, but which looks at the intersection of

Experimental Subject:		
Campaign Coordinator	ccord@campaign.ex.com	Experimental subjects are told: "You are the Campaign Coordinator."
Campaign Personnel:		
Maria Page	mpage@campaign.ex.com	Campaign Manager and the Coordinator's boss.
Paul Butler	butler@campaign.ex.com	Campaign finance manager.
Ben Donnelly	bend@campaign.ex.com	IT coordinator. Officially Paul's assistant, but also a full-time student at the University of Pennsylvania.
Sarah Carson	carson@campaign.ex.com	"A full-time graphics designer."
Dana McIntyre	dmi@campaign.ex.com	Office manager, but away for the week because her husband is having surgery. (Don't worry, it's a routine procedure.)
Attacker:		
Attacker Paul	butler@campaign.ex.com	Claims to be Paul Butler, having computer problems.
Attacker Sarah	sara_carson_personal@hotmail.com	Claims to be Sarah Carson, sending email from home using her "personal Hotmail account" because she can't get to her campaign email from home.
Attacker Maria	mpage@campaign.ex.com	Attacker "Maria" sends an unsigned message to the Campaign Coordinator asking that the schedule be sent to both Ben and Sarah.

Table 7.4: Personas used in the *Johnny 2* experiment.

usability and security issues that are likely to arise if the underlying problems that were diagnosed in the *Johnny* experiment are resolved. Another way of interpreting *Johnny 2* is that it is the *Johnny* experiment updated to one of the leading security problems of our time: the "phishing" attack.

Figure 7-6 summarizes the similarities, non-material differences, and the material differences between *Johnny* and *Johnny 2*.

7.3.5 The *Johnny 2* messages and the experimenter's work bench

The *Johnny 2* test consists of a series of eight email messages sent to the experimental subject playing the role of the Campaign Coordinator. Each message has a specific purpose and is designed to elicit a particular response. Table 7.5 presents a summary of the messages. The actual messages appear in Appendix C on page 381, where they are discussed in detail.

It is apparent from reading Whitten's reports and thesis that messages sent to test subjects during the *Johnny* trial were composed interactively during the experiment and sent by the experimenter.[5] This approach was rejected out-of-hand for *Johnny 2* for several reasons, including:

- Composing messages during the trial could lead to mistakes such as typographical errors,

[5]Although the Whitten's writings contain many technical details of the *Johnny* experiment, notably missing are the actual messages that were sent by the experimenter to the study participants. In December 2004 Whitten was contacted and asked for a copy of the messages. Whitten responded that she had not retained them, but recalled that the messages were "pretty minimal" and consisted of little more than a three-message sequence:

1. "I couldn't decrypt that message, something must be wrong."
2. "I still can't decrypt that message, something is still wrong."
3. "I still can't decrypt that message, are you using my key to encrypt?"[Whi04b]

msg #	CoPilot Color	Sender	Content
#1	Yellow	Maria Page	Introductory message introducing Maria and giving the Campaign Coordinator details of the campaign worker's stories. The Coordinator is told to reply. This message provides the subject with information and verifies that they can read and respond to written instructions. This message is also an internal control: Subjects that do not respond to Message #1 within a reasonable amount of time are disqualified and withdrawn from the experiment.
#2	Green	Maria Page	The Campaign Schedule and a command telling the Coordinator to send a copy of the schedule to Paul Butler and Dana McIntyre. This message further tests that the subject can respond to a written command from Maria. It also gets the subject into the rhythm of reading an email message and responding by sending out the schedule.
#3	Green	Ben Donnelly	Ben asks the Campaign Coordinator for a copy of the schedule. The message is green because Ben's Digital ID was previously installed on the computer.
#4	Red	Attacker Paul	Paul says that he is having computer problems and asks the Coordinator to send a copy of the schedule to both Paul's campaign account and his personal Hotmail account, `Paul_J_Butler@Hotmail.com`. This message is digitally signed with a Digital ID that claims to be from `butler@campaign.ex.com` but which is signed by a different Digital ID. This is a *new key attack*. Note: This message has a "Reply-to:" header that causes a reply to be sent to Hotmail. In retrospect the Reply-to header complicated the scenario and should not have been present.
#5	Yellow	Attacker Sarah	Attacker Sarah sends email from her Hotmail account `sara_carson_personal@hotmail.com` saying that she is working at home and asking that the schedule be sent to the personal account. This message is digitally signed with a valid Digital ID—it is simply an email address and ID that the subject has not previously seen, making this a *new identity attack*.
#6	Gray	Attacker Maria	If the subject does not succumb to both message #4 and message #5, then message #6 is sent. This message is an unsigned message that purports to come from Maria Page, the Campaign Coordinator's boss. Attacker Maria says that she has tried to call the office but that the phones are not working. Maria says she has been on the phone with both Paul and Sarah and that they both need copies of the schedule; please send them! Now! Do it! This is an *unsigned message attack*.
#7	Green	Maria Page	In this message, the real Maria Page asks the Campaign Coordinator to send copies of the schedule to Ben Donnelly and Sarah Carson. Some subjects were confused that Maria sent this message, as they had already sent a copy of the schedule to Ben in response to message #3. (In the scenario, Maria didn't know that Ben had asked for the schedule.) Participants who fell for Attacker Maria in message #6 were especially confused; they couldn't understand why Maria was now asking them to email the schedule to Sarah's campaign address when she had just asked that the schedule be sent to Sarah's personal Hotmail address. This message was a very useful test message to probe precisely what the subject thought had happened in message #6.
#8	Green	Maria Page	Maria thanks the subject for participating in the experiment and tells the subject that it is now time for the "Debriefing Interview." Although it wasn't strictly needed, this message gave the experimenter a gentle and in-scenario way to end the experiment.

Table 7.5: The *Johnny 2* Messages

Similarities:

- The same recruitment poster was used, except that the name and contact information for "Alma Whitten" was substituted with the name and contact information for "Simson."
- The compensation to subjects was the same.
- The same preliminary interview was used to weed out individuals who had experience with programs like PGP, who knew the basics of public key cryptography, or who knew the difference between "asymmetric key cryptography" and "symmetric key cryptography."
- Similar language was used in the consent form, with minor changes to reflect that the study was taking place at MIT with Outlook Express and CoPilot and not at CMU with PGP.
- The campaign team personas all have the same names and email addresses.

Non-material Differences:

- Instead of testing users on a Macintosh with Eudora and PGP, users were tested on a computer running Windows and Outlook Express.
- Instead of being recorded with a video recorder, the computer's screen and user comments were recorded using Camtasia Studio 2.[Tec05]
- The email domain used in *Johnny* was `wanton.trust.cs.cmu.edu`, while the email domain used in *Johnny 2* is `campaign.ex.com`.
- Campaign members are given specific roles, making the personas more believable.
- Instead of being given the secret campaign schedule on a piece of paper, a more detailed secret schedule was sent to the subject playing the role of the Campaign Coordinator in an email message.

Material Differences:

- *Johnny 2* tests Key Continuity Management, not mutual certification.
- *Johnny 2* has an articulated attack that is consistent between user trials.
- *Johnny 2* has both internal controls on each experiment run and a control group.
- *Johnny 2* has statistically significant results.

Figure 7-6: Similarities, non-material differences, and material differences between *Johnny* and *Johnny 2*

messages being sent to the wrong address, messages being sent without encryption or signing, and so on.

- If different test subjects received different messages, it would be difficult to perform anything but a qualitative analysis on the research findings.
- Given the need for the experimenter to take notes, the added overhead of writing detailed replies to email messages would have been very demanding.
- If the experimenter was obviously responding to the subject's email, the experiment would have lost considerable verisimilitude.

