Electronic Documents and Digital Signatures

Dartmouth Computer Science Department, Technical Report TR2003-457

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science

by

Kunal Kain

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 30, 2003

Examining Committee:

Sean Smith (chair)

Edward Feustel

Chris Hawblitzel

Carol Folt
Dean of Graduate Studies

Abstract

The natural progression today is to use electronic documentation instead of paper in order to streamline the business environment. For digitally signed electronic documentation to be incorporated in today's market, people need to have at least as much faith in them as paper. My thesis explores the flaws in how many popular electronic document formats and COTS PKI packages mesh together, and how adversaries can construct digital documents whose viewed contents can change in useful ways without invalidating the signature using these flaws, and will also discuss some countermeasures.

Acknowledgments

There are many people to thank and I would like to start with my girlfriend Elizabeth who has supported me throughout this process and helped me through this writing, my parents who have supported me throughout my life and encouraged all my choices. Another important person I must thank is my advisor, Professor Sean Smith who has provided guidance, assistance and continuous support over the last two years. He has always been there to point the way, provide insight, and take part on all aspects of this thesis work. His intellectual originality, astute suggestions and fresh approach have been invaluable for this thesis research. I could never thank him enough for being an exceptional adviser.

My thanks also go to the other members of my thesis committee; Professor Chris Hawblitzel, who helped me look at this problem from a systems perspective; Professor Edward Feustel for his ideas on exploring this field and considering the bigger picture.

This work was supported in part by the Mellon Foundation, AT&T/Internet2, and the U.S. Department of Justice (contract 2000-DT-CX-K001). Therefore, my thanks go to them for making this project possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The purpose of digital signatures is to authenticate digital documents and give them a purpose—e.g., "I approve this.". Here, digital signatures provide us a way of checking who "I" is and the purpose is to "approve". This allows us to expand the application of digital documents so that people do not use them just as a convenient storage medium. This presents obstacles because the way in which digital signature schemes are integrated with applications leaves unconventional avenues of attack for devious minds to explore.

In today's business world, complex work flow formats and convenient interfacing allow us to introduce attacks that are mere manipulations of the internal ability of workflow format to introduce dynamic content.

Assessing a wide range of common digital signatures packages embedded in our popular workflow formats, this thesis creates taxonomy of attacks designed to effectively counter the security these COTS PKI packages are hyped to provide. It also provides a preliminary series of countermeasures to the attacks.

This thesis includes material from our paper "Digital Signatures and Electronic Documents: A Cautionary Tale", K. Kain, S.W. Smith, R. Asokan. [10], revised and extended.

Chapter 2 discusses the problem with common digital signature schemes and Chapter 3 provides scenarios in which an adversary (perhaps "Alice", or person "X" who gave Alice the document) can benefit by constructing dynamic objects. In Chapter 4, we discuss how popular workflow formats permit the construction of such objects. In Chapter 5, we discuss how many common ways of integrating PKI with workflow can still appear to validate signatures on these objects (although some packages resisted our attacks). In Chapter 6, we shall discuss some case studies of attacks. In Chapter 7, we consider some countermeasures. Chapter 8, concludes this thesis.

1.1 Digital Signature Principles

A digital signature associates a digital sequence with an electronic document to represent a handwritten signature on a paper printed document. This digital sequence should be considered similar to a handwritten signature. In principle, a digital signature is based on the use of two different digital keys known as a key pair. Each key pair is made up of a private key and a public key. The two are interdependent but can be used separately. Typically each key pair may belong to a specific key holder. The algorithm works in such a way that it is computationally infeasible for a third party to compute the private key even if they are in possession of the public key.

1.2 Signing Electronic Documents

To create a digital signature one usually uses a one-way hash function. A hash function is a transformation that takes an input and returns a fixed-size output. A hash function is said to be one-way if it is computationally infeasible to find some input given the output of the hash function.

Public-key cryptography is used to sign a digital document using a few simple steps carried out on the bits of the document as it is stored:

- Alice produces a one-way hash of the document to be signed using some pre-specified hash function. This creates a fixed length message from the document data. Any change in the bits of the document would profoundly alter the hash value.
- Alice performs her private key operations on this fixed length hash of the document.
- This value is the “signature” and is appended to the document, which is now said to be “signed”.

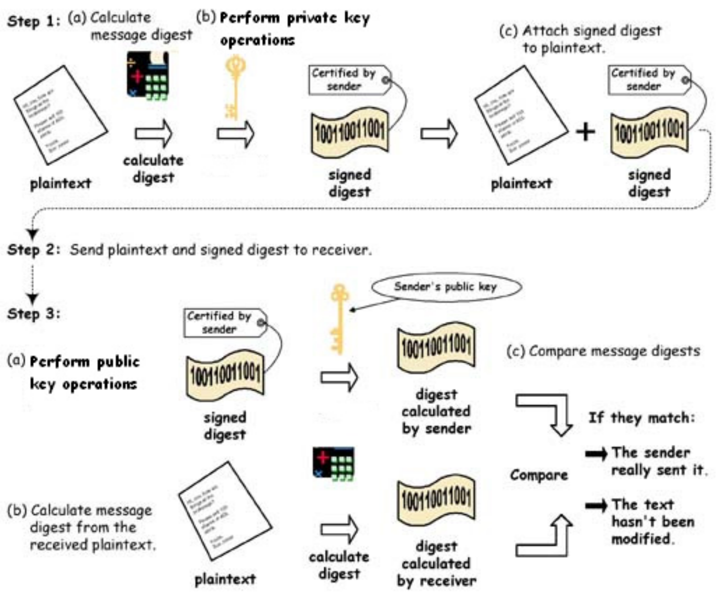

Step 1: (a) Calculate message digest
(b) Perform private key operations
(c) Attach signed digest to plaintext.
plaintext
calculate digest
Certified by sender
10011001100
signed digest
plaintext
+
signed digest
Step 2: Send plaintext and signed digest to receiver.
Step 3:
(a) Perform public key operations
Certified by sender
signed digest
Sender's public key
digest calculated by sender
(c) Compare message digests
If they match:
The sender really sent it.
The text hasn't been modified.
Compare
(b) Calculate message digest from the received plaintext.
plaintext
calculate digest
digest calculated by receiver

Figure 1.1: Digital signatures creation and verification

- Alice sends this signed document to Bob.
- Bob first uses the same hash function on the document.
- Bob does Alice's public key operations on the "signature" appended to document.
- These two values are compared.
- Finally if they match it verifies that the document came from Alice and it was not altered.

This scenario is sketched in Figure 1.1.

Properties of digital signatures are:

- The signature should be authentic, i.e. should convince the recipient that the signer deliberately signed the document.
- It should not be reusable, i.e. it should not be possible to transfer the signature to another document.
- it should be unalterable, i.e. the digital document should not change after it is signed without detection.

- It should be non-repudiatable, i.e. the signer should not be able to claim he/she did not sign it.

Chapter 2

Our Dilemma

2.1 Problems with digital signatures

One of the most common uses of public-key cryptography is for *digital signatures.* If Alice performs a private-key operation on an electronic object O (usually via hash and padding functions) to yield a signature $S(O)$, then those who believe in the PKI can verify that $S(O)$ came from O via Alice's public key, and thus conclude that Alice generated this signature.

Typically, O itself may consist of an object O_1 and a field I indicating intention, such as "Alice approves O_1" or "Alice witnesses that O_1 arrived by some point in time". In the process of signing, Alice takes O_1, adds I, and signs the result: indicating her approval or witness.

In the non-digital world, people must frequently take such action on *paper* documents: e.g., signing forms and expense sheets and contracts; recording when a bid or homework assignments was submitted. The last decade has seen a revolution: these paper documents have migrated into electronic settings; rather than dealing with a paper expense form, we deal with an Excel spreadsheet. This change in media permitted a revolution in the ease and speed of creating and sharing documents, even between parties on opposite sides of the Internet.

The natural question arises: since we often need to sign paper documents, and a PKI would permit a nice

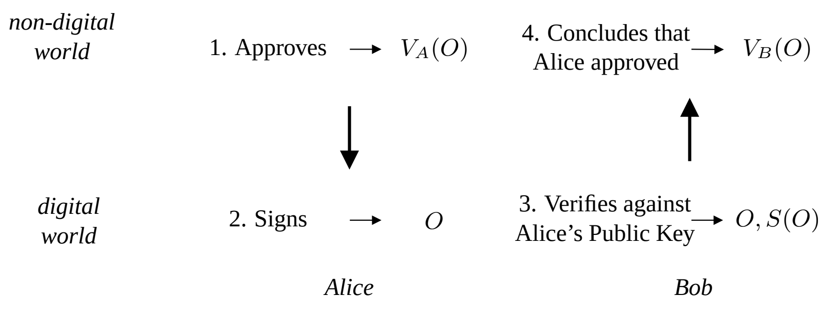


Figure 2.1: Current PKI/workflow integration attempts to capture the non-digital process of signing paper, by digitally signing the corresponding electronic object. To "sign" the virtual document $V_A(O)$, Alice digitally signs the corresponding electronic object O. If Bob receives O and $S(O)$ and verifies this signature, then he concludes that Alice approved the virtual document $V_B(O)$ that he sees.

way to sign electronic objects, can we compose the two, and indicate personal approval of a "virtual" paper document, by digitally signing the corresponding electronic object? The ability to do this composition is often a main motivating factor in deploying enterprise PKI (such as ACES [17]); businesses and products exist to provide exactly these services. Many have been surveyed [2, 3, 5, 11, 12] for this thesis.

Typical electronic workflow tools replace a paper document with an electronic object O, yielding a virtual piece of paper when a party opens the object. When Alice signs an object O, she commits to the virtual piece of paper she sees when she views the object. By PKI, when Bob verifies $S(O)$, he concludes that Alice digitally signed O. By composition—if we accept that digital signatures imply workflow approval—Bob concludes that Alice approves the virtual paper document that O represents. Figure 2.1 sketches this scenario.

This thesis investigates whether this composition actually works. The particular angle that was questionable is the implicit assumption that electronic objects generate semantically equivalent virtual documents each time they are opened. If $V_B(O)$ can be made to be substantially different from $V_A(O)$, then Bob's conclusion about what Alice signed will differ from what Alice thought she signed.

2.2 Related Work

My work was motivated by planned deployment of real applications that used digital signatures on electronic documents. We consequently began investigating the extent to which these these document formats were malleable, and to the surprising degree to which COTS PKI packages tolerated this malleability.

However, these issues certainly have an older history.

1. The German digital signature act came into force in the August of 1997 even before the EU's Directive, in fact the directive was adjusted to conform to the German Digital signature Act [15].

2. In the December of 1997 the European Union invited a proposal for its electronic signature directive [4] with the purpose of making electronic signatures at least as binding as paper-based signatures in a bid to facilitate "free movement of goods and service in the internal market". The purpose of the directive was also to build trust in this new medium of business. The directive dealt with the truth that documents today have various means to introduce dynamic content which causes the document to become malleable.

3. Herzberg [8] in a private communication suggested signing the viewing program as well as the document, insuring that the data displayed is viewed as it was intended.

4. Austria fully implemented the directive [1] in 1999; and concretized the malleability problem by specifying that only data formats may be used which have an "available specification" and which exclude "dynamic changes" or "invisibilities."

5. Ulrich Pordesch [13, 14] a German researcher viewed it a risk to have other agencies verify and sign a document, "imbedding and using the schemes in application systems involves considerable risks, in particular, if the signer or the verifier uses an application environment which is maintained, used, or controlled, by other persons or organizations." He considered personal signature verification applications that the user could himself administer and verify signatures. His scheme involved having a *Personal Digital Assistant* to which documents could be transmitted for scrutinization. This scenario would behave like a safe room where the documents are inspected and verified. This strategy though avoids the problem of how secure the Personal Digital Assistant is and whether it has the capability to process dynamic content.

6. Concurrent with the conference publication [10] of the main results of this thesis, Audun Jϕsang, a senior research scientist with the "Distributed Systems Technology Centre", published [9] a very interesting paper based on the same area but with orthogonal results. Audun Jϕsang's approach differed in terms of the depth and direction of attacks proposed. In his example of changing content based on browser type he used the differences in handling of HTML tags in *Netscape Communicator* and in

Internet Explorer. Thus showing that there are many ways to carry out attacks. In this paper he also considered various attacks on XML signatures. “Not only can the same XML document look different in two different applications, but different XML documents can look the same in the same application”.

Chapter 3

Attack Principles

3.1 Introduction

An attack on a digitally signed document would imply any action on behalf of the attacker that would directly or indirectly change the contents of the electronic document to benefit himself without invalidating the digital signature. In this chapter I will talk about the scenarios in which attacks on digitally signed documents would be feasible and upon what conditions these attacks could be structured to execute.

3.2 Motivation

Whenever someone performs a vulnerability analysis, and discovers some avenue that would cause the system to engage in unexpected ways, the typical response encountered is that no one would attempt to do that. Consequently, we begin by pro-actively addressing this concern. Within our university, two applications motivating campus PKI are *timestamped homework submission* and *signing of payroll and expense forms*.

- The timestamped homework application typically posits that a student Alice submits homework to an automatic system Bob that appends the current time, digitally signs the result, and forwards both on to the instructor Cathy. (One could also easily imagine a system using more advanced timestamping techniques. [7].)

Suppose student Alice could construct an electronic document O that, whenever it was viewed, first consulted a remote file or Web location, and rendered a virtual document based on the contents of that remote file R. Alice could then completely subvert the purpose of timestamping by submitting O before the assignment deadline, but continuing to work on R up to the time the project was graded. If the assignment is one where the instructor Cathy posts sample solutions before grading, Alice could guarantee herself high scores by preparing R so that it can change its contents to incorporate the posted solutions.

- In the typical processing of expense forms, a submitter Alice sends a form to an approver Bob, who then sends it to Cathy, who actually issues checks. Suppose malicious Alice has two sets of numbers: a set S_1 with illegitimate expenses that Bob would not approve but which would result in Cathy issuing a large check, and a set S_2 with smaller numbers that Bob would approve. If Alice could construct an electronic document O that displays S_2 when viewed by Bob but S_1 when viewed by Cathy, then Cathy will receive a Bob-approved expense form indicating S_1, and Alice will receive a larger reimbursement than she should.

In the above two examples, Alice benefits by obtaining Bob's signature on an electronic object that, when viewed in some contexts, displays a virtual document that Bob would not have signed. Scenarios also exist where Alice could benefit by being able to retroactively change documents to which she previously committed. For example, Alice might submit a signed bid to provide (or purchase) services, and might benefit from retroactively changing the terms of that bid. Other scenarios of digital signature include the US government's *ACES project* (which gives citizens an electronic ID), legal documents, etc. Thus subversion of an electronic signature would indeed be a profitable avenue for the attacker to explore.

3.3 Attack Taxonomy

A taxonomy of attacks that would be worth considering is sketched here. Many parameters that could be involved in an attack by an adversary to construct malleable documents are considered.

3.3.1 Hidden Parameters

This avenue of attack works when the virtual document that appears when someone views an electronic object O is not completely determined by O alone, but instead depends in part on other parameters. One way to characterize attack strategies is to consider these parameters. We offer some:

- **Time.** An attack involving usefully changing the content of the virtual document based on the time the object is viewed.
- **Viewer Data.** Suitably changing the content of the virtual document depending on the identity of the viewer, their machine, their operating system, or other such context data advantageous attacker.
- **Viewer Action.** Changing the content of the virtual document depending on actions the viewer takes to benefit the attacker.
- **Remote Control.** Usefully changing the content of the virtual document depending on the existence or contents of a file controllable by the adversary. A secondary issue here would be the proximity of this file to the viewer: a strategy that permitted the file to be an arbitrary URL would require only that the attacker control a web site, but also would require that the viewer have a good connection to the Web (in some cases, we have been able to overcome this restriction by using a 1x1-pixel image to pre-load the required document in the viewer's cache). Having the file to be closer to the viewer may require the attacker to have greater access.

Note that a Web-based remote control attack can potentially be used to mount a viewer-specific attack, if the viewer visits from a predictable host.

3.3.2 Fraudulent Content

In these attack scenarios, could the adversary devise ways to usefully change the apparent content of a signed document? The question then arises of *when* the adversary must determine this alternate content. Two natural choices suggest themselves:

- **Pre-signature.** The alternate content must be fixed at the time the signature is applied.

- **Post-signature.** The alternate content may be chosen at some point after the signature has been applied.

3.3.3 Nature of Change

Another parameter is how the display of alternate content affects the “current” electronic document.

- **Static Content.** The most powerful attack strategy is one where viewing alternate content does not change the working object.
- **Dynamic Content.** A strategy that requires the working object (e.g., the Word document the viewer opened) to be modified as part of displaying alternate content can still be effective, but only when the signature is verified against the original object.
- **Dynamic Content and Signature.** It is also conceivable that an attack strategy might change signatures at the same time it changes the contents of the object, perhaps by shipping pre-established object-signature pairs.

3.3.4 Other Angles

The above mentioned paratemers are intended simply as an illustrative taxonomy, not as a complete one. In particular, we also want to leave open other avenues for attack. For one example:

- **Spoofed Signature**. In standard Web spoofing [6, 18, 19], the adversary uses the richness of the user interface to create the illusion of the desired result. When attacking signatures,the adversary might use a similar technique: the document might modify itself and invalidate its bona fide signature—but mimic the signature-verification user interface sufficiently well that the user is still convinced the signature is valid.