Instead, a program called the "*Johnny 2* Experimenter's Work Bench" was created for adminis-

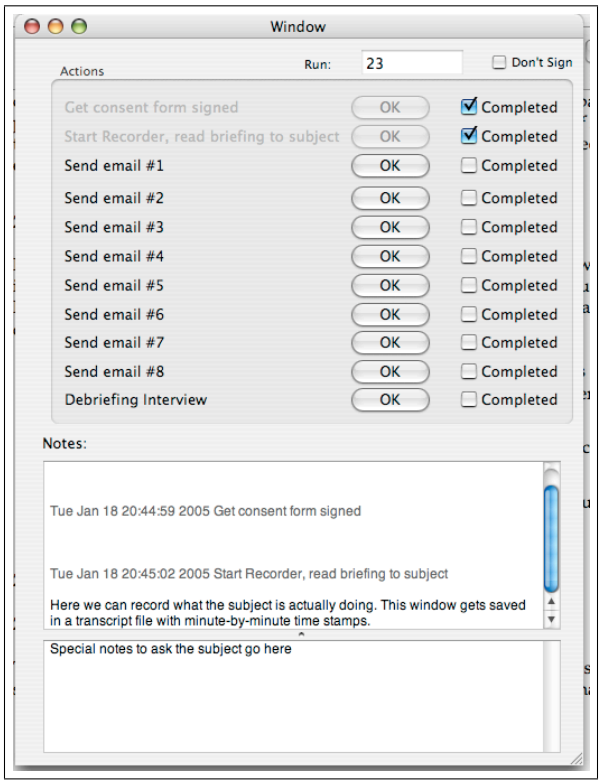


Figure 7-7: The *Johnny 2* Experimenter's Work Bench. As the experiment takes place, the experimenter successively presses each "OK" button on and takes notes in the two text areas on the bottom. Notes are automatically timestamped every minute and the notes file is saved with every keystroke. The transcript is stored in an RTF file where it may be reviewed by the experimenter or processed with automated tools.

trating the experiment (Figure 7-7). This program consisted of a graphical user interface running on the experimenter's Macintosh computer and two underlying programs, `sendmessage` and `send_signed`, that performed the actual business of sending email messages to the subject. The work bench program gives the experimenter a place to make notes, automatically timestamping them each minute and noting when each test message is sent to the subject:

- `send_signed`, the program that actually sent the S/MIME-signed email messages called for in the *Johnny 2* experimental protocol. Written in the Python programming language, `send_signed` has command-line arguments for specifying:
  1. The private key to sign the message
  2. The matching public key certificate, which is appended to the end of the message
  3. The recipient's email address (`ccord@campaign.ex.com` in the *Johnny 2* experiment)
  4. The message body
  5. The CoPilot template to use. Templates available include the red, yellow, green (2 version), gray (2 versions), and "none."
  6. The previous certificate serial number that CoPilot saw with a given email address. This option was used for creating the first attack message.

7. The message subject.
8. The `reply-to` address that should be used, if any. This option was also used for creating the first attack message.
9. The number of previous email messages that have been seen using this certificate.
10. Carbon-copy recipients. (These recipients are listed on the message sent to the experimental subject, but not actually sent to the email addresses.)

- `sendmessage`, the program that implemented the "business logic" of the *Johnny 2* experiment. It is a Unix shell script that accepts two arguments: the message number to send and an optional "-z" argument. Providing the "-z" argument caused the subject to receive messages bordered with the uninformative grey border, rather than with a colorful and informative CoPilot border. Essentially, this option turned off the CoPilot program. It was used for the **NoColor** cohort, as described in the next section.

The `send_signed` Python program contains much of the machinery that is needed to implement a fully functional CoPilot program. All that is needed in addition is either an Outlook Express plug-in or a functioning POP and SMTP proxy, such as that created for the Stream program, and a persistent database. Although such a program could easily have been created, it was not necessary to do so to complete the *Johnny 2* testing.

7.3.6 Three cohorts: NoColor, Color, and Color+Briefing

By now it was clear that the *Johnny 2* experiment was very different from the original *Johnny* experiment. Thus, *Johnny* could not be used as a control for *Johnny 2*. Instead, it was necessary to devise a new set of controls for *Johnny 2*.

Each *Johnny 2* experimental run contained three internal controls implemented as specific test user tasks. The tests consisted of messages #1, #2, and #7. Before and after the attacks, the Campaign Coordinator is asked by Maria Page to reply to Maria's first message (#1) and send a copy of the campaign schedule to the four legitimate campaign personas (#2 and #7).

Because *Johnny 2* was designed to test the effectiveness of the CoPilot user interface and Key Continuity Management approach, *Johnny 2* further divided the experimental pool into two categories: those for whom the CoPilot program was engaged (the **Color** group), and those for whom it was not engaged (the **NoColor** group). Those in the **Color** group saw all email messages with the colored borders and explanatory text, while those in the **NoColor** group saw all email messages in a gray boarder with no explanatory text. Figure 7-9 compares the Outlook Express interface that users in the **Color** and **NoColor** groups saw.

A disturbing trend emerged during the first dozen runs: although the **NoColor** group was being routinely spoofed, as expected, the **Color** group was also being spoofed! This was *not* what we wanted to have happen. Careful observation of the subjects revealed that many were simply screening out the information that CoPilot was providing: the HTML interface integrated so successfully into Outlook Express that many users reported that they thought it was just another email header that they could safely ignore. And without reading the mail headers, the colors had no meaning. Subject S10 went so far as to report on her debriefing interview that she "found the colors very annoying"—entirely missing the point of the CoPilot interface.

In the test, you will be asked to play the role of a volunteer in a political campaign. After you volunteered, you were given the role of Campaign Coordinator. Your task is to send updates about the campaign plan out to the members of the campaign team by email. It is very important that the plan updates be kept secret from everyone other than the members of the campaign team, and also that the team members can be sure that the updates they receive haven't been forged. In order to ensure this, you and the other team members will need to use CoPilot to make sure that all of the email messages are secure.

Your email address for the purpose of this test is `ccord@campaign.ex.com`, and your password is volnteer. You should use the title "Campaign Coordinator" rather than using your own name.

Outlook Express and CoPilot have both been installed, and Outlook Express has been set up to access the email account. No manuals for these programs are provided, but there may be some online help. A pad of paper and pens are also provided, if you want to use them.

Before we start the test itself, I'll be giving you a very basic demonstration of how to use Outlook Express to send and receive mail. The goal is to have you start out the test as a person who already knows how to use Outlook Express to send and receive email, and who is just now going to start using CoPilot to make sure your email can't be forged or spied on while it's being delivered over the network. The Outlook Express tutorial will take about 5 minutes, and then we'll begin the actual testing. ~~You can also use Mozilla Thunderbird if you would prefer, but not all of the advanced features of CoPilot work with Mozilla.~~

Figure 7-8: The task description that was hidden in the consent form. The entire consent form appears in Section C.2.2 on page 384. The option to use Mozilla Thunderbird was removed after the consent form was granted approval by MIT's Committee On the Use of Humans as Experimental Subjects..