Chapter 4

Worflow Formats and Attacks

Today with so much variety in colors, fonts, graphics, etc. no one is satisfied with plain ASCII text anymore. In this section, we will consider various popular formats for electronic documents, and the potential to realize the above attack scenarios in these formats.

4.1 Microsoft Word 2000/XP

In Microsoft Word 2000/XP there are many avenues to introduce dynamic content into electronic documents. In this section we will discuss some of the attack strategies discussed in Chapter 3 in a more practical setting. In particular these attacks shall show how difficult it is to completely remove all malicious dynamic content from an electronic document.

4.1.1 Macros

Since releasing 6.0, Microsoft Word has permitted users to add active content to documents via *macros*. The most common approach to add active content would involve using Macros to carry out a *remote control, post-signature* attack. With some trial and error, this is easily done: with the opening of the document, a

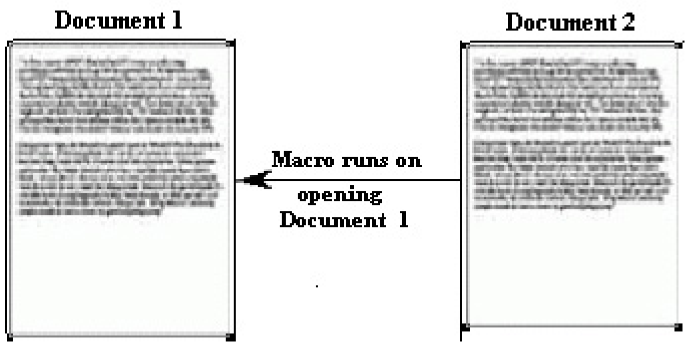


Figure 4.1: Microsoft Word macros

associated macro is run that replaces the current contents with the contents of a remote file.

```
Private Sub Document_Open()
Set doc = Documents.Open(URL of file)
Set rng2 = doc.Range
Documents("signed.doc").Activate
Set rng = ActiveDocument.Range
rng.FormattedText = rng2.FormattedText
doc.Activate
ActiveDocument.Close
```

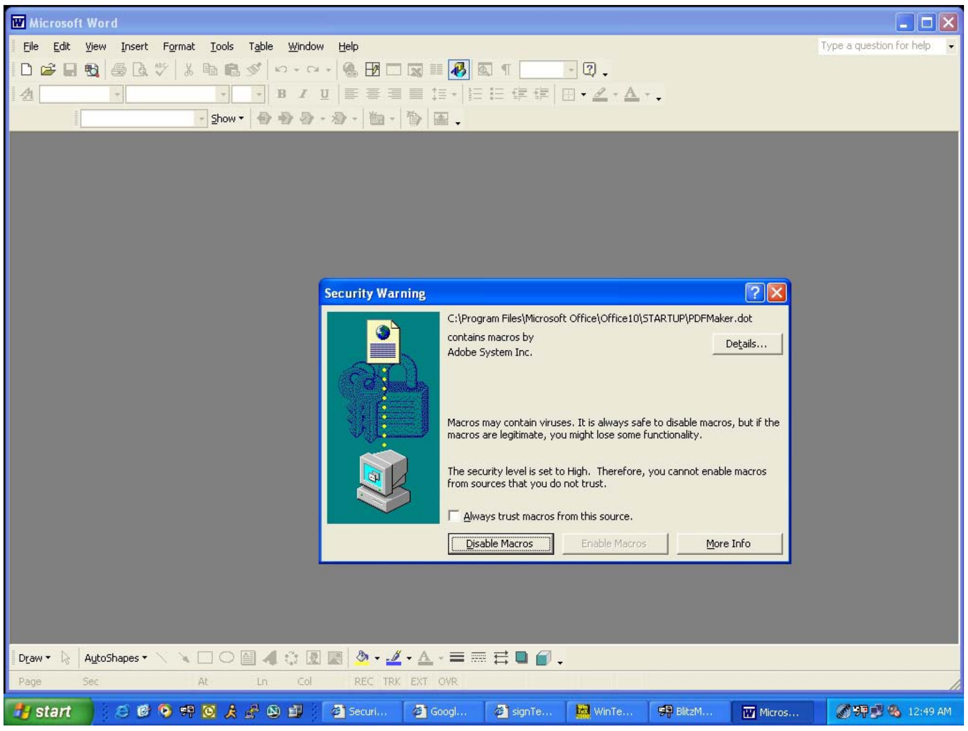
A *Document_Open()* function is called whenever a document is opened in Word. The code demonstrates how the content of the document being opened is replaced with another document over the internet.

This is a *dynamic content* attack that changes the file contents.

Signed Macros in Office 2000/XP

The danger of having macros in a Microsoft Office document are well know and thus everyone sets the security setting of Microsoft Office documents to High. But with the introduction of signed macros in Microsoft Office 2000, using macros to conduct attacks to digitally signed Microsoft Office documents has become feasible.

In Microsoft Word 2000/XP a signed macro that is accepted by a user and allowed to execute will never raise a warning window again. Thus this provides a convenient attack scenario. One can sign a macro that



C:\Program Files\Microsoft Office\Office10\STARTUP\PDFMaker.dot contains macros by Adobe System Inc.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

The security level is set to High. Therefore, you cannot enable macros from sources that you do not trust.

☐ Always trust macros from this source.

Figure 4.2: Screenshot of initial warning window to accept the macro from a “Trusted Source”

manipulates document content, and once this macro is approved by the user, it is executed even with the security setting set to High without giving any warning message. This allows devious ways to fool a user. This is shown in the screenshot Figure 4.2.

The twist in this strategy comes with a utility provided by Microsoft Office called *Selfcert.exe*. It is possible to create a self generated certificate using the *Selfcert.exe* application and create a certificate capable of code signing. The utility also allows arbitrary names to be selected as the party the certificate is issued to, thus an attacker could choose names like ”Microsoft Corporation”, ”Dartmouth College PKI Lab” etc to trick the user into accepting the signed macro.

This approach to macro signer certificate management means that, once a user decides to trust a vendor for some specific plug-in, they give that vendor the right to forge signatures in perpetuity.

4.1.2 Fields

One set of promising techniques is the *Insert Field* feature in Word. Fields are different from Macros in the sense the code is embedded inside the text and it needs to be updated to reflect any change in it. Fields only allow very specific code to be inserted into a document thus the scope of their use is limited. From the *Insert* menu, if one selects *Field*, one is given a rich set of fields and operators with which to build active content.

For example, one can carry out a *time* attack by using the conditional *IF* operator on the *DATE* field. In the fragment below, the author revises his testimony after the 16th. (A more complex conjunction operation would give us month and year checks as well.)

```
{ IF { DATE \@ "d" \* MERGEFORMAT } < 16
"I did not have" "I did have"}
```

Installing field code is tricky: one cannot simply “type” the code into the field box. For code such as the example, one must first select the *IF* operator from the menu, then *within* the resulting code, one can insert the *DATE* field. This scenario is sketched in *Figure 6.1– Figure 6.5*.

The above attack has the limitation of being *pre-signature* but the advantage of being *static content*: the binary document appears to contain the entire conditional, and does not appear to change depending on the branch taken.

Fields also offer promising hooks such as *USERNAME*.

There have been some changes in the Field update policy. In Microsoft Office 2000 the *Date* and *Time* were auto update fields and others had to be manually updated but with Microsoft Office XP all fields were made manual update. Auto update fields are very useful to the attacker because the field contents change according to the conditional statement inserted

Fields can also be updated automatically by via macros, but that reduces us to the case discussed in Section 4.1.1 by relying on macros.

```
Sub UpdateAllFields()
   Dim aStory As Range
   Dim aField As Field
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

4.1.3 Links

Somewhat unexpectedly, Office documents allow users to insert material from remote documents by reference. To do this in Microsoft Word the user copies text from another Microsoft Office document (eg. Word, Excel, Powerpoint etc.); then, in the target document, the user selects *Edit*, then *Paste Special*, then *Paste Link*, then pastes the text in as a *link* (*unformatted text* works nicely).

This approach permits a *post-signature, remote control* attack. Unfortunately, this also appears to be a *dynamic content* attack—Word asks whether to save the document.

A surprising side-effect of the existence of this feature in Word is that a remote Web site can track each time one reads a document, and even plant cookies. The cookie being an invisible 1×1 pixel image file inserted into the Microsoft Word document via *Paste Special*. (The University of Denver [16] has also noticed this "feature".)

4.2 Microsoft Excel 2000/XP

In this section I will talk about introducing dynamic content into the popular spread sheet package *Microsoft Excel*.

4.2.1 Macros

Similar to Word, Excel also has powerful Macro capabilities, however the plausibility of Excel macros as an attack vector might be greater than Word's. Many organizations use macro-laden spreadsheets as standard practice. For example, universities preparing grant proposals for the *National Science Foundation (NSF)* are required to download Excel spreadsheets with Macros. Since these spreadsheets typically get routed throughout university administrative staff, at least one large population is primed to always hit the “accept macros” button. This makes the feasibility of this attack quite high.

4.2.2 Time and Date

In Excel, the user can associate functional behavior with specific cells. This behavior has interesting potential for our purposes.

For example, an attacker can mount a *pre-signature*, *time-based* attack by selecting *Insert*, then *Function*, then building an *IF* construct using *NOW( )*. A simple example:

```
IF(DAY(NOW())<16, 2000,20000)
```

Unfortunately, this is a *dynamic content* attack, since Excel appears to notice the cell is “volatile”. Which means it does notice that the contents of the cell are dynamic, and asks whether the document should be saved. This scenario is sketched in *Figure 6.33– Figure* **??**.

4.2.3 Operating System

It is possible to use functions in Microsoft Excel to mount *viewer-data* attacks—although the most useful viewer data I found was mounting attacks based on operating system. (or perhaps file path name).

For example:

```
IF(INFO("osversion")<>"Windows (32-bit) NT 5.00",
"I love Linus","I love Bill")
```

Again, this attack is *pre-signature*, but also is *dynamic-content.*

4.2.4 Links

The same techniques of Section 4.1.3 apply to Excel as well: the attacker can copy data from a Word or Excel file under his control, then *Paste Special* a link into an Excel cell. As long as the Word file is tab delimited it seems to place the data very accurately. As before, this enables a *remote-control*, *post-signature*, *dynamic-content* attack.

Unfortunately, this appears to trigger a warning each time the file is opened. Options exist to disable this warning (*Tools*, then *options*, then *Edit*), but these appear to remain with the installation of Excel, rather than traveling with the file.

4.2.5 External Queries

Excel includes features to make explicit queries to remote files. These features enable *post-signature*, *remote control* attacks that have *dynamic content*.

From the *Data* menu, the attacker can select *Get External Data* and then set up a query to a remote text file. The text file should be written with tab spaces between words to specify different fields in the spreadsheet. By right-clicking on the cell and selecting *Data Range Properties*, the attacker can configure the query to update on open or even regularly (in the background).

Sometimes, this technique gives a warning and an external data toolbar pop-up, which indicates that Excel realizes that there is external data involved.

4.3 Adobe Acrobat PDF

Adobe's *Portable Document Format (PDF)* is fast becoming an alternative to Word as the *de facto* standard for electronic documents.

Various tools exist to view PDF and to convert other formats into PDF; however, using the official Acrobat 5 product, one can more directly investigate nuances in creating PDF documents.

With this flexibility, and with Acrobat's use of Javascript with event-driven actions, I have been able to explore attacks based on Viewer Action (although I have not been able to carry out a remote-control attack here).

4.3.1 Time

Using some Javascript functions I was able to make attacks similar to our time-based attacks on Word and Excel.

For example, one can use the *form* toolbar to create a form field, and then add Javascript code in its *calculate* field to change the value of the field according to the date.

```
var f = 9000
var g = util.printd("d", new Date())
if(g < X) f = 5000
event.value = f
```

The variable `g` here removes the day out of the date value, and the `X` represents some day against which we want to check the value of `g`. In this type of form field, I can make it appear without a border—like an ordinary text. I was also able to make it "Read-only" using the *appearance* tab, which means that the user cannot differentiate between normal text and the form fields.

4.3.2 Viewer Action

Above, I made a form that appeared to be text. I can also make a form that is invisible—but which has Javascript associated with viewer actions.

For example, when the mouse moves over the form, I can trigger Javascript to change the value of another field. If one were to click on the *form creation* tool on the tool bar and use the mouse to create a box in the document by dragging it across the screen to the size we want the form box to be. Then click the *mouse enters* from the *Actions* bookmark, and select *Run Javascript* from the drag-down menu. I could then use the Javascript to give some form named "danger" another value.

```
event.value = 1
```

4.4 HTML mail

As with documents, people are not satisfied with just plain ASCII mail, and instead want mail content to incorporate feature such as colors, different fonts, and pictures. The MIME standard permits email to be formatted as HTML, and consequently many popular clients (such as Yahoo Mail, Hotmail, iname.com, and Mailcity) do this.

HTML email provides a rich breeding ground for a variety of attacks. To test these techniques, one needs to discover how to convince one's mail client to send an arbitrary HTML file as MIME mail; in Microsoft Outlook, when sending new mail, one clicks on the *view* menu item, then selects *source edit*, then selects the *HTML* tab.

4.4.1 Date

Using JavaScript as a tool one can carry out a variety of attacks in this area.

For example, It is possible to change the content of the document with a change in date by using the `document.write()` method which allows us to write a new page using HTML tags. With many mail clients (including Netscape and Outlook) if the user attempts to look at the HTML source for this mail they only see the final HTML—and not the Javascript with the dynamic content.

```
<HTML>
<HEAD>
<TITLE>HTML Mail</TITLE>
<SCRIPT language=JavaScript >
function InitForm()
{
var today = new Date()
var day = today.getDate()
if (day > 8)
{
 document.write("<PRE><BR>Kunal    4000<BR>Sean    8000<BR></PRE>")
}
else {
 document.write("<PRE><BR>Kunal    8000<BR>Sean    16000<BR></PRE>")
}
}
</SCRIPT>
```

```
</HEAD>
<BODY onLoad="InitForm();">
<PRE>
</PRE>
</BODY>
</HTML>
```

4.4.2 Remote Control and Viewer-Specific Attacks

Another type of attack that one can carry out is include, in the HTML, references to inline images which reside on a remote system. This technique permits post-signature remote-control attacks (since we can change the images after the fact) as well as viewer-specific attacks (since our Web server can track which user is requesting the images).

I was able to successfully carry out this attack with an image that looked like text.

```
<HTML>
<HEAD>
<TITLE>HTML Mail</TITLE>
<BODY>
<PRE>
<IMG name=thumbnail src="http://www.cs.dartmouth.edu/˜kunal/trial1.jpg">
</PRE>
</BODY>
</HTML>
```

As noted, it is possible to get all kinds of information from already existing methods from the browser type to the operating system being used.

We can do this at the client-side by using Javascript:

```
<HTML>
<HEAD>
<TITLE>HTML Mail</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function InitForm()
{
if (navigator.appName == "Microsoft Internet Explorer")
{
 document.write("<PRE><BR>Kunal    4000<BR>Sean    10000<BR></PRE>")
}
else  {
 document.write("<PRE><BR>Kunal    8000<BR>Sean    16000<BR></PRE>")
}
}
</SCRIPT>
</HEAD>
<BODY onLoad="InitForm();">
<PRE>
</PRE>
</BODY>
</HTML>
```

Using Microsoft Outlook, which employs Microsoft Internet Explorer to view HTML mail, I was able to successfully carry out this test.

4.5 Netscape Communicator signed forms

Netscape Communicator 4.04 provided the ability to associate a digital signature with data generated as the result of a transaction, such as a purchase order or other financial document. This is extremely useful as it provides a standards-based way to digitally sign forms or other transaction-related text on the Web. Called Form Signing, this technology allows web sites to ask users to digitally sign transactions, thus providing the higher level of assurance that a verifiable digital signature brings.

After the data has been signed and both the signature and the data have been sent across the network, a CGI script running on the server can use the Signature Verification Tool(provided by Netscape) to extract the digital signature and validate it.

These signed forms allow us to insert JavaScript and conduct the HTML mail attacks as described in Section 4.4. Newer versions of Netscape do not seem to support this. We successfully tried the remote control attack described in Section 4.4.2, using an image. This attack would have the user sign an image which would reside on website which we could then change.

Chapter 5

PKI packages

In this section I will introduce some of the packages I looked at for signing electronic documents, and discuss how the attacks proposed here affected electronic documents signed using these packages.

5.1 External PKI

One approach is to use a PKI package that is completely oblivious to documents it is signing. For example, one might use PGP to sign and verify documents sent as email attachments. For another example, we obtained a certificate from Digital Signature Trust [2] and used this certificate with their *CertainSend* application to sign, transmit, and verify signatures on electronic documents. With these approaches, the attack strategies on Microsoft Word discussed in Section 4.1, Microsoft Excel discussed in Section 4.2, and Adobe Acrobat PDF discussed in Section 4.3 would effectively function.

I also worked with S/MIME and found it vulnerable to the HTML email attacks. Experimenting with Outlook and using its built-in signature and encryption features, I was successfully able to carry out the attacks proposed in Section 4.4.

5.2 Assured Office

Another approach is to use a PKI package that is explicitly integrated with the document software. For example, the *Assured Office* product from E-Lock Technologies [5] adds sign and verify buttons in one's Office installation, and uses a key pair resident in the user's browser to sign documents.

With this approach all of the Word and Excel attacks I have outlined thus far work. Even the dynamic-content attacks work, because the signature is verified at document open, before the changes have been made.

(E-Lock had been acquired by Lexign [11], and the *Assured Office* product has been renamed *Prosigner*. Lexign recently sold the *Prosigner* package to Frontier Technologies Corporation and it is called *E-Lock*.)

I tried all the attacks discussed in Section 4.1 and Section 4.2 with the latest version of E-Lock *Prosigner* and they still work even with repeated notification of these facts to E-Lock.

5.3 Acrobat Signed PDF

Adobe Acrobat has two types of signature features built in: visible and invisible signatures.

5.3.1 Visible Signatures

In Adobe Acrobat's *visible signatures*, the document's signature is visible as an image that indicates whether the signature has been verified and whether the document was modified since. My time-based attack of Section 4.3.1 succeeded (although some versions with updates every second seemed to change the document before the signature was verified).

In theory, one could also use the techniques of Section 4.3.2 to add mouse-over action to the digital signature field—for example, to change a global value when the user verifies the signature, so interesting things can happen after the signature is verified.

The visible signature approach is also potentially vulnerable to plain old spoofing: including an image that looks just like a valid, verified signature.

5.3.2 Invisible Signatures

In Acrobat's *invisible signatures*, the signature is applied as an invisible tag. This technique works extremely well against the attacks I have proposed since the *image* of the document can easily be matched after a change in the file content with the previous image and the changes can be easily noticed.

5.4 Utimaco