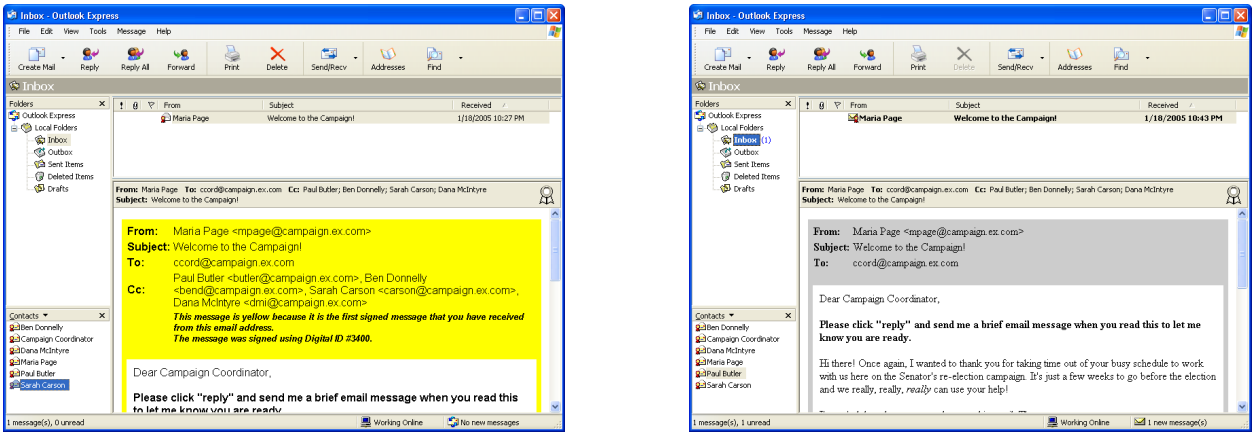


Figure 7-9: *Johnny 2* message #1 with CoPilot engaged (left) and not engaged (right). In both cases the messages are signed. However, when CoPilot is engaged, the program displays a yellow border and explains that the border is yellow because this is the first time that the user has received a signed message from the email address `mpage@campaign.ex.com`. In both cases the Outlook Express address book is displayed in the lower left-hand corner.

Digital IDs allow Outlook Express to authenticate the sender of email messages.

A Yellow Border will appear around an email message the first time a particular Digital ID is used with an email address.

A Green Border will appear around an email message each successive time that a particular Digital ID is used with an email address.

A Red Border will appear around an email message if the Digital ID used with that email address changes. This might indicate that the sender has moved to a different computer, or that someone else is trying to impersonate the sender.

A Gray Border indicates that no Digital ID was used to send the message. The sender might have forgotten or have a computer problem. Alternatively, the message might be sent by someone else who is trying to impersonate the sender.

Figure 7-10: The briefing received by the subjects in the **Color+Briefing** group. Each box was typeset with the background of the box being the color that the box purported to describe. The briefing took 50 seconds to read out loud to the subjects; the briefer directed the subject's attention to the printed words by pointed to each word as the word was read.

Rather than scuttle the experiment, at this point a decision was made to add a third cohort. This group, called **Color+Briefing**, received a written briefing before the experiment started that consisted of one sentence that described what a Digital ID is and four color-coded boxes that described what the CoPilot colors red, green, yellow and gray mean. The briefing appears in Figure 7-10.

This briefing was seamlessly incorporated into the study by printing up a separate set of "Initial Task Description" documents which appear in Section C.2.2 on page 384. To help ensure a random distribution of the remaining subject trials, the number of test subjects in each pool was increased from 12 to 14. The remaining number of **NoColor**, **Color** and **Color+Briefing** scheduled subject trials were randomly shuffled and assigned. The final ordering appears in Section C.3.1 on page 402. Table 7.6 on the following page summarizes these cohorts.

This rather small intervention resulted in a significant change for those who received it: nearly all of the subjects in the **Color+Briefing** group detected the spoof and were able to avoid being tricked by the attacker some of the time. One possibility is that the subjects were primed to the possibility of a spoofing attack and were now on the lookout. But another possibility is that as a result of being told what these colors meant, the subjects now knew what to look for and, as a result, were not screening out indicators that they would have otherwise ignored. These results are discussed in detail in Section 7.5.

It is important to note that the only difference between these three groups was the activity of the CoPilot program and the presence (or absence) of the written briefing. All three groups received the same digitally signed (or unsigned) messages. All three groups were free to use the security features in Outlook Express to learn more about the Digital IDs that were used to sign the messages.

Cohort	# subjects	Distinguishing characteristics
NoColor	14	Subjects in the **NoColor** group were presented with an interface that had CoPilot's Key Continuity Management system disabled and all messages were surrounded with a gray border.
Color	14	Subjects in the **Color** group were presented with CoPilot's standard multi-color interface, as discussed in Section 7.3.3.
Color+Briefing	15	Subjects in the **Color+Briefing** group were presented with CoPilot's standard interface and given a briefing (Figure C-15) describing what a Digital ID is and what the different CoPilot colors might mean. This briefing was included on the "Initial Task Description" document that the subjects received and additionally read to the subjects by the experimenter.

Table 7.6: Differences between the **NoColor**, **Color** and **Color+Briefing** cohorts

7.3.7 S/MIME setup

Johnny 2 makes extensive use of the S/MIME facilities that are built in to Outlook Express 6. OE6 is used to verify the signature on incoming signed S/MIME messages; to allow the user to send S/MIME signed and sealed message by clicking a button; to verify the contents of a certificate; and to manage certificates through the OE6 address book. In order to make use of these capabilities, S/MIME certificates needed to be created for each of the experiment personas. Ideally, such creation would be performed automatically by CoPilot. Because this feature of CoPilot was not implemented, certificates had to be manually created or obtained by the experimenter.

One way to create the certificates would have been to obtain them from a commercial CA such as VeriSign or Thawte. This option was considered and rejected for three reasons. First, there was the issue of expense. Second, there was the issue of legality: because they are attempting to create a true Public Key Infrastructure, CAs such as VeriSign and Thawte require that users click through many legal agreements to get a certificate: it was not clear whether or not obtaining certificates for fictional entities would be consistent with the spirit, let alone the letter, of these organizations' Certificate Practice Statements. The third reason that commercial certificates were rejected was a matter of pride: the author felt that he could not claim to really understand how S/MIME works, and thus argue how to improve it, unless he was able to create his own S/MIME certificates, import those certificates into programs such as Outlook Express, use those certificates, and also write command-line programs to create and send S/MIME-signed and encrypted email.

Creating the certificates turned out to be an important learning experience. A complete discussion of the process appears in Section C.4.

7.4 Walk-Through

This section describes the specific experimental procedure that we used for *Johnny 2*.

7.4.1 Windows and Microsoft Outlook Express 6 configuration

User testing was done on a Dell Optiplex GX270 computer with a 2.4GHz Pentium 4 CPU, 1 Gigabyte of RAM and a 40 Gigabyte hard drive. The computer was Windows XP Professional Version 2002 Service Pack 2. Display was a 17-inch Dell 1703PFt LCD display set at a resolution of 1280x1024 pixels, although the resolution was lowered to 1024x768 if the user had problem reading the small text. A photograph appears in Figure C-7 on page 385.

Subjects were given the option of using a Dell mouse (2 button with a scroll-wheel) or a Logitech Marble Mouse trackball. (None of the subjects chose the trackball.) A Dell 103-key keyboard was provided.

Testing was done with a specially created account named "Campaign Coordinator" with the password volnteer—the same account and password as used by Whitten and Tygar in 1998. The email program was Microsoft Outlook Express 6 (OE6) version 6.00.2900.2180 (xpsp_sp2_rtm.040803-2158).

Outlook Express 6 Email Accounts

OE6 was pre-configured with a single account named "email." This account was for a user name "Campaign Coordinator" with the email address `ccord@campaign.ex.com`. The incoming mail server was pop.ex.com with a POP3 account named "ccord" and the password "volnteer". POP3 mail was downloaded over SSL. Outgoing messages were sent to the SMTP server csail.mit.edu.