Another product we discovered is the *SafeGuard Sign&Crypt for Office* from Utimaco Safeware [3], in Germany. Although we have not been able to obtain sample tools, even after repeated requests to the respective personnel, we was very pleased to notice that their online documentation indicates that they were concerned about "invisible dynamic content," and the screenshots indicate they appear to sign and verify TIFF images of documents. We would speculate that the attack strategies outlined in this paper would not be effective here. This strategy also has its drawbacks as it removes all the flexibility of signed electronic documents.

5.5 Silanis

Silanis [12] of Canada makes a *ApproveIt* package for Word and Excel documents. In my tests *ApproveIt* seems to be aware of most of our attempts to change document contents and blocks the field contents to be updated. However, our tests also showed that *ApproveIt* still permits macro-based attacks. This creates a problem as, once they are accepted, Microsoft allows signed macros in its Office documents to execute without a warning window, as described in Section 4.1.1

5.6 Office XP built-in signatures

Office XP in its first release provided ways to digitally sign electronic documents and like Office 2000 allows the digital signing of macros in Office XP documents. These digitally signed macros are assumed to be secure and thus allowed to execute without any warning.

The "auto-update" feature in signed documents, using Office XP's built-in signing application, has been removed, but using the signed macro attack all the attacks proposed work if the user accepts the initial claim of the macro being from a trusted source.

5.7 Summary of Attacks

This is a summary of all the attacks discussed in Chapter 4 applied to some COTS PKI packages discussed in Chapter 5.

Table 5.1: Summary of attacks on Office 2000/XP, Adobe Acrobat and Outlook express using COTS PKI

Attack Methods	**Signature Format**	**Workflow format**	**Results**
Time/Date-Based Attacks	E-Lock *Prosigner*, E-Lock *Assured Office*, Digsigtrust *Certainsend*	Microsoft Word/Excel	Successful
	Silanus *ApproveIt*	Microsoft Word/Excel	Unsuccessful
	Outlook internal signatures	HTML Mail via Outlook	Successful
	Visible Adobe Acrobat PDF signatures	Adobe Acrobat PDF	Successful
	Invisible Adobe Acrobat PDF signatures	Adobe Acrobat PDF	Unsuccessful
Macro-Based Attacks • Time/Date • Fields • Linked files	E-Lock *Prosigner*, E-Lock *Assured Office*, Digsigtrust *Certainsend*, Silanus *ApproveIt*	Microsoft Word/Excel	Successful
Linked file Attacks	E-Lock *Prosigner*, E-Lock *Assured Office*, Digsigtrust *Certainsend*, Outlook internal signatures	Microsoft Word/Excel, HTML Mail via Outlook	Successful
	Silanus *ApproveIt*		Unsuccessful
Platform-Based Attacks	E-Lock *Prosigner*, E-Lock *Assured Office*, Digsigtrust *Certainsend*	Microsoft Word/Excel	Successful
	Silanus *ApproveIt*		Unsuccessful

Table 5.2: Summary of attacks on Office XP internal signatures

Attack Methods	**Signature Format**	**Workflow format**	**Results**
Time/Date-Based Attacks	Built-in signatures	Microsoft Word/Excel	Unsuccessful
Macro-Based Attacks • Time/Date • Fields • Linked files	Built-in signatures	Microsoft Word/Excel	Successful
Linked file Attacks	Built-in signatures	Microsoft Word/Excel	Successful
Platform-Based Attacks	Built-in signatures	Microsoft Word/Excel	Unsuccessful

Chapter 6

Case Studies

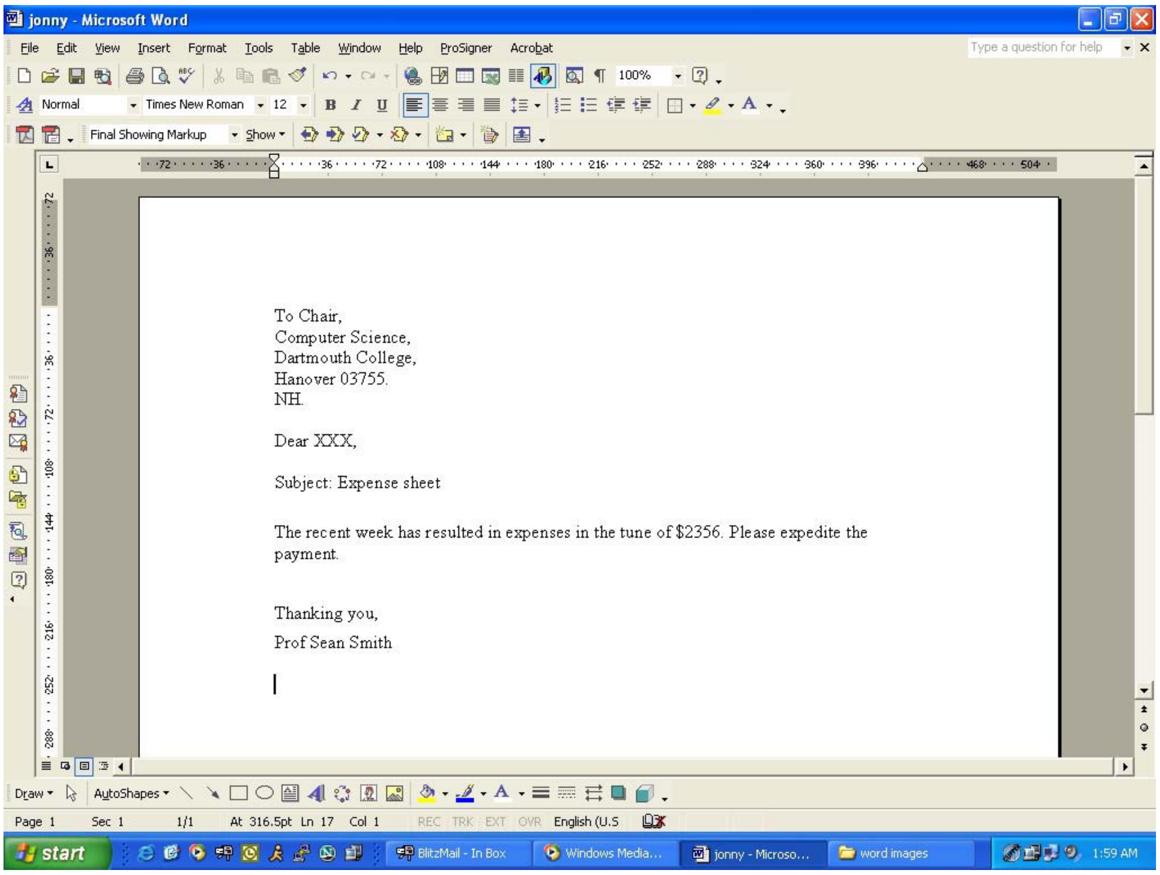
In this section I shall show some of the results produced from the attacks described in Chapter 4.

6.1 An attack on Microsoft Word

Microsoft Word allows many ways to insert dynamic content in a document without using macros. This section describes attacks using Microsoft Word with E-Lock *Prosigner* as the digital signature package.

6.1.1 An attack on Microsoft Word using Fields

As I show, it is easy to carry out an attack in Microsoft Word. In this example, I use the *Insert Field* as demonstrated in Section 4.1.2. Using this attack, if the document were viewed on different days then the amount of money requested changes. Figure 6.1 shows a simple Word document written and signed by Prof. Sean Smith and sent to the head of the department for release of funds. Figure 6.2 shows the same Word document. Figure 6.3 shows the dynamic content in the file. I sign the document using E-lock *Prosigner* and verify it in Figure 6.4. When the document is opened the next day, its contents have changed as shown in Figure 6.5, but *Prosigner* still says the document contents are signed and verified.



To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

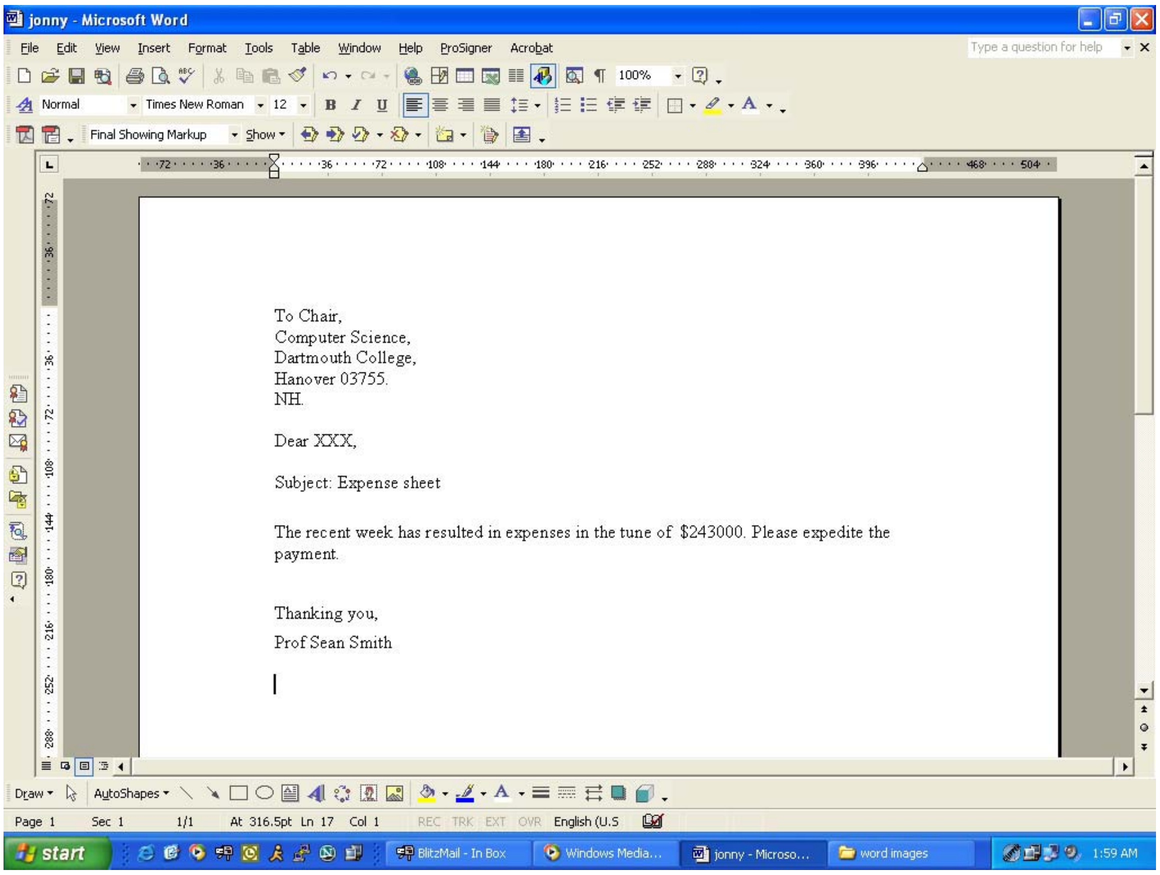
Subject: Expense sheet

The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.1: A simple Microsoft Word document


jonny - Microsoft Word
File Edit View Insert Format Tools Table Window Help ProSigner Acrobat
Normal Times New Roman 12
Final Showing Markup Show
Page 1 Sec 1 1/1 At 316.5pt Ln 17 Col 1 English (U.S
start BlitzMail - In Box Windows Media... jonny - Microso... word images 1:59 AM

To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

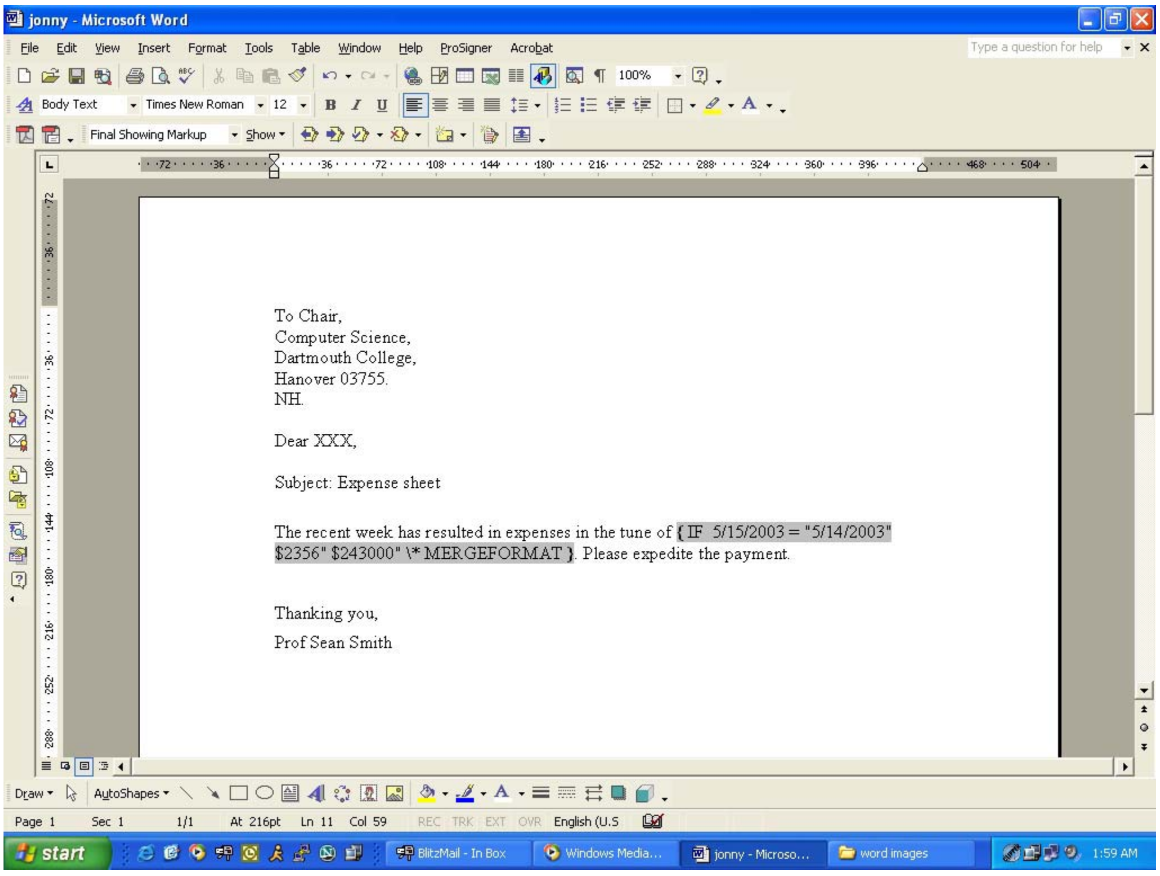
Subject: Expense sheet

The recent week has resulted in expenses in the tune of $243000. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.2: The same Microsoft Word document viewed on a different day



To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

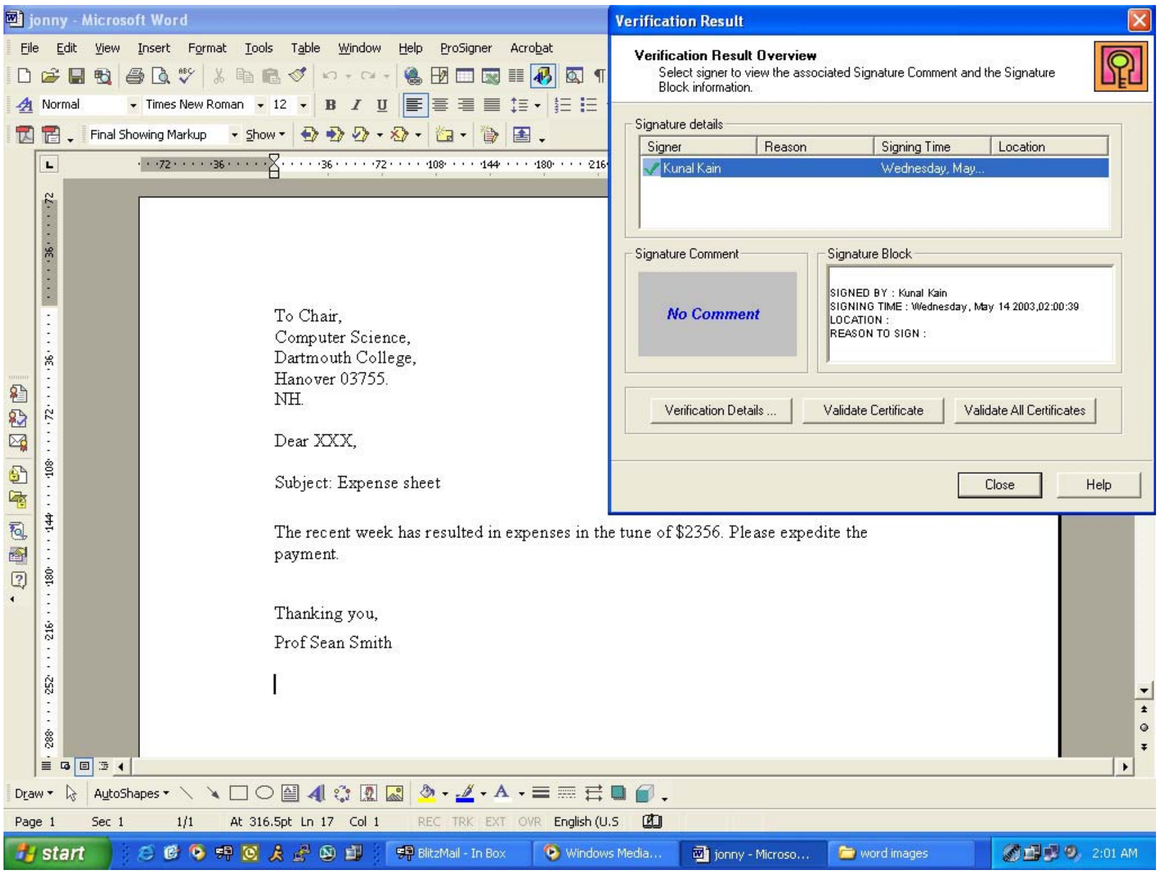
Dear XXX,

Subject: Expense sheet

The recent week has resulted in expenses in the tune of { IF 5/15/2003 = "5/14/2003" "$2356" $243000" * MERGEFORMAT }. Please expedite the payment.

Thanking you,
Prof Sean Smith

Figure 6.3: By right-clicking on the field one can “Toggle Field Codes” to open the field and see the actual inserted code


jonny - Microsoft Word
File Edit View Insert Format Tools Table Window Help ProSigner Acrobat
Verification Result
Verification Details ...
Validate Certificate
Validate All Certificates
Close
Help

Verification Result Overview
Select signer to view the associated Signature Comment and the Signature Block information.

Signature details

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Wednesday, May...	

Signature Comment

No Comment

Signature Block

SIGNED BY : Kunal Kain
SIGNING TIME : Wednesday, May 14 2003,02:00:39
LOCATION :
REASON TO SIGN :

To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

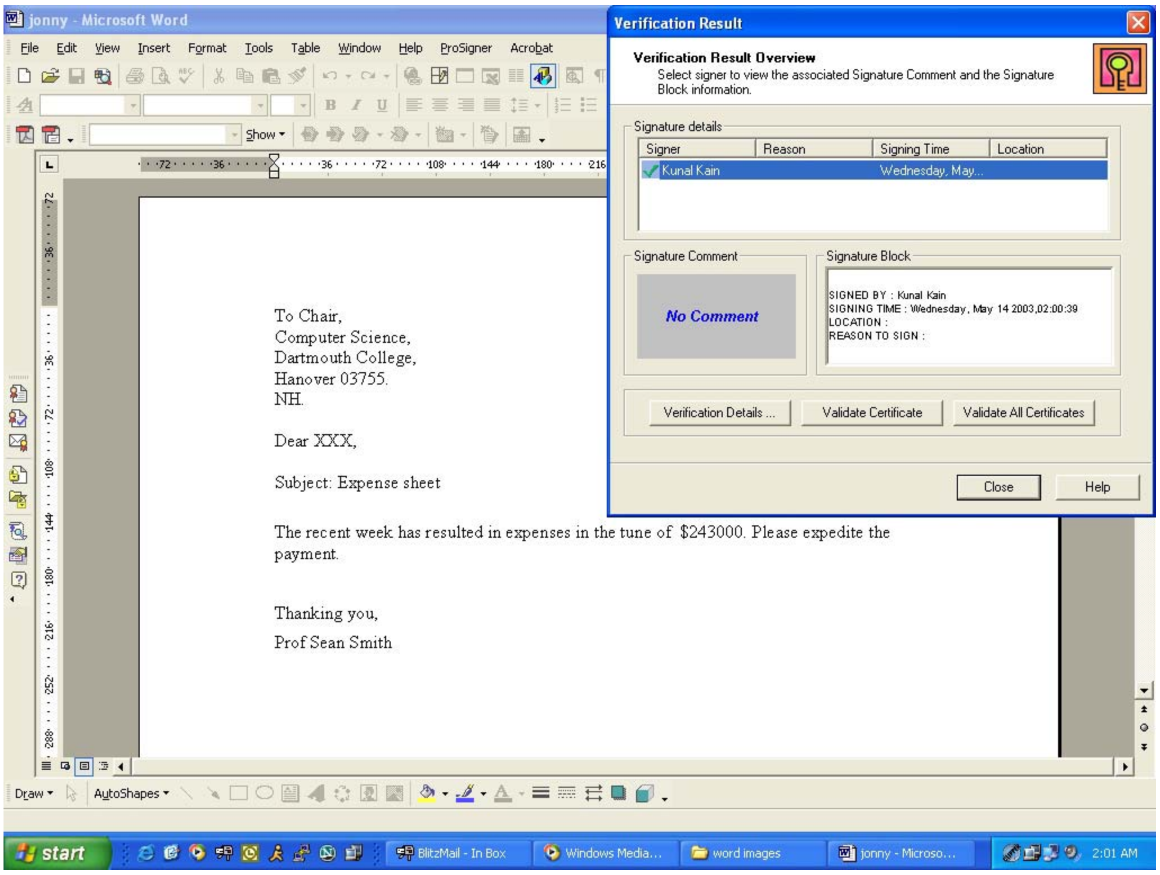
Subject: Expense sheet

The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.4: E-Lock *Prosigner* is used to sign this document and deduce the hidden content



To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

Subject: Expense sheet

The recent week has resulted in expenses in the tune of $243000. Please expedite the payment.

Thanking you,
Prof Sean Smith

Verification Result Overview
Select signer to view the associated Signature Comment and the Signature Block information.

Signature details

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Wednesday, May...	

Signature Comment: No Comment

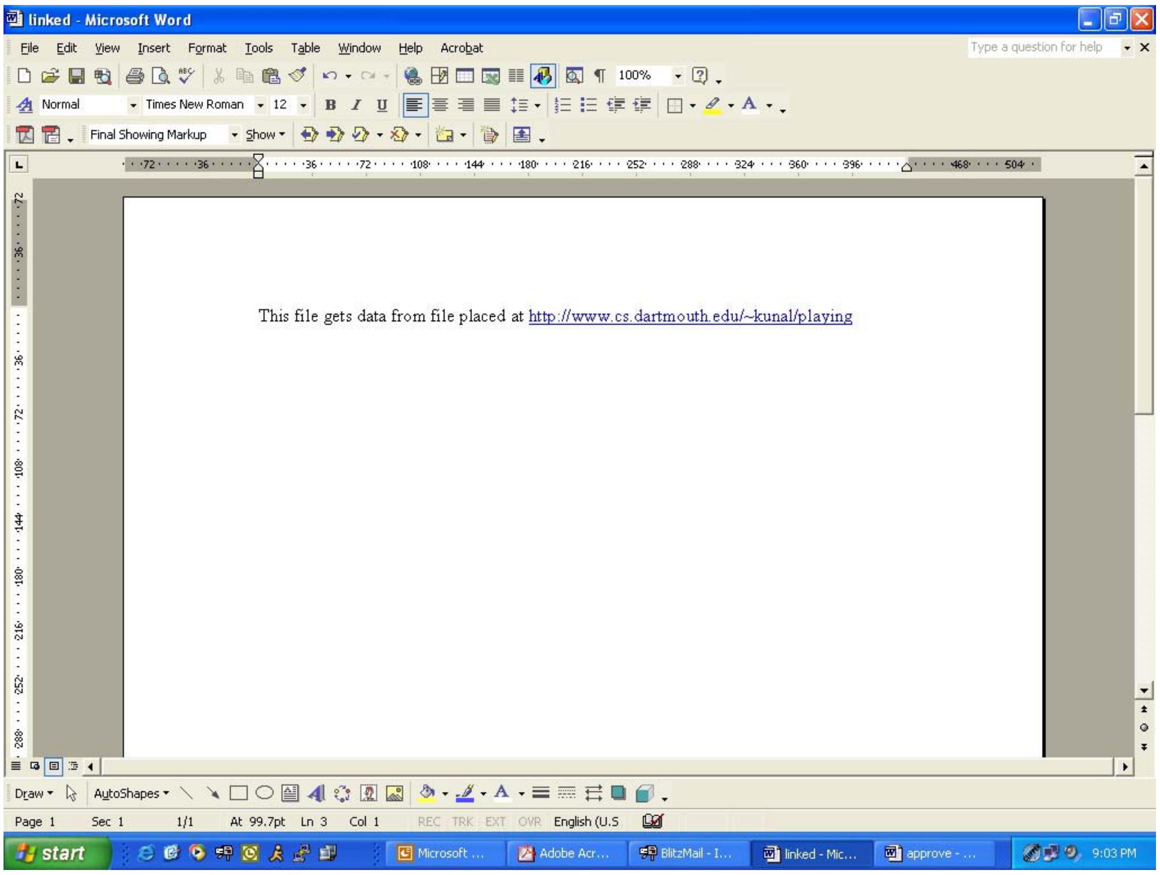
Signature Block:
SIGNED BY : Kunal Kain
SIGNING TIME : Wednesday, May 14 2003,02:00:39
LOCATION :
REASON TO SIGN :

Figure 6.5: E-Lock *Prosigner* fails to do its job

6.1.2 An attack on Microsoft Word using Linked Files

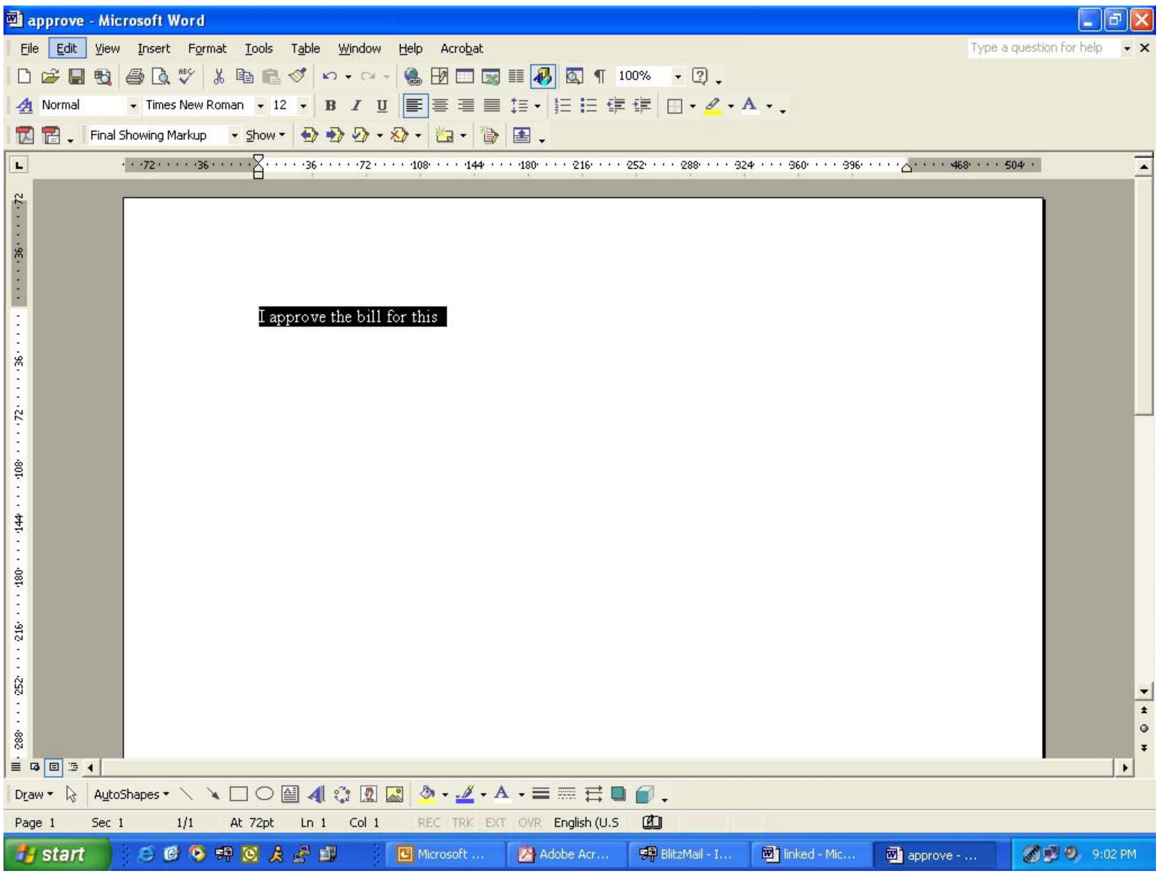
In this section, I show an attack on a Microsoft Word document signed using E-Lock *Prosigner*. The linked file “approve.doc” resides on the internet and whenever the file “linked.doc” is opened, it updates the link and replaces the content with the content of “approve.doc”. In this way, one could modify the content of the linked document and thus change the content of the main document.

Figure 6.6 shows “linked.doc”. Figure 6.7 shows “approve.doc” that resides on a website. I copy the contents of “approve.doc” in Figure 6.8. In “approve.doc”, I *Paste Special* the contents of the clipboard demonstrated in Figure 6.9. I select *Paste as link* in Figure 6.10. In Figure 6.11, I select inserting link as unformatted text. Now, when the document is viewed on *May 14, 2003*, it shows approval but when viewed on on *May 15, 2003* after I modify “approve.doc”, it shows the opposite.



This file gets data from file placed at http://www.cs.dartmouth.edu/~kunal/playing

Figure 6.6: The Microsoft Word document “linked.doc”


approve - Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Normal
Times New Roman
12
100%
Final Showing Markup
Show
Draw
AutoShapes
Page 1 Sec 1 1/1 At 72pt Ln 1 Col 1 REC TRK EXT OVR English (U.S
start
9:02 PM

I approve the bill for this

Figure 6.7: The Microsoft Word document “approve.doc”

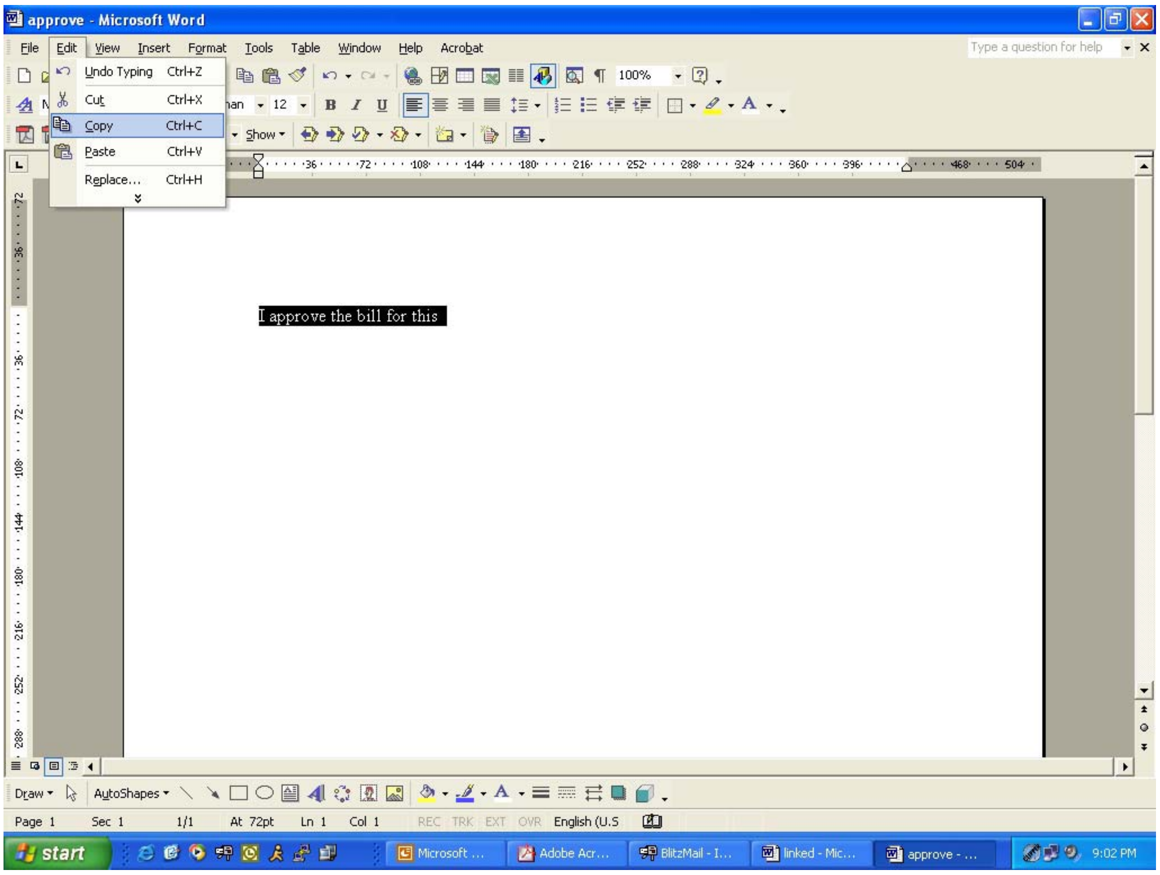
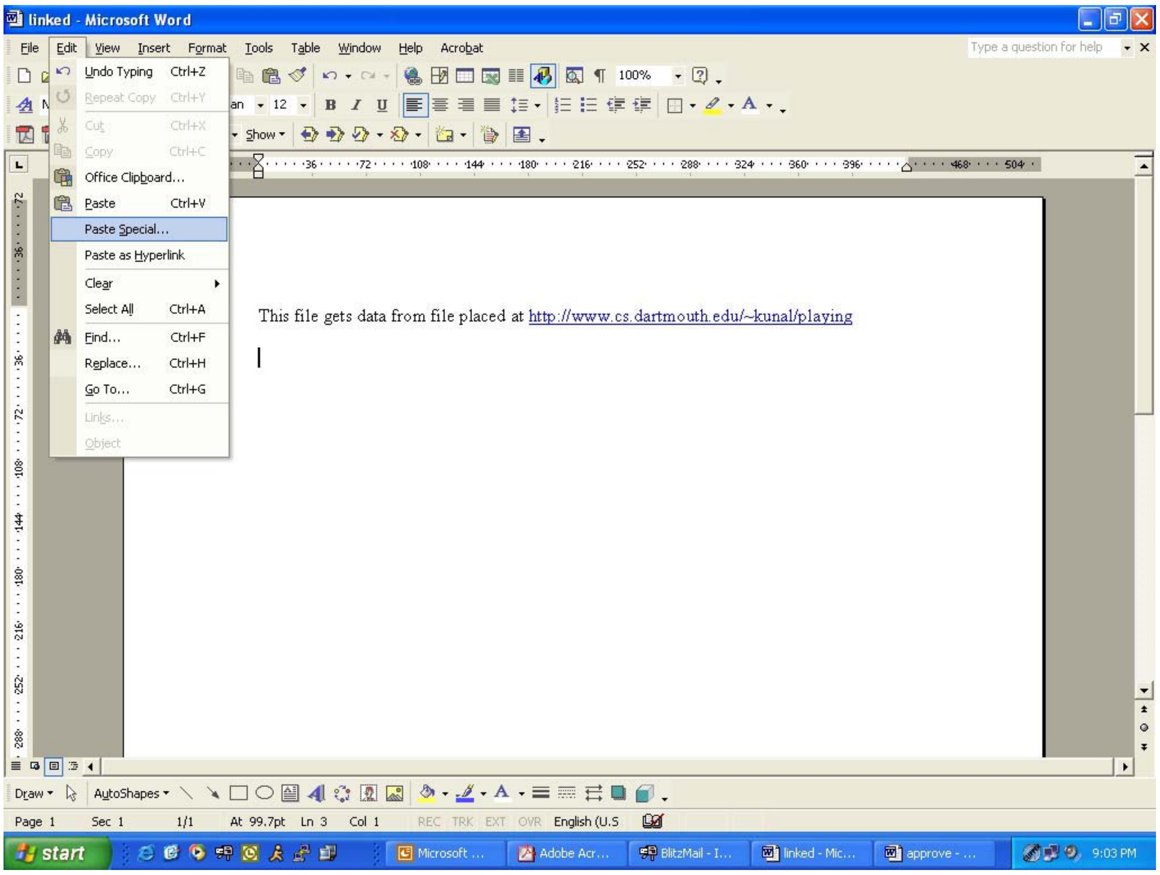
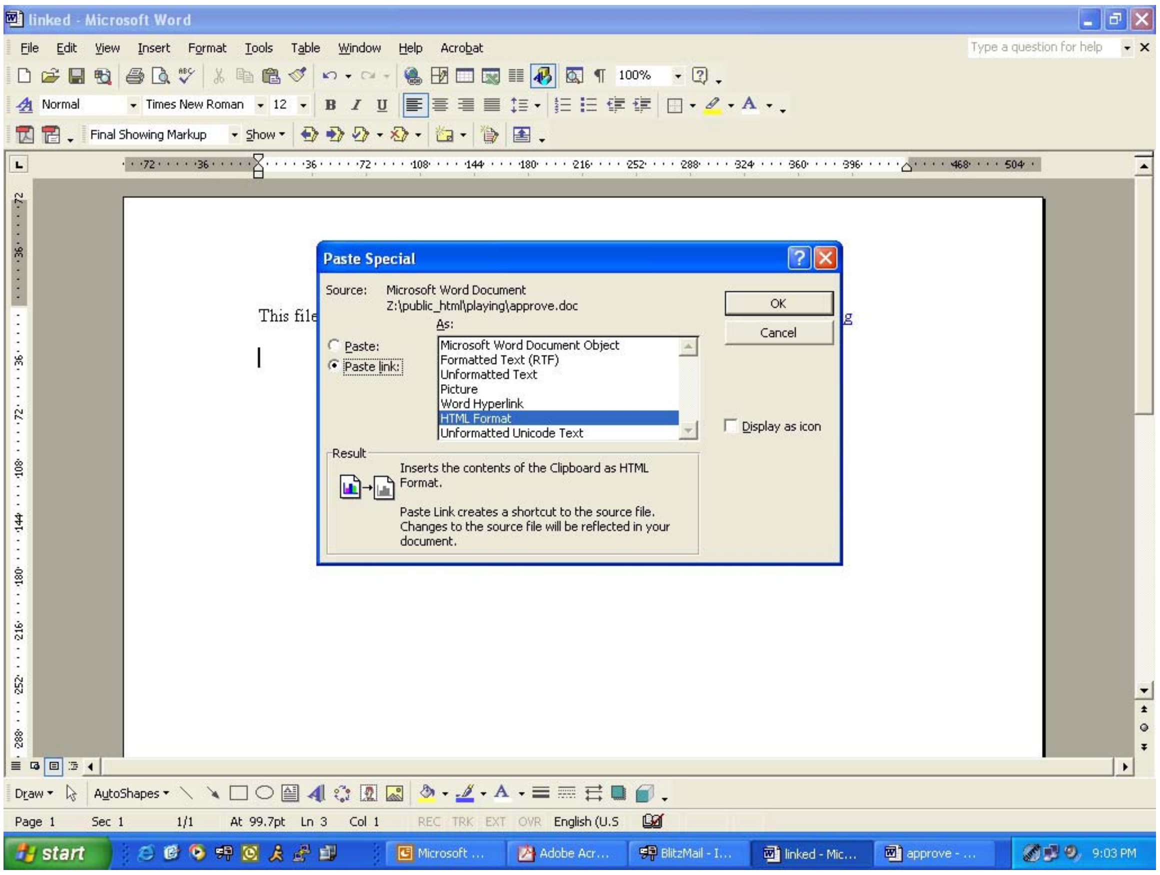


Figure 6.8: Copying content into clipboard



This file gets data from file placed at http://www.cs.dartmouth.edu/~kunal/playing

Figure 6.9: “Paste Special” in “linked.doc”


linked - Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Normal
Times New Roman
12
Final Showing Markup
Show
This file
Draw
AutoShapes
Page 1 Sec 1 1/1 At 99.7pt Ln 3 Col 1 REC TRK EXT OVR English (U.S
start
9:03 PM

Paste Special

Source: Microsoft Word Document
Z:\public_html\playing\approve.doc

As:

( ) Paste:
(●) Paste link:

Microsoft Word Document Object
Formatted Text (RTF)
Unformatted Text
Picture
Word Hyperlink
HTML Format
Unformatted Unicode Text

OK
Cancel

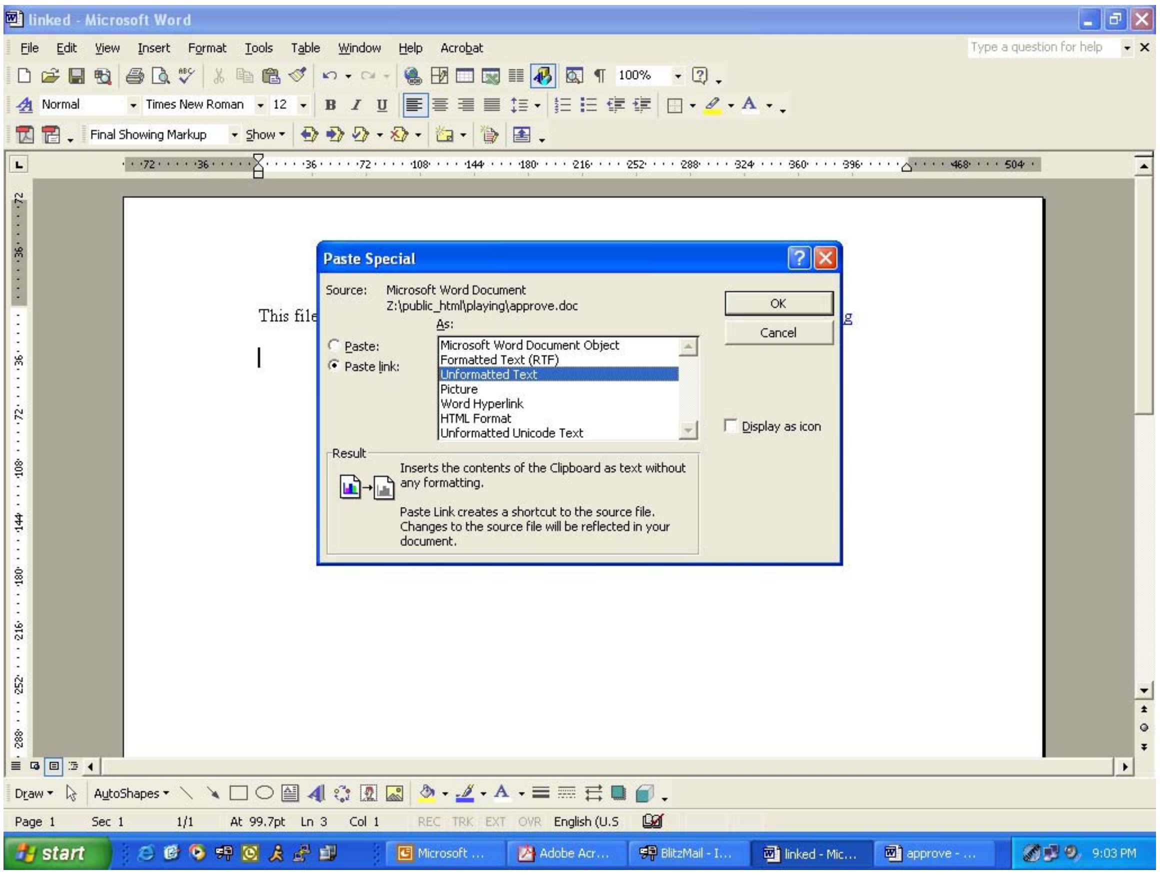
☐ Display as icon

Result

Inserts the contents of the Clipboard as HTML Format.

Paste Link creates a shortcut to the source file. Changes to the source file will be reflected in your document.

Figure 6.10: Click Paste as link



Paste Special

Source: Microsoft Word Document
Z:\public_html\playing\approve.doc

As:

○ Paste:
◉ Paste link:

Microsoft Word Document Object
Formatted Text (RTF)
Unformatted Text
Picture
Word Hyperlink
HTML Format
Unformatted Unicode Text