The email account's Security tab was configured with a certificate called "Campaign Coordinator." The certificate was issued to "Campaign Coordinator" by the "Certification Manager," valid from 12/9/2004 to 12/9/2005. OE6 was configured to use this same certificate for both signing and encrypting.

As in *Johnny*, each of the five campaign team members were represented by an email account that was accessible to the experimenter. Attacker accounts consisted of actual Hotmail accounts that had been obtained for the purpose of the experiment. All Digital IDs used in the experiment were created with OpenSSL, manually loaded in to the Outlook Express Address Book by sending messages digitally signed with the certificates to the Campaign Coordinator account.

Figure 7-11 shows the computer's screen at the beginning of the user test.

OE6 Options

Like most Microsoft programs, OE6 program preferences are sent through an Options panel that is accessed through the Tools menu. The OE6 options panel contains 10 sub-panels accessed through a set of double-decker tabs. These tabs contain 49 check boxes, 3 pull-down menus, 3 spinners, and push-buttons that can display 21 different sub-panels.

For the purposes of the *Johnny 2* study, there were two very important settings. Both of these options are checked by default in the standard Outlook Express 6 installation:

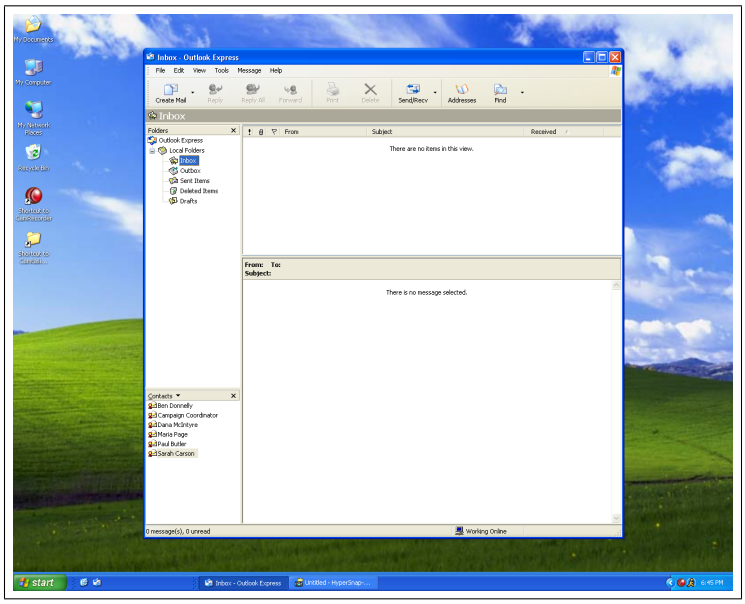


Figure 7-11: The computer's desktop at the start of the *Johnny 2* experiment. Outlook Express 6 is the main program, with the program's folder list (top left), message list (top center), message preview (bottom center), and address book (bottom left) clearly visible. This screen is displayed for all subjects, no matter whether they are **NoColor**, **Color** or **Color+Briefing**.

- The option "Automatically put people I reply to in my Address Book" on the "Send" tab of the "Options" panel was checked. When checked, this option causes a sender's email address to be automatically incorporated into the user's address book *when the message is replied to*.
- The option "Add senders' certificates to my address book" on the "Advanced Security Settings" panel of the "Security" tab of the "Options" panel was checked. When checked, this option automatically incorporates received S/MIME certificates into the user's address book *when the message is viewed*.

7.4.2 Reminder, greetings and briefing

Subjects were sent an email reminding them of the time and location of the experiment at 5pm the day before their trial. This email contained detailed instructions on how to navigate through the MIT Stata Center to room 32-G828, where the testing took place.

The orientation for the test had four components:

1. Prior to the subject's arrival, subjects were assigned to one of three groups: **NoColor**, **Color** or **Color+Briefing**.
2. All subjects were presented with a copy of the consent form (see Figures C-10 through C-13. This form was read to the subjects, signed, and then placed in a locked file cabinet.

   The consent form made the following points clear:

   - That the subjects were helping to test **Outlook Express and CoPilot**, not being tested themselves.

- That it would be extremely helpful if the subjects could "think aloud" as much as possible during the test.
- That the premise of the test was that the subjects were volunteering for a political campaign, and that their task would be to send email updates to the members of the campaign team, **using CoPilot to make sure that all of the email messages are secure—a term which is defined by context to mean "not forged" and "secret from everyone other than the members of the campaign team."**
- What their email address and password would be for the purposes of the test.
- That **Outlook Express** and **CoPilot** were already installed, and that the **online documentation**, pad and pen were there for them to use as much as they liked.
- That they would be giving a **brief** tutorial on the basic use of **Outlook Express** before the actual testing began.[WT98, p.25, with differences noted **in bold**]

3. Subjects in the **NoColor** and **Color** group were presented with the Initial Task Description shown in Figure C-14 on page 391, while those subject in the **Color+Briefing** group were presented with description shown in Figure C-15 on page 392.

   (Subjects were not told if they were in the **NoColor**, **Color** or **Color+Briefing** groups—subjects were not even made aware of the fact that there were multiple groups to which they could be assigned.)

   The Initial Task Description document was read to the subjects by the experimenter, and placed next to the computer's keyboard so that the subjects could easily refer to it at a later point if desired.

4. At this point, subjects were given a brief demonstration of Outlook Express. Points specifically mentioned were that the Send/Recv button could be used to send and receive mail; that new mail comes into the inbox; and that information on people in the address book could be learned by right-clicking on the name and then selecting the "Properties" menu. No mention was made of how to use the "Sign" and "Encrypt" buttons visible in the OE6 interface—in fact, no mention was made of those buttons at all.

7.4.3 Experiment sequence

The experiment began when the first email message was sent. During the experiment new email messages were sent when it was clear that the subjects had finished responding to an email message, or when roughly 10 minutes had passed since the sending of the previous email message.

The experimenter sat next to the left of the subject, outside the subject's field of view, taking notes. Questions that the subjects asked regarding non-security features of Outlook Express (for example, how to forward a message) were answered, but any question regarding the operation of an Outlook Express S/MIME feature, the Outlook Express Address Book, or the CoPilot interface was answered "I don't know." Subjects who asked for additional information regarding the briefing were referred back to the briefing.

Subjects who were quiet for extended periods of time were reminded "don't forget to think out loud."

At the conclusion of the experiment, subjects were given a "Debriefing Questionnaire" and asked

additional questions by the experimenter to clarify their understanding and the motivation of the actions that they had taken.

7.4.4 Experiment screens

The following photographs show how the messages are framed by CoPilot and displayed to the test subject by Outlook Express. Figures 7-12 through 7-15 show a sample of the screens that were seen by the **Color** and **Color+Briefing** groups. Figures 7-16 compares the colored messages with those seen by the **NoColor** group.

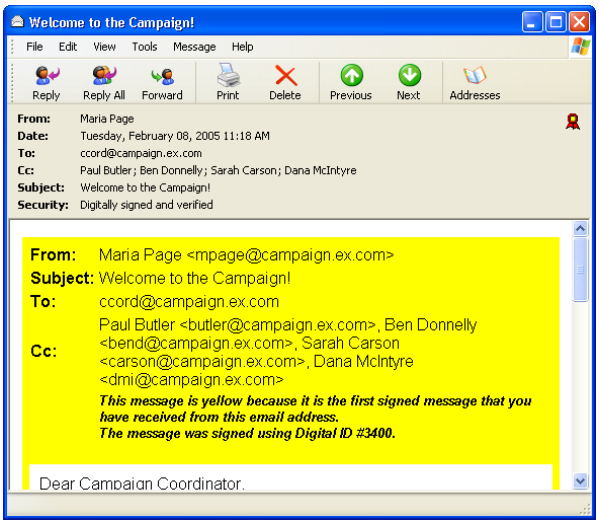


Figure 7-12: Message #1 (Yellow): Maria Page welcomes the Campaign Coordinator to the team and introduces the campaign members. Because this is the first time that CoPilot has seen this message, it is displayed with a yellow border.