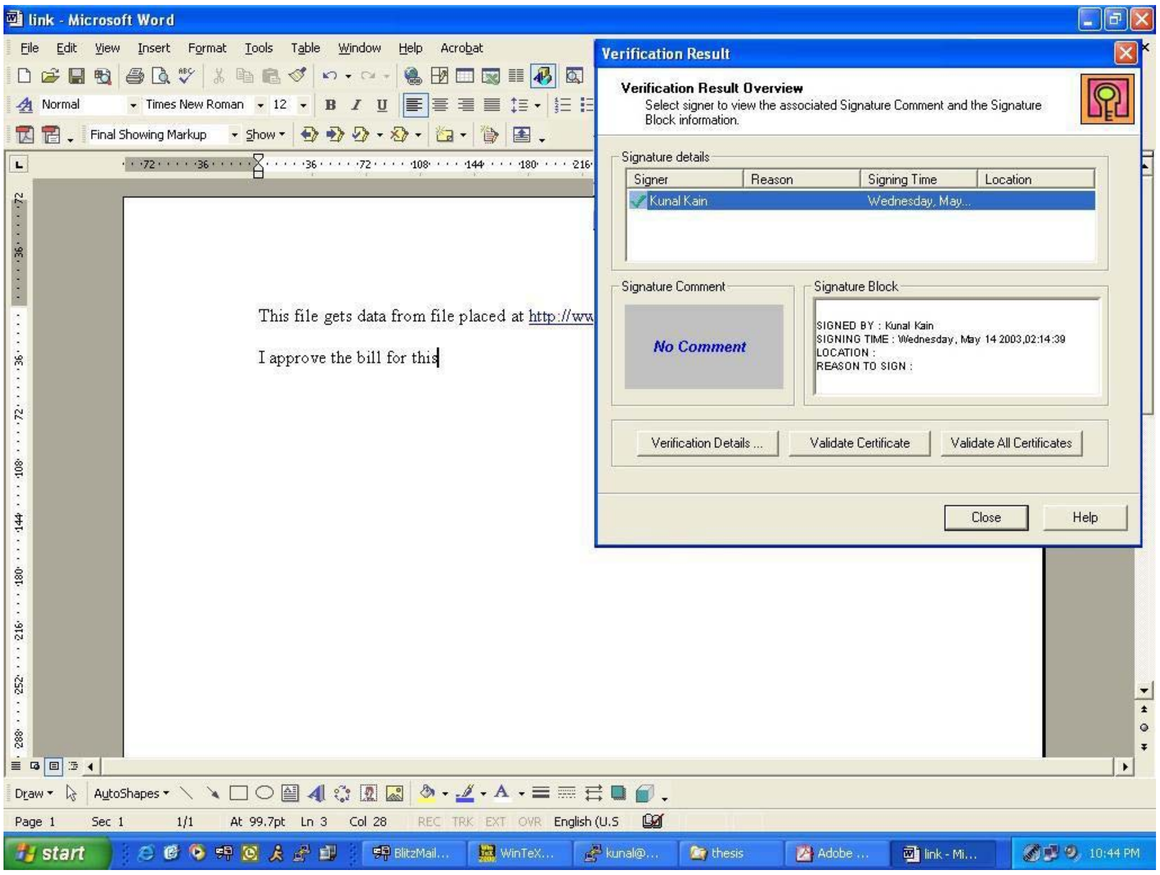
☐ Display as icon

Result

Inserts the contents of the Clipboard as text without any formatting.

Paste Link creates a shortcut to the source file. Changes to the source file will be reflected in your document.

Figure 6.11: Pasting link as unformatted text



This file gets data from file placed at http://ww

I approve the bill for this

Verification Result Overview
Select signer to view the associated Signature Comment and the Signature Block information.

Signature details

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Wednesday, May...	

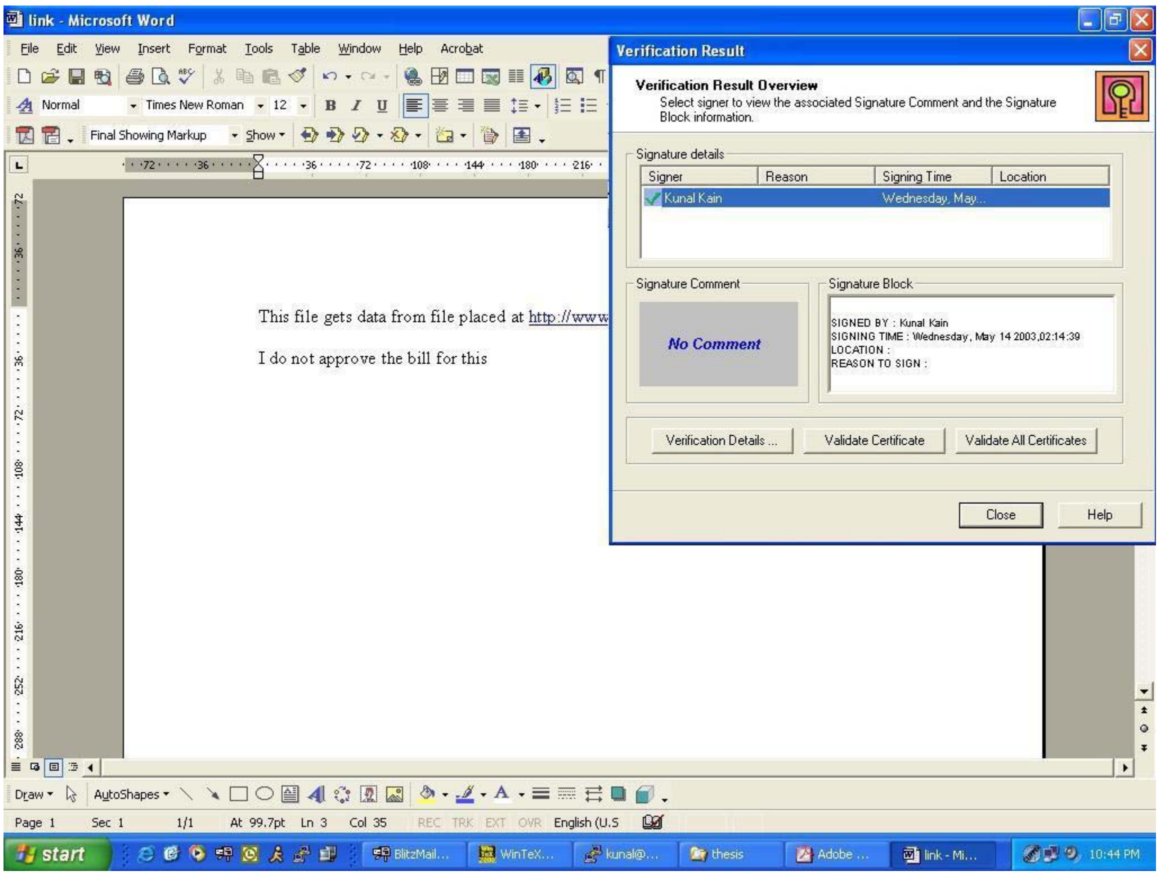
Signature Comment

No Comment

Signature Block

SIGNED BY : Kunal Kain
SIGNING TIME : Wednesday, May 14 2003,02:14:39
LOCATION :
REASON TO SIGN :

Figure 6.12: E-Lock *Prosigner* verified document viewed on *May 14, 2003*



This file gets data from file placed at http://www

I do not approve the bill for this

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Wednesday, May...	

No Comment

SIGNED BY : Kunal Kain
SIGNING TIME : Wednesday, May 14 2003,02:14:39
LOCATION :
REASON TO SIGN :

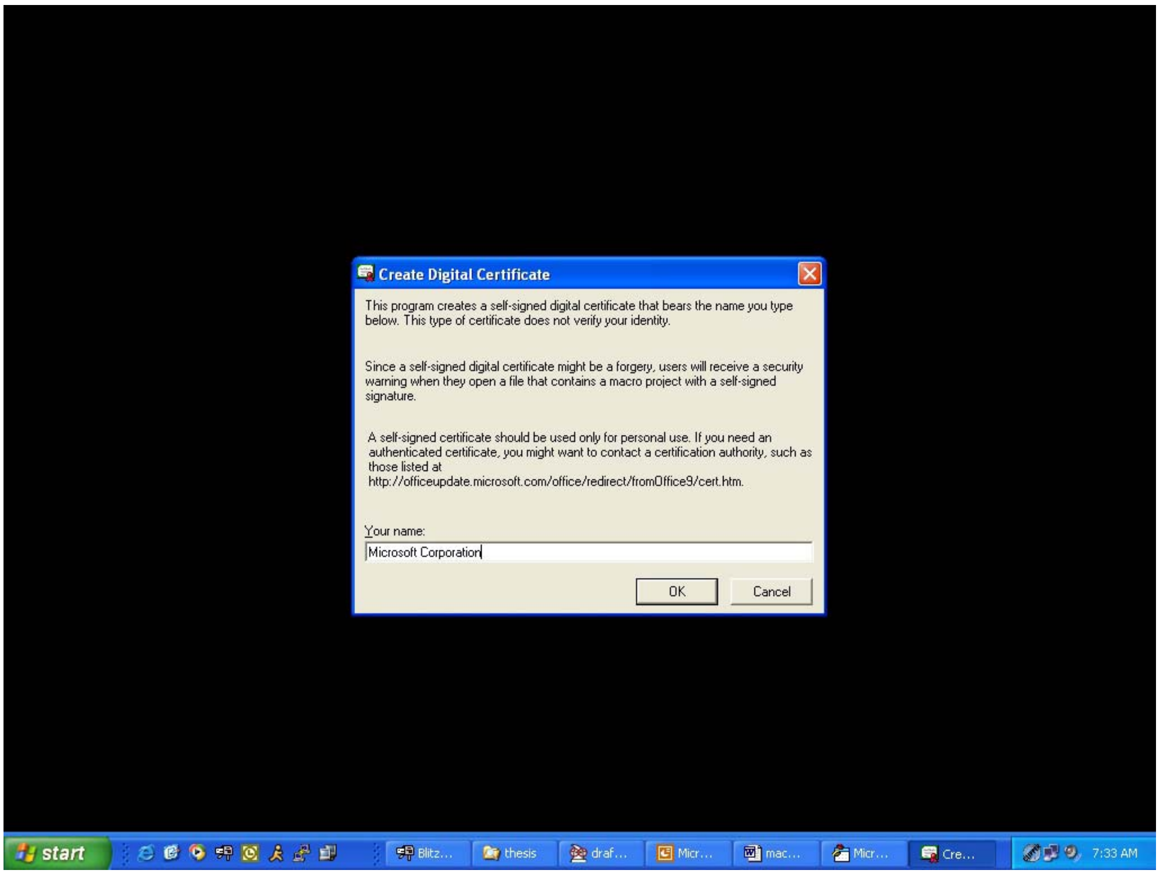
Figure 6.13: E-Lock *Prosigner* verified document viewed on *May 14, 2003* after “approve.doc” is modified

6.1.3 An attack on Microsoft Word using Signed Macros

Microsoft Word 2000/XP allow the signing of macros. This allows users to accept signed macros from some entity they consider trusted and run these macros without a warning window everytime. I will show an attack using this feature to trick users into accepting a macro from someone whom they consider trusted. I create a certificate using the *Selfcert.exe* application provided by Microsoft, and use this certificate to sign the macro.

Microsoft Office XP in allowing users to digitally sign Office documents (using there built-in signature application) blocks all auto-updates in the signed document. My attack will use a signed macro to update all the fields and link on the document to carry out this attack.

Figures 6.14 and 6.15 indicate how the *Selfcert.exe* application is used to create a certificate from *Microsoft Corporation*. Figure 6.16 shows a simple Office document. It is easy to insert macros in Word as shown by opening the Visual Basic Editor in Figure 6.17 and inserting the "update all fields" macro in the *Document_Open()* function call in Figure 6.18. In Figures 6.19, 6.20, 6.21 and 6.22, I sign the macro using the certificate generated using *Selfcert*. Figures 6.23 and 6.24 show how one can use the Microsoft Office's built-in PKI. Figures 6.25, 6.26, 6.27 show the steps in signing a *Microsoft Word* document. To show the depth of this attack, I set the macro security settings to hign in Figure 6.28. When the file is opened for the first time, a warning window pops up (displayed in Figure 6.29) which says it does not trust the certificate, but says it is from Microsoft. Figure 6.30 shows how most users are tricked into accepting the macro. Thus, when the document is opened on *Monday, May 26, 2003*, it shows one amount as shown in Figure 6.31, and when it is opened on *Tuesday, May 27, 2003*, it shows another amount as shown by Figure 6.32. The small mark at the bottom and the *signed* comment next to the document name indicates that the document is signed and the content has not changed according to Microsoft's signature application.


Create Digital Certificate

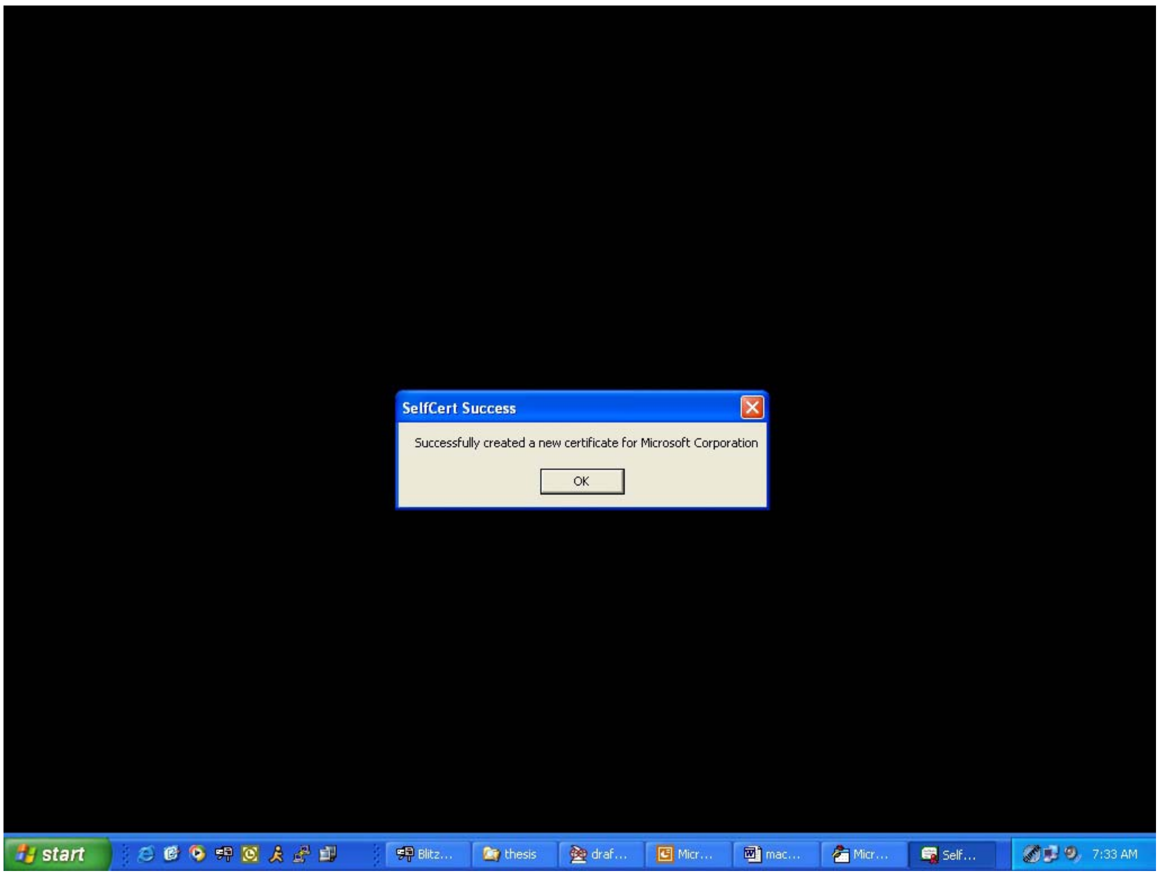
This program creates a self-signed digital certificate that bears the name you type below. This type of certificate does not verify your identity.

Since a self-signed digital certificate might be a forgery, users will receive a security warning when they open a file that contains a macro project with a self-signed signature.

A self-signed certificate should be used only for personal use. If you need an authenticated certificate, you might want to contact a certification authority, such as those listed at http://officeupdate.microsoft.com/office/redirect/fromOffice9/cert.htm.

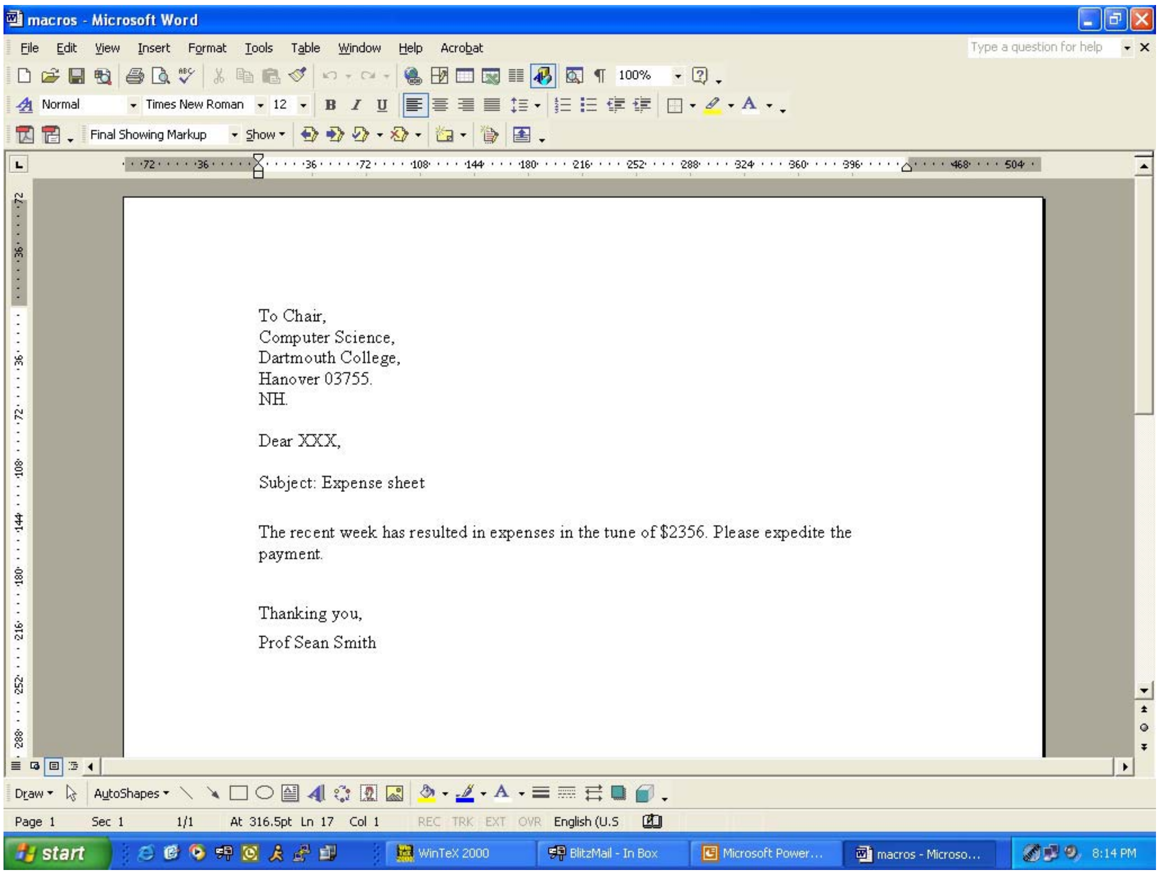
Your name:
Microsoft Corporation

Figure 6.14: The *Selfcert.exe* application


SelfCert Success

Successfully created a new certificate for Microsoft Corporation

Figure 6.15: Using *Selfcert.exe* to create a certificate from “Microsoft Corporation”



To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

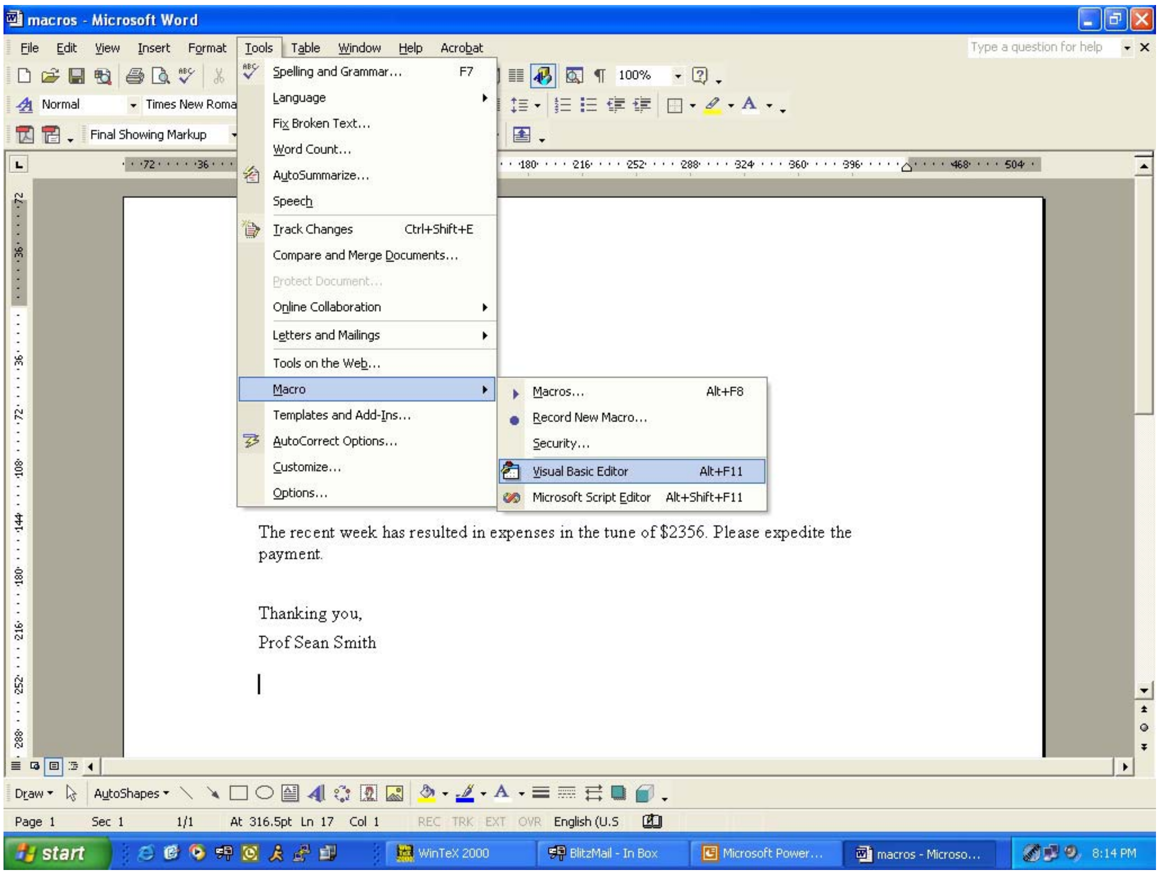
Subject: Expense sheet

The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.16: A simple Office document containing a field

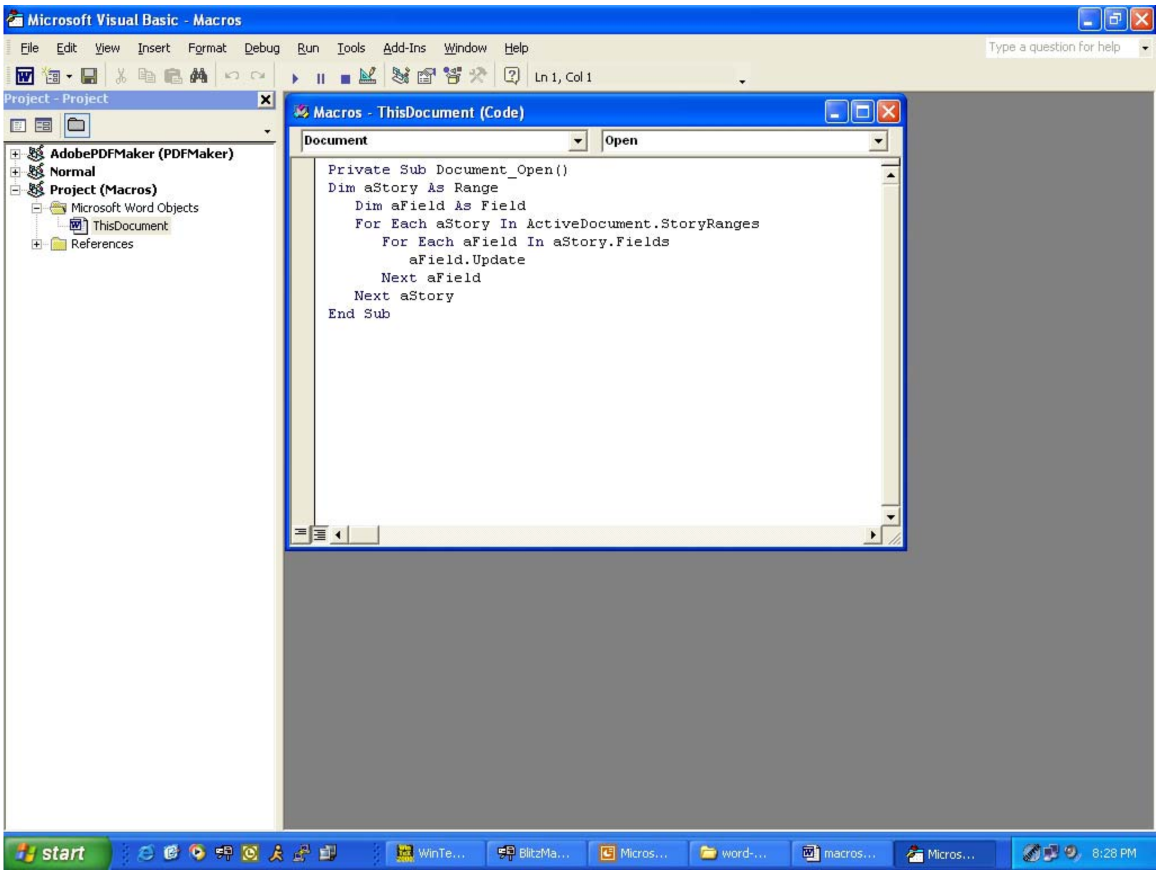


The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

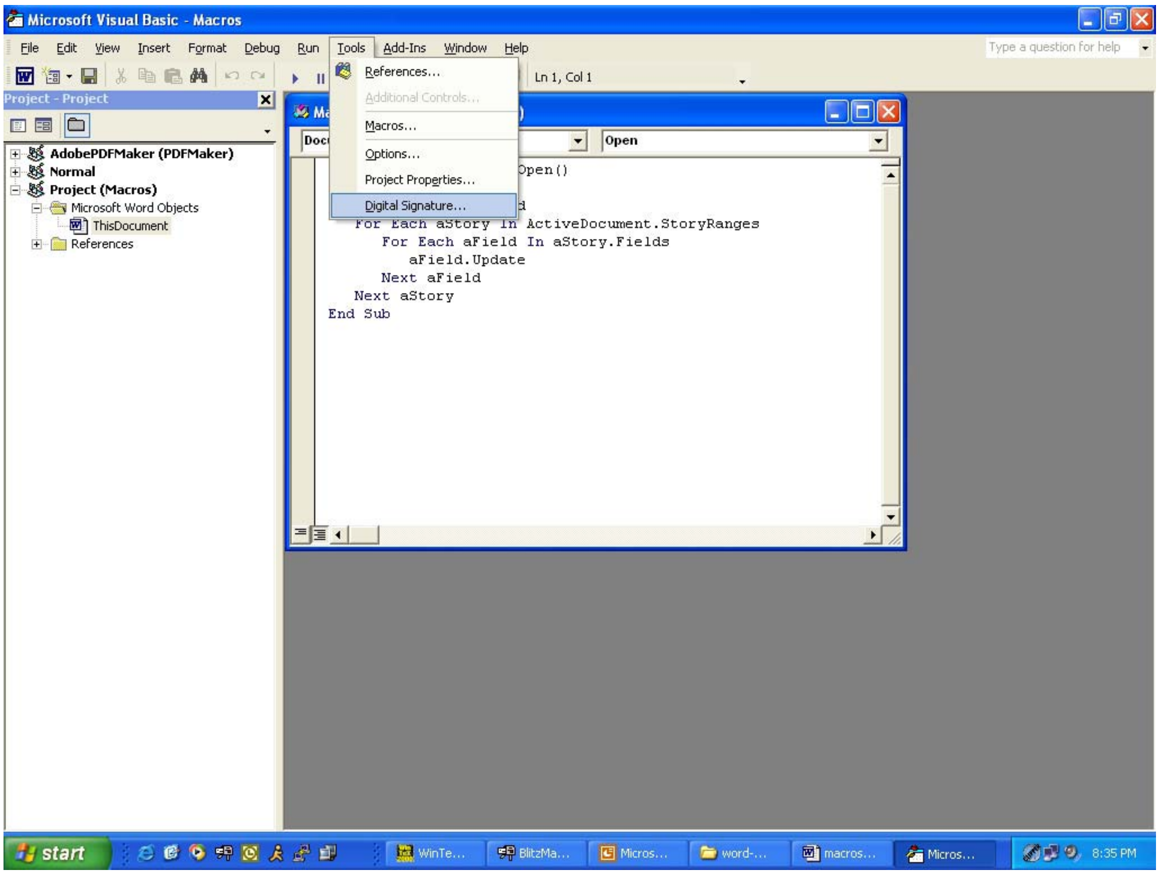
Prof Sean Smith

Figure 6.17: Opening the Visual Basic Editor from the “Tools” toolbar



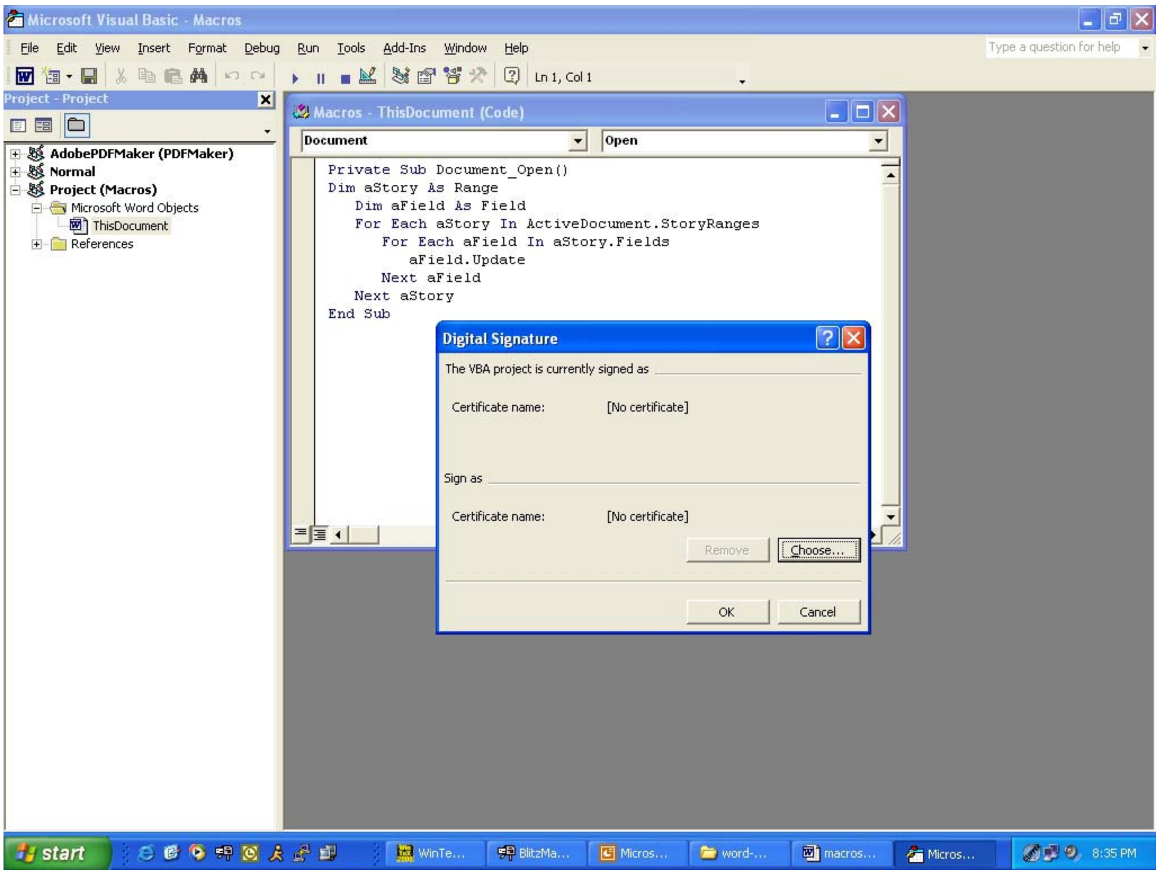
```
Private Sub Document_Open()
Dim aStory As Range
   Dim aField As Field
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

Figure 6.18: Inserting the “update all fields” macro in the “Document_Open()” function



```
                    Open()

                    d
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

Figure 6.19: Digitally signing the macro



```
Private Sub Document_Open()
Dim aStory As Range
   Dim aField As Field
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

Digital Signature

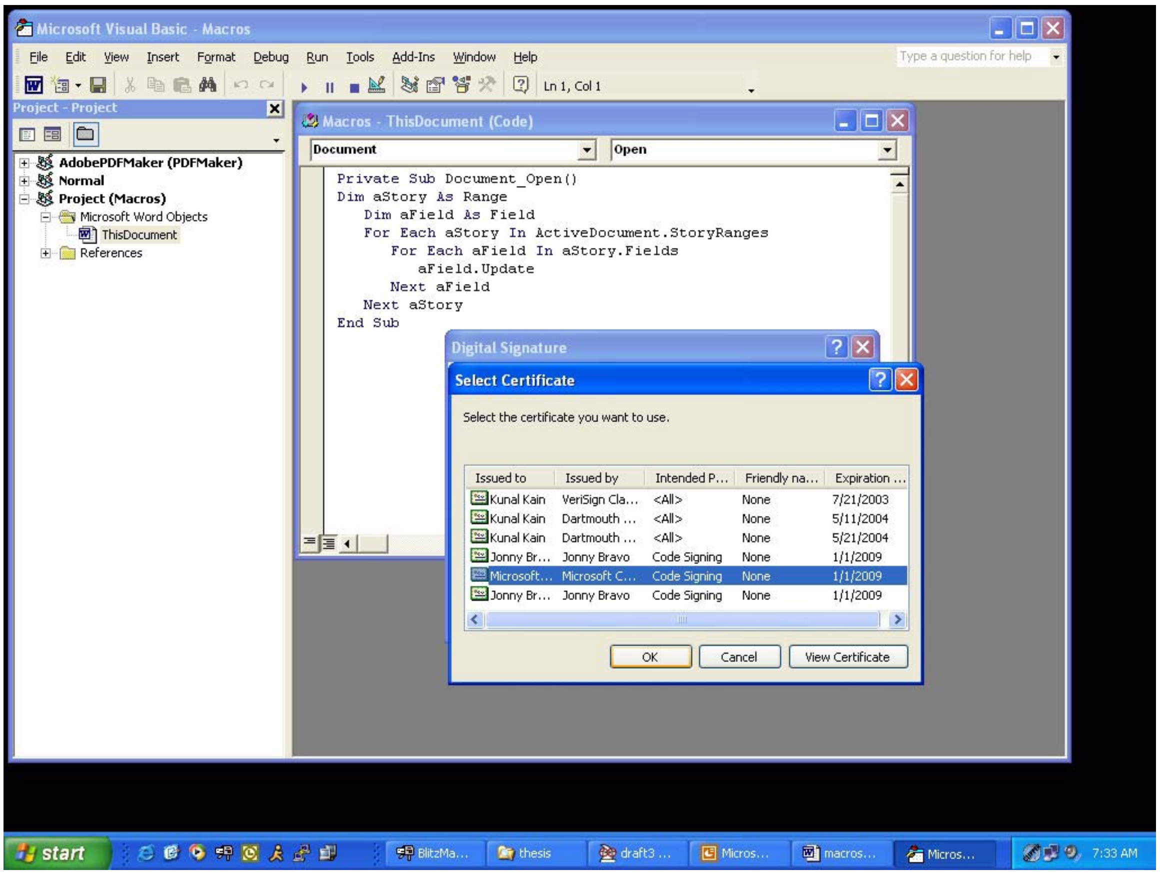
The VBA project is currently signed as

Certificate name: [No certificate]

Sign as

Certificate name: [No certificate]