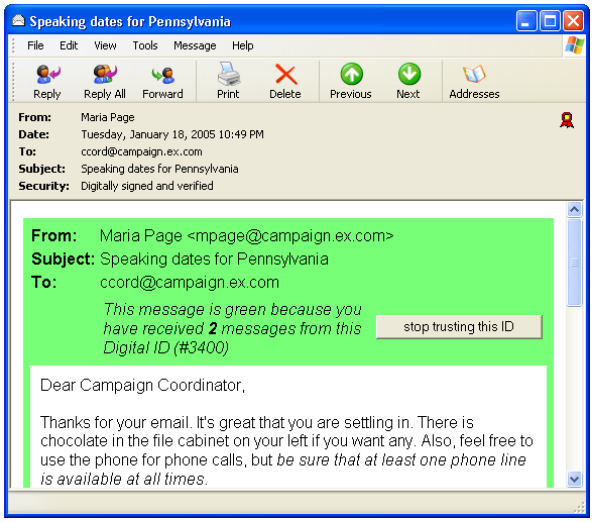


Figure 7-13: Message #2 (Green): Because this is the second message that CoPilot has seen from Maria Page, CoPilot displays this message in green.

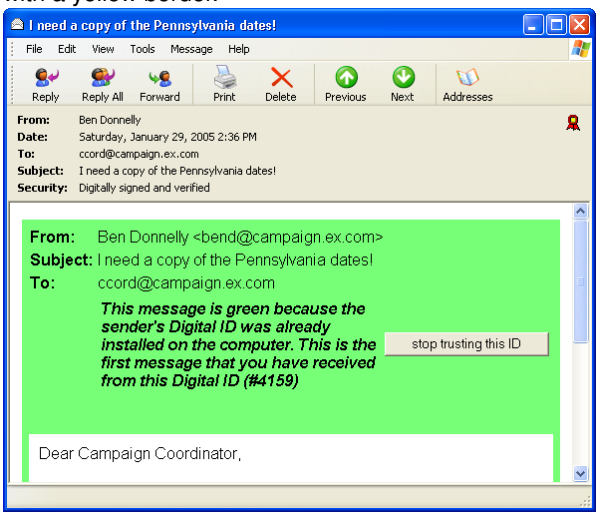


Figure 7-14: Message #3 (Green): CoPilot received Ben's key from Maria (because Maria cc'ed Ben in her message). Because Maria's key is trusted, this key is trusted as well, and it appears in green.

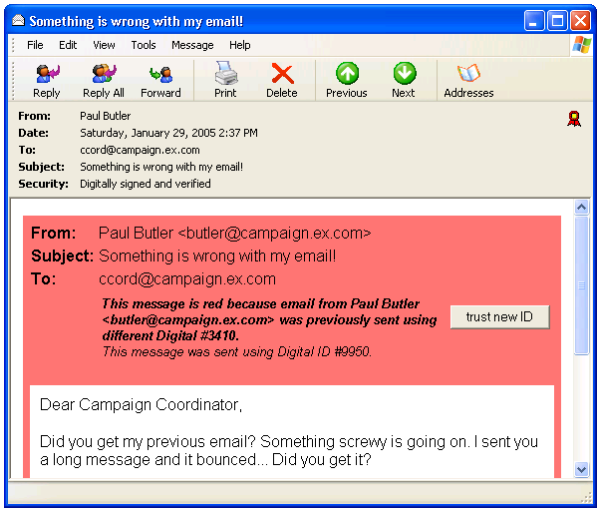
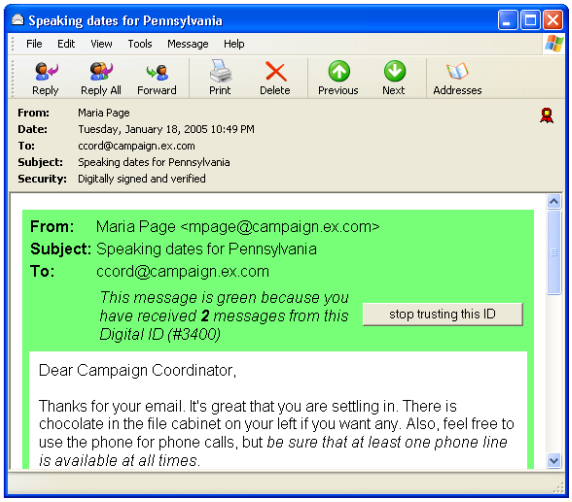
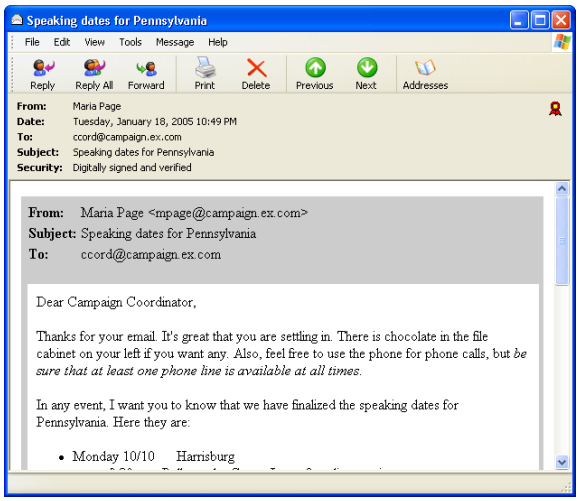


Figure 7-15: Message #4 (Red): CoPilot displays Attacker Paul's message in red because CoPilot had previously seen a Digital ID with this email address that is different from the Digital ID that Attacker Paul is actually using. This attack is possible because CoPilot uses Key Continuity Management with self-signed keys, but CoPilot can detect it.



Displayed to **Color** and **Color+Briefing**.



Displayed to **NoColor**.

Figure 7-16: A side-by-side comparison of the second message as displayed with CoPilot enabled and with it disabled.

7.5 Results and Discussion

A total of 43 subjects were run, with 15 in the **Color+Briefing** group and 14 in the other two. Details of subject recruitment appears in Section C.1.1 on page 381.

Runs averaged 40 minutes in time, with the shortest run lasting 17 minutes and the longest lasting 53. This section summarizes the most significant results observed from the subjects. When reported in tables, χ^2 values were calculated using a logistic regression.

7.5.1 Task comprehension

Overall, the subjects clearly comprehended both the task in which they were asked to participate, and the tools that they were given for performing that task. No users were confused by the fact that they were being sent messages that were digitally signed.

Subjects also quickly dropped into the routine of the scenario. Subjects who were very suspicious and interested in the security of messages #1 and #2 lost their interest by the time message #3 arrived. This was a surprising observation, as our subjects had signed up for a "Security Study" and had specifically been told that the other campaign might be trying to steal the campaign schedule.

Many subjects said that they felt as if they were under "time pressure" to complete the task within a certain period. This pressure appeared to come from the scenario itself, rather than from other commitments that the subject might have created outside the scenario. For example, several subjects noted that Attacker Paul said that he needed the schedule within the next 30 minutes—these subjects kept looking at the clock and at the timestamp of the that Attacker Paul had sent, to see if there was still time to satisfy his request! The subjects wanted to help the fictional personas.

In follow-up interviews it was clear that users generally understood that signing a message allowed

Cohort	n	% subjects resisting attacks sometimes	% subjects resisting attacks always	Clicked "encrypt" to seal email sometimes	Clicked "encrypt" to seal email always
NoColor	14	43%	0%	50%	21%
Color	14	50%	29%	36%	36%
Color+Briefing	15	87%	33%	20%	13%
χ^2		6.13	3.61	2.96	0.29
$p =$		0.013	0.57	0.087	0.59

Table 7.7: Summary Results of *Johnny 2* User Study

a recipient to verify who had sent the message and that "encrypting" (or sealing) the message prevented "the wrong people" from viewing the message's contents. Several of the users who received the unsigned message attack from Attacker Maria asked her to resend the message signed with her Digital ID so that they could verify that the message really did come from her. Most of were not sure if they were being attacked or not, but they felt that they could rely on the Digital ID to tell them if the Maria Page who sent message #6 was the same Maria Page who had sent the initial campaign email messages.

Interestingly, these same users were generally unaware that signing a message also provided integrity guarantees. In our experience, most email users are not aware of the fact that a message can be intentionally and maliciously modified as it moves through a computer network or waits for delivery on a mail server. Although we did not specifically ask our users if they realized this possibility, only one (S39) of the users in the study raised this possibility that a message might be maliciously modified. That user was so paralyzed by the notion that a malicious attacker might be modifying the email messages she was receiving that she was unable to complete all of the experiment tasks!

Many users, especially those in the **NoColor** group, struggled for some way to verify the authenticity of the attack messages. Some settled on a form of Email-Based Identification and Authentication[Gar03a]: they sent an email message to the attacker's apparent campaign address to see if the attacker could read and reply to such messages. Unfortunately, this approach was sometimes their undoing: the subjects occasionally succumbed to the unsigned message attack from Attacker Maria because the message appeared to have been written in response to a message that the subject had just written!

7.5.2 Evaluating KCM

As evidenced in Table 7.7, we found that CoPilot's KCM interface significantly ($p < 0.001$) enabled users in the **Color** and **Color+Briefing** groups to resist some attacks—in particular, the "new key attack" and the "unsigned message attack" (Table 7.8). The interface did not inoculate subjects against the "new identity attack." A discussion of these attacks appears in Section 7.1.6 on page 247.

	% of subjects that tried to send the schedule when requested by:					
Group	Maria 1	Maria 2	Ben	new key attack	new identity attack	unsigned message attack
NoColor	100% (14/14)	92% (11/12)	100% (14/14)	71% (10/14)	79% (11/14)	75% (9/11)
Color	93% (13/14)	100% (13/13)	92% (11/12)	64% (9/14)	50% (7/14)	58% (7/12)
Color+Briefing	100% (13/15)	100% (14/15)	100% (13/14)	13% (2/15)	60% (9/15)	43% (6/14)
χ^2	2.20 $p = 0.14$	0.018 $p = 0.89$	0.79 $p = 0.37$	10.61 $p = 0.001$	1.02 $p = 0.31$	3.98 $p = 0.046$

Table 7.8: Percentage of subjects that sent email containing the secret campaign schedule in response to commands from Maria and Ben, and in response to the three attacks. Numbers in parenthesis indicate the number of subjects who responded compared to the number who were subjected to the test condition. Subjects who misinterpreted the Maria 1 message and sent email to *all* campaign workers did not feel the need to comply with the Maria 2 or Ben messages because they had already done so; they were omitted from the sample. Because of the way in which the subjects responded to earlier messages in the scenario, not all subjects were exposed to the unsigned message attack.

KCM Against the New Key Attack

KCM worked significantly ($p = 0.001$) better against the new key attack than no KCM—especially when the subjects were briefed that a new key might indicate that "someone else is trying to impersonate the sender" (Figure C-15). The improvement was dramatic when users were specifically briefed of the two likely conditions that might result in a red message: "that the sender has moved to a different computer, or that someone else is trying to impersonate the sender."

KCM Against the New Identity Attack

KCM did not work significantly ($p = 0.31$) better than no KCM against the new identity attack. It can be argued that this is because the subjects were not primed that a yellow border could be an attack. We do not think that this criticism is warranted, however, because many subjects verbally debated whether or not the yellow message was in fact from the real Sarah who was in the campaign or from some other Sarah. Many rationalized that the key and the email address were different because Sarah was using her home computer—the justification present in message #5. Our subjects knew that they *might* be under attack: they simply decided to trust Attacker Sarah.

Only two subjects noticed that Attacker Sarah's Hotmail address had a misspelling in the name. S27 discovered the inconsistency before sending the message to Attacker Sarah but decided to send the message anyway; S33 used the misspelling to help confirm the decision not to trust a yellow message.

KCM Against the Unsigned Message Attack

KCM was more successful against the unsigned message attack, conveying statistically significant ($p = 0.046$) immunity from spoofing to those in the **Color** and **Color+Briefing** cohorts. Many users readily understood that there was no way to verify the sender of a message that wasn't signed. For example, **Color** subject S33 wrote to Attacker Maria:

> "You didn't sign this email so I can't verify that it is from you. Is it actually from you? If so, please sign the response!" (S33) [Thu Jan 27 13:26:12 2005]

After S33 hit the "Send" button, she said "So now I need to find out if she actually sent it." S33 was not content to use answerback authentication (see Section 7.5.5). A moment later, S33 laughed and realized that she hadn't signed her message, either.

We were surprised that the unsigned message attack wasn't more successful against users in the **NoColor** group. During the follow-up interview, we were told that what frequently protected subjects from following Attacker Maria's instructions was not the fact that message #6 was not signed: the indications in Outlook Express 6 that messages are signed are very subtle, and as a result not a single user in the **NoColor** group realized that message #6 was not signed while the other messages were signed. Instead, what seemed to protect users in the **NoColor** cohort from responding to message #6 was that Attacker Maria was asking them to send the secret campaign schedule to a Hotmail address: many subjects said that they simply did not trust Hotmail's security because Hotmail accounts can be obtained under any name that is available.

7.5.3 Evaluating the CoPilot interface

CoPilot's HTML-based interface was designed to look like the program had been tightly integrated with Outlook Express. As it turns out, the integration was a little too transparent. Although in the debriefing interview every subject in the **Color** and **Color+Briefing** group said that they saw the colored borders, we observed that users in the **Color** group frequently did not read the text that CoPilot displayed underneath the "To:" header (for example, in Figure 7-12). CoPilot integrated so well that users were ignoring it!

The "Trust this ID" button was never explained to the subjects. Only a few subjects experimented with the button to see what it did. Two subjects (S31 and S39) misunderstood: when they saw the green-bordered message with the button labeled "stop trusting this ID," these users thought that the legend on the button was an instruction *from* CoPilot *to them*, telling them that they should *stop trusting this ID*! Both users clicked the button, the CoPilot border changed from green to red, and the users were pleased that they had complied with the computer's instructions and apparently gotten the correct result.

Even though we excluded subjects who had used PGP or could distinguish between a symmetric and asymmetric cryptography, invariably some of our subjects had a working knowledge of cryptography. These subjects reacted very positively to the CoPilot interface: they liked the way that the interface made it so easy to know which messages were signed and which were not. It is not clear if subjects would feel this way if *every* message had such bold and colorful notifications, but this result indicates that companies like Microsoft and Apple might wish to raise the importance of S/MIME signatures in their interfaces.

7.5.4 "Encrypt"

Unprompted by the instructions but given the option by the Outlook Express interface, roughly a third of the users in our study clicked the "encrypt" button to seal the candidate's confidential schedule before it was sent by email.

The OE6 "encrypt" button is a toggle switch. Pressing the button once causes a little blue icon to appear next to the `To:` field in the message composition window. No encryption happens, though, until the user tries to send the message. At this point OE6 scans the Outlook Express Address Book to see if there is an S/MIME certificate on file that matches each `To:` address. If all of the addresses match, the message is encrypted and sent. If one or more of the addresses do not match, a warning appears (Figure 5-6).

Users who did not have the CoPilot Key Continuity Management interface were significantly ($p = 0.097$) more likely to use encryption than those who had the interface. Interviews with users revealed that many were using the intended recipient's ability to *unseal* a message as a proxy for *recipient authentication*. That is, subjects believed that only members of the campaign would be able to unseal messages that had been properly sealed. In follow-up interviews, several subjects said the campaign IT coordinator should have configured Outlook Express so that it would *only* send sealed messages if sealing messages was a campaign priority.

However, subjects were mistaken: OE6 was very happy to seal the message for Attacker Sarah, as Attacker Sarah's "yellow" message had been digitally signed and, as a result, her certificate had been automatically incorporated into the OE6 address book. Users didn't understand that a message could be encrypted for an attacker: those who were asked said that they thought that something about the CoPilot system would prevent encrypted messages being sent to someone who was not affiliated with the campaign.

Every user who discovered the "Encrypt" button and tried to send a message to Attacker Paul was confused when they could not send a sealed message to the Hotmail address (Figure 5-6). They couldn't do this, because Attacker Paul's message was digitally signed with a certificate that had the address `butler@campaign.ex.com`, and not `paul_butler@hotmail.com`. (It is appropriate that Attacker Paul was able to obtain such a certificate because the Campaign is using Key Continuity Management, and not third-party certification.) A handful referred to the online help or did web searches with Google to try to diagnose the problem: all of these individuals determined that the problem was that they did not have a Digital ID on file for Attacker Paul's Hotmail address. Several users attempted to change the email address on Paul's campaign Digital ID to his Hotmail address so that they could send sealed mail to his Hotmail account; others tried in vain to figure out how to "make" a Digital ID for the Hotmail Account. Two of the users sent mail to Attacker Paul telling him that they could not send him the schedule until he got a Digital ID and sent him instructions for obtaining one.

7.5.5 Use of email answerback as an authenticator

Many subjects attempted to use some form of email answerback as an authenticator. That is, when subjects received the message from attackers Paul, Sarah and Maria, they sent mail to the campaign email accounts belonging to Paul, Sarah and Maria asking for verification of the message.

A good example of this was Subject S11, a 28-year-old PhD candidate in education with 11 years' experience using computers. S11 engaged with the experiment and sent chatty, in-scenario emails to the fictional characters. Her first email to Maria said "Hi, Maria. I'm ready to assist when you need help." She followed the instructions of email #2 and sent the schedule to Paul and Dana, then followed the instructions of email #3 and sent the message to Ben.

When S11 received the first attack email, her reaction was to send an email not to Attacker Paul's Hotmail account, but to worker-Paul's campaign account, stating:

> Hi, Paul.
>
> I rec'd this message but would prefer to send you the dates to your campaign email address.
>
> Please reply when you have a moment.
>
> thank you,
>
> CC

Even though Attacker Paul said that Paul's campaign email account was not working, S11 decided that the only way she had to authenticate the message was to send it to Paul's campaign email and wait for sensible confirmation.

There is a problem with S11's attempts at answerback authentication: because the message sent to Paul (and a later message sent to Maria) did not contain a nonce, a password, or some other kind of secret, there is no obvious way for S11 to evaluate a message that appears to be sent in response. That is, S11 constructed a situation in which the response method had to be self-authenticating, and there was no way for this to take place.

Indeed, when S11 received message #6 from Attacker Maria, S11 believed that the message was sent in response to one of her challenges. Even though the message was not signed, S11 decided to ignore CoPilot's warning and sent out the schedule to the attackers. She said:

> "This one is not sent using a digital ID. That's neat that it knows that. That the sender usually uses digital IDs, but it is coming from her ex address. So now I am terribly confused and I don't understand how the digital ID works. If it is something that someone logs on with . . . I thought it was with something special about the ex address, but apparently I was very wrong—and very naïve.
>
> Now at this point, she is my boss. She is the big wig. So I will do what she tells me."[Fri Jan 7 15:19:04 2005]

Even in the face of an unsigned email message telling her to do something that she knows is wrong, S11's resolve crumbled in the face of a determined social engineering attack. After receiving attack message #6, she sends the secret to Attacker Paul and Attacker Sarah's Hotmail addresses. But S11 cc's Maria's campaign address on the email—both as proof to Maria that she did it, and as a way of alerting Maria that something might be wrong.

Next, S11 receives legitimately signed message #7. The fact that message #7 is signed makes S11 all the more suspicious of unsigned attack message #6. S11 now realizes that message #6 was an attack and sends the following letter to Campaign Manager Maria Page:

Cohort	% Asking for phone	Subjects
NoColor	(6/14)	S1 S9 S15 S24 S38 S43
Color	(6/14)	S4 S6 S7 S11 S23 S33
Color+Briefing	(10/15)	S16 S18 S20 S22 S27 S32 S34 S36 S39 S44

Table 7.9: Subjects asking for the phone. A logistic regression comparing these three groups found a $p = 0.19$, indicating that there is no significant difference between the groups.

> Maria!
>
> I received an email from you, although it was not signed w/ a digital ID. I sent the schedule to Paul and Sarah, w/ a cc to you.
>
> Apparently, by the looks of this email, you did not just send the previous one (that I copied you on).
>
> Please assist ASAP.
>
> Apologies, CC

S11 then goes to send a copy of the schedule to Ben and Sara (S11 is so flustered that she actually forgets to send the schedule to Sara). Following the "think out loud" protocol, she says:

> "So I might as well send it to Ben and Sarah, although they are probably going to change it because I messed up."[S11, Fri Jan 7 15:26:04 2005]

The message to Ben says:

> Ben,
>
> Per Maria's request. Please be careful as there seems to be sketchy email communications occurring.
>
> Sincerely,
>
> CC

In the debriefing, S11 said that if she had been working at a real campaign, she expects that she would have been briefed as to what a Digital ID was and what it means to sign and encrypt a message. Indeed, such a briefing could have been done in less than two or three minutes.

7.5.6 Use out of band authentication

Many of our subjects attempted to use out-of-band channels to authenticate the sender of the messages. The most obvious out-of-band channel was the provided phone: 22 out of the 43 subjects asked to use the phone, as shown in Table 7.9.

S36 wondered aloud why the campaign phone should be secure, when the campaign's email might not be.

S40 sent email to Maria, asking for Paul's phone number. When S40 then got message #6 from Attacker Maria saying that the phone was out of order, S40 thought that Attacker Maria's message was a direct answer to S40's message asking for the phone, and took Attacker Maria's response as confirmation that the schedule should be sent to the Hotmail addresses.

Although subjects were specifically told that they could ask the experimenter for a phone during the initial briefing, many subjects nevertheless chose not to make use of the apparent opportunity. S14 wrote on the debriefing questionnaire "I regret that I didn't use the phone." S41 had a similar response, saying "I definitely would have used the telephone to verify a Hotmail address" if the scenario had been real. S21 read attacker Maria's message that she had gotten off the phone with Sarah, but never thought to use the phone to call either of them.

The conclusion is that some but not all of the subjects thought that they could and should use the telephone in an attempt to verify the sender of the email message: although they were guided to this decision by the scenario, it was not preordained. Given that even unsophisticated computer users are generally familiar with telephone systems, designers should look for ways to leverage out-of-band authentication systems when practical. As noted in Section 6.3.4, Groove already has provisions for such authentication.

7.5.7 Other observations

In addition to the reported data above, there were several behaviors that were common among a noticeable minority of subjects. These behaviors included:

- Some subjects decided to embellish the copy of the schedule that they sent with messages for the recipient imploring them to maintain operational security. Ironically, many of these messages were sent to the Attacker personas!
- Some subjects cc'ed Maria Page on copies of the calendar that were sent to both the campaign and the attacker personas. Although it seemed that subjects were more likely to cc Maria Page when they were not sure if they were making a mistake, no statistical analysis was performed of cc behavior.
- Although the phrase "Digital ID" appeared in the "Initial Task Description" document, only one subject (S17) actually asked "what's a Digital ID?" during the briefing. Subjects who were asked "do you know what a Digital ID is?" or "can you give me a definition of a Digital ID?" during the debriefing could give only a very poor and largely inaccurate definition of the phrase—even though they had been using Digital IDs for 30-45 minutes. An interesting follow-up study would be to re-run the *Johnny 2* protocol and ask subjects in a formal manner to define the term "Digital ID" and explore its perceived uses.
- We found that many Webmail users had fundamental problems understanding the Windows-style button-based interface. Two users exemplified this problem by not understanding the CoPilot button labeled "Stop trusting this Digital ID." Yes, that button could have been labeled "click here to stop trusting this Digital ID." But many other behaviors were seen, such as users who clicked on the Inbox folder to get their mail (a common idiom with webmail systems), users who were confused by the Outlook Express "New Message" window which appeared and then got lost underneath the main Outlook Express window, and by the modal panels associated with the Outlook Express address book itself. Watching these users made the

experimenter wonder how successful a desktop mail client would be that cloned the easy-to-use properties of todays webmail systems.

- The attackers always say that there are phone problems, but the legitimate campaign personas don't. When the experimental subjects asked for the "phone" and discovered that, yes indeed, there really are phone problems, this tended to legitimize the attacker personas.
- Some subjects used the Outlook Express tools for viewing certificate properties. However none of the subjects were able to make any sense of the information that Outlook Express provided. (Usability difficulties with current certificate viewers are discussed in Chapter 6.)
- Many subjects were confused by the effect of the `Reply-to:` field in Attacker Paul's message: even though the message appeared to come from the Paul's campaign address, hitting "reply" created a new message that would go to Attacker Paul's Hotmail address. This confusion could have been avoided through the use of a `Reply-to:` attack, as explained in Section 11.1.3.

7.5.8 *Johnny 2* problems and possible improvements

Finally, in the course of conducting this user test, many ways became readily apparent by which the scenario and experimental machinery could be improved. These possible improvement appear below.

Scenario Improvements:

- In general, the number of members on the campaign team should be increased from 4 to 6 or 10. This would make the scenario more realistic. It would also avoid such confusing-producing events as when Maria asks the subject to send the schedule to Ben Donnely when the subject has already sent the schedule to Ben.
- Many subjects were confused that the Campaign Coordinator was asked to forward the email message, rather than having Maria send the message herself. Maria said that there was something wrong with her email system. As a result, the claims from Attackers Paul and Sarah that there were problems with the campaign's email system requiring that email be sent to Hotmail were that much more believable.

  One way to avoid this problem would be to change the nature of the "secret" that the Campaign Coordinator needs to protect. Furthermore, different secrets could be used for different attackers. Attacker Paul could want the password to the campaign's bank account. Attacker Sarah could want a copy of the briefing book for an upcoming debate. These and other potential "secrets" could be made available to the subject as icons on the test computer's desktop.
- It was also confusing that both Ben and the attackers said that they had recently been in telephone contact with Maria. The attacker was a little too good.
- At least one of the subjects seemed to be fundamentally disinterested in maintaining campaign security. One way to make this requirement more personal for the subjects would be to base their payment on whether or not campaign security was maintained—for example, by paying the subjects $10 for participating and another $10 if they didn't leak the secret.
- Instead of telling the campaign coordinator that their cell phone didn't work, it might have been more realistic to say that they forgot to charge their cell phone and the battery is dead.

- The debriefing interview should have included questions asking the subjects to define terms such as *Digital ID*, *Encryption*, *Security*, and *Sign*.
- The test should have included a legitimate request to send the campaign secret to a non-campaign address. Otherwise, subjects can adopt a simple rule that *campaign.ex.com* is good while *hotmail.com* is bad.
- Instead of using the same campaign worker names as Alma Whitten, different names should have been used. Several subjects tried in vain to find other contact information for the campaign works; if they had used Google, they would have discovered not their worker's phone numbers, but Whitten's 1998 CMU technical report![WT98] Reading the report would have revealed so much about the study as to have invalidated the subject's run.
- Giving different attacks to different subjects, or varying the attack order, would have reduced the impact of subjects that discussed the details of the test with others. As it turns out, this was not a problem, but it could have been.
- It might be desirable to directly test people on their ability to use Outlook Express and disqualify those who cannot pass a minimal functional requirement.

Technology improvements:

- All mail sent from experimental subject should have clearly indicated the Subject number in the email header. One way that this could have been done would be by changing the Outlook Express "Real Name" from "Campaign Coordinator" to "Campaign Coordinator S6" (for example). This would have simplified data analysis.

7.6 Conclusion

Using a modified version of the *Johnny* study developed by Whitten and Tygar, we tested the Key Continuity Management (KCM) proposal using three different user interfaces. We found that significant increases in security can be with relatively minor enhancements to the way that programs like Outlook Express handle digitally signed mail.

We found that people who had less than a minute of training seemed to be immediately comfortable with the concept of digital signatures and cryptographic protections. Although we did not test the depth of their knowledge of these facts, our subjects felt that the signature authenticated the sender of a message and that "encryption" (sealing) prevented someone who was unauthorized from viewing its contents. Subjects had a much harder time understanding that you needed to get a correspondent's Digital ID in order to send them a sealed message.

We found that KCM made it possible for all but one of our experimental subjects to send and receive secure messages, a 98% success rate. This compares favorably with Whitten's *safe staging* approach, which also achieved a significantly lower success rate. Because of methodological flaws in the Lime study, only a qualitative comparison is justified. We also found that KCM allows users to defend themselves, with a high probability of success, against some of the attacks that are thought to be prevented by traditional certification approaches. We note that there is a lack of user testing of these traditional approaches that would allow us to determine if traditional approaches actually provide the security that they are thought to provide.

We have shown that even though the deployment of KCM could improve security, it is not the panacea to the mail security problem for which we are looking. In particular, KCM provides users with no readily apparent tools for deciding whether or not to trust new identities that show up with new keys. But this isn't a problem that is created by KCM. With today's PKI-based systems, for example, an attacker can create a new Hotmail address and then get the key certified by VeriSign or Thawte. And this problem is no different from similar problems in the offline world. You receive a letter from an old friend asking a favor: is it really from the same person?

In all likelihood, different kinds of email need and can employ different kinds of certification. Within some organizations a centralized PKI might be appropriate; other organizations that do not have the wherewithal to manage such a deployment might choose to employ KCM—perhaps with some form of additional out-of-band certification, as was suggested by several of our subjects. It isn't hard to see that the current CoPilot interface could be modified to support different kinds of "green" messages: those that simply reflect trust from continued use of a key, those that have been explicitly certified by a close third party (such as a co-worker), and those that had been certified by a commercial certification authority.