Figure 6.20: Selecting certificate to sign macro

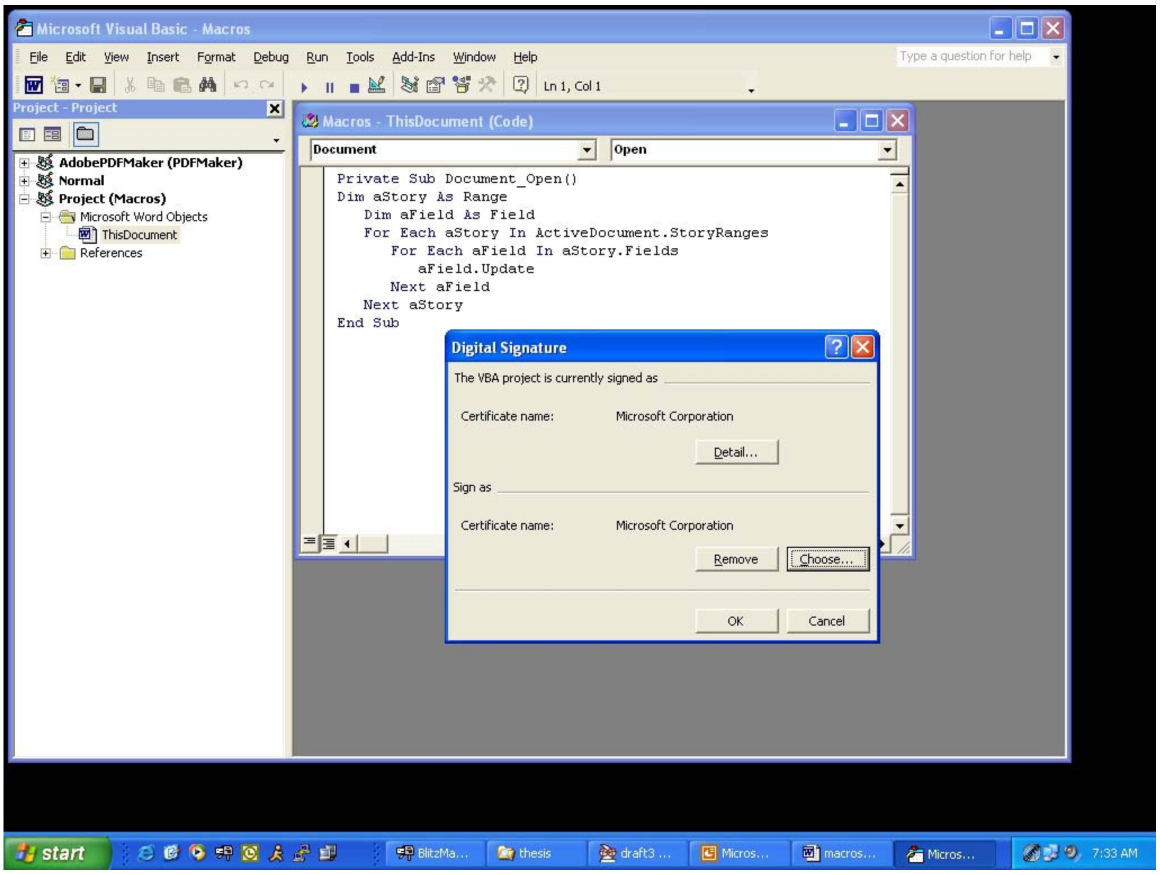


```
Private Sub Document_Open()
Dim aStory As Range
   Dim aField As Field
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

Select the certificate you want to use.

Issued to	Issued by	Intended P...	Friendly na...	Expiration ...
Kunal Kain	VeriSign Cla...	<All>	None	7/21/2003
Kunal Kain	Dartmouth ...	<All>	None	5/11/2004
Kunal Kain	Dartmouth ...	<All>	None	5/21/2004
Jonny Br...	Jonny Bravo	Code Signing	None	1/1/2009
Microsoft...	Microsoft C...	Code Signing	None	1/1/2009
Jonny Br...	Jonny Bravo	Code Signing	None	1/1/2009

Figure 6.21: Selecting the “Microsoft Corporation” certificate created by *Selfcert.exe*



```
Private Sub Document_Open()
Dim aStory As Range
   Dim aField As Field
   For Each aStory In ActiveDocument.StoryRanges
      For Each aField In aStory.Fields
         aField.Update
      Next aField
   Next aStory
End Sub
```

Digital Signature

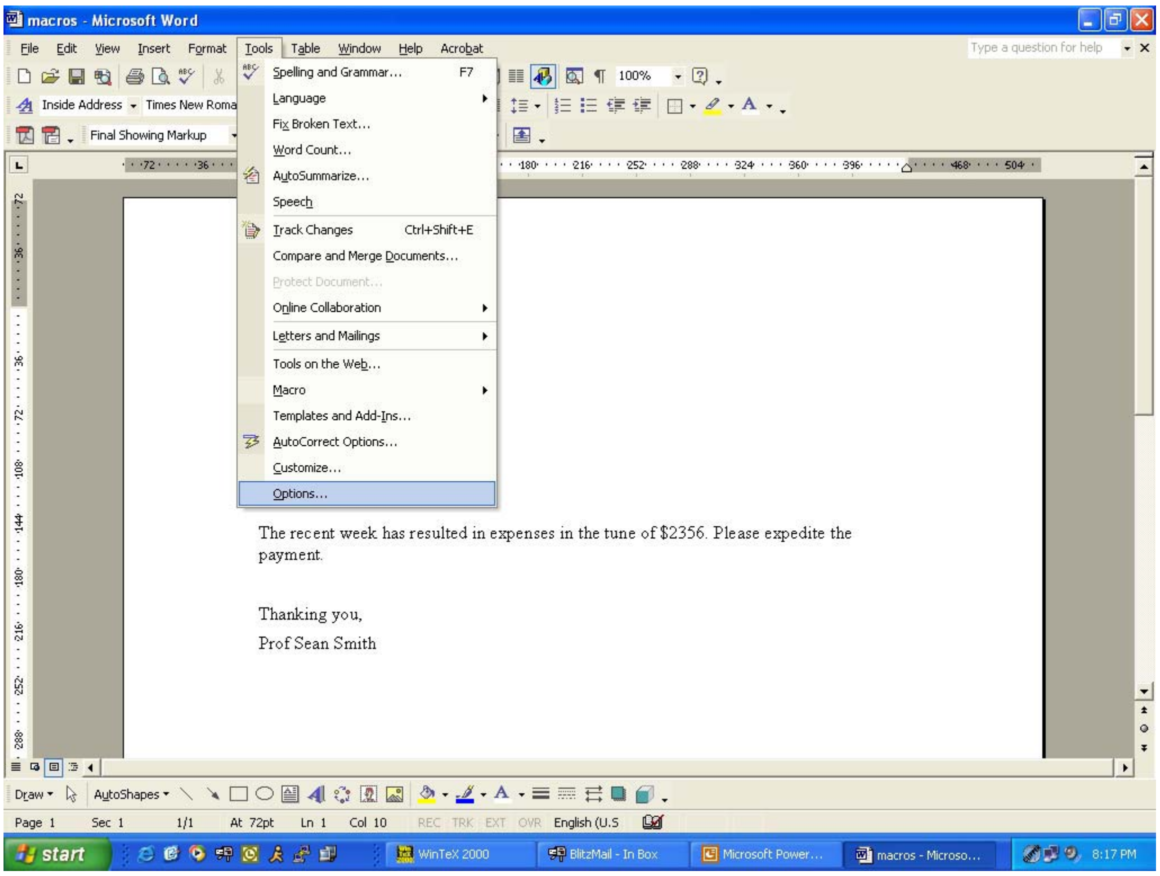
The VBA project is currently signed as

Certificate name: Microsoft Corporation

Sign as

Certificate name: Microsoft Corporation

Figure 6.22: Signing the macro with the chosen certificate

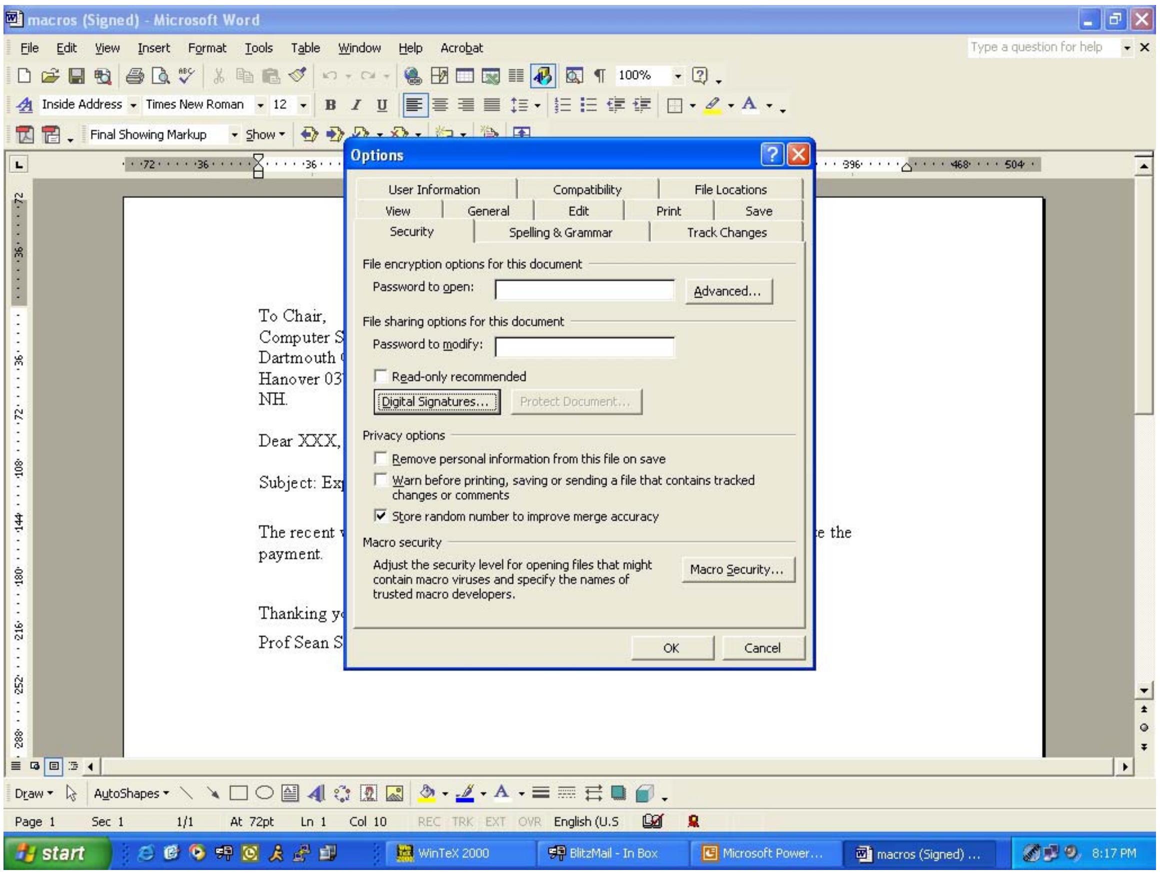


The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.23: Selecting the options menu item in the "Tools" toolbar



File encryption options for this document

Password to open:

File sharing options for this document

Password to modify:

☐ Read-only recommended

Privacy options

☐ Remove personal information from this file on save

☐ Warn before printing, saving or sending a file that contains tracked changes or comments

☑ Store random number to improve merge accuracy

Macro security

Adjust the security level for opening files that might contain macro viruses and specify the names of trusted macro developers.

To Chair,
Computer S
Dartmouth
Hanover 03
NH.

Dear XXX,

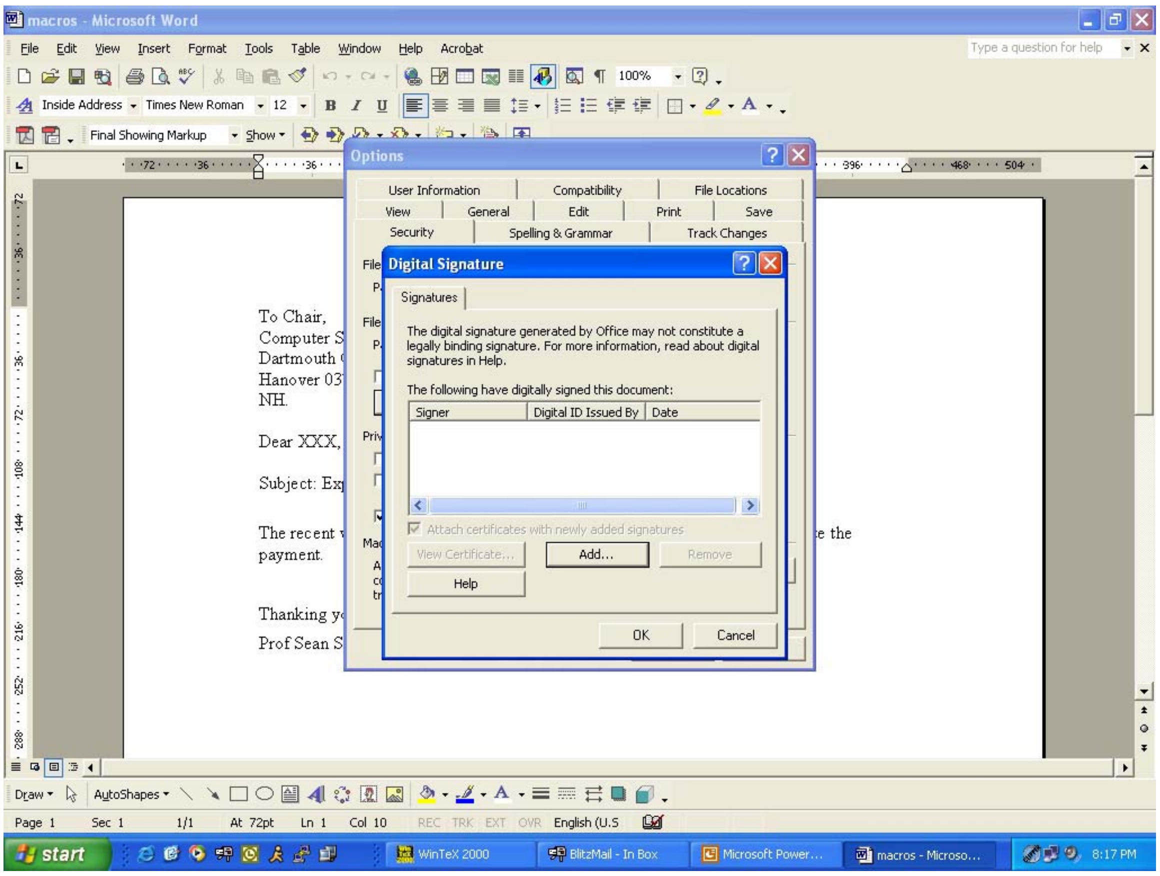
Subject: Ex

The recent … e the payment.

Thanking y

Prof Sean S

Figure 6.24: Digitally signing the document using built-in PKI


macros - Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Type a question for help
Inside Address Times New Roman 12
Final Showing Markup Show
Options
User Information Compatibility File Locations
View General Edit Print Save
Security Spelling & Grammar Track Changes
Digital Signature
Signatures
Signer Digital ID Issued By Date
Attach certificates with newly added signatures
View Certificate... Add... Remove
Help
OK Cancel
Draw AutoShapes
Page 1 Sec 1 1/1 At 72pt Ln 1 Col 10 REC TRK EXT OVR English (U.S
start WinTeX 2000 BlitzMail - In Box Microsoft Power... macros - Microso... 8:17 PM

The digital signature generated by Office may not constitute a legally binding signature. For more information, read about digital signatures in Help.

The following have digitally signed this document:

To Chair,
Computer S
Dartmouth C
Hanover 03
NH.

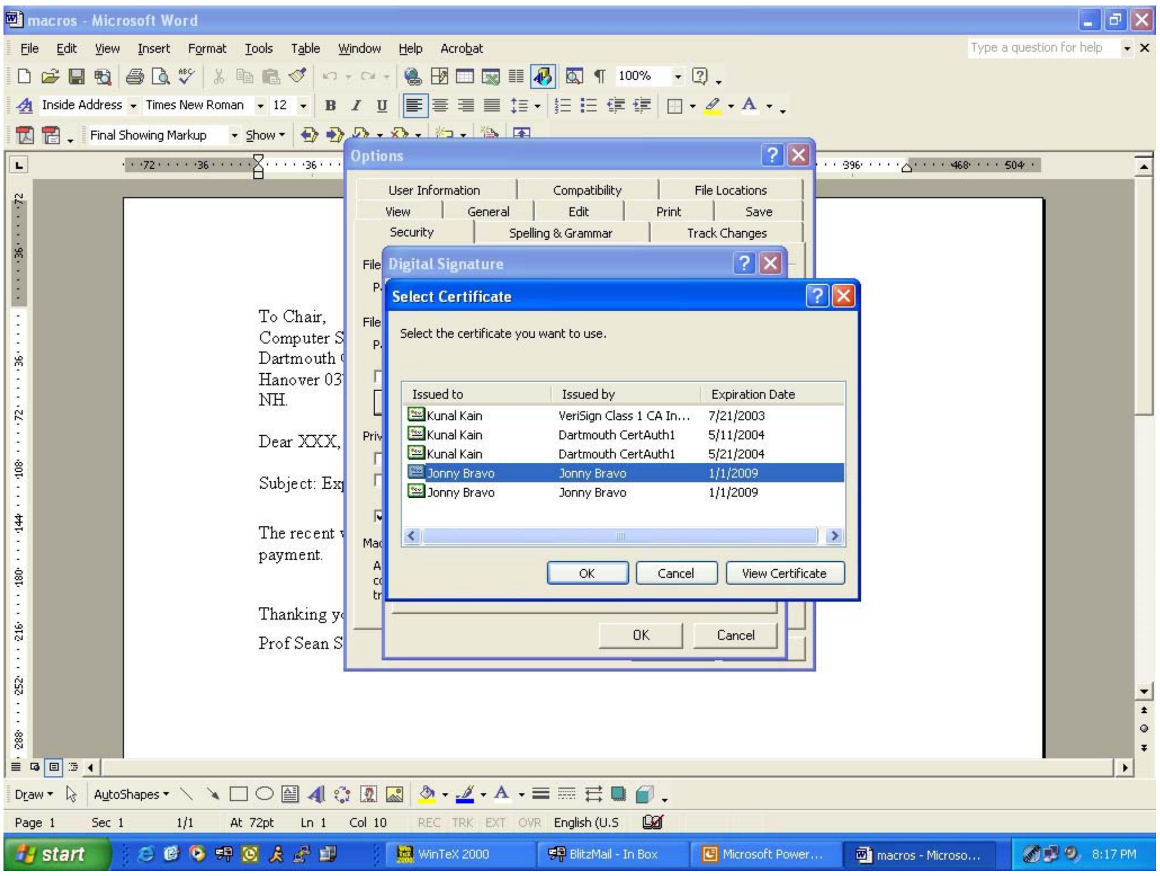
Dear XXX,

Subject: Ex

The recent v ... e the payment.

Thanking y
Prof Sean S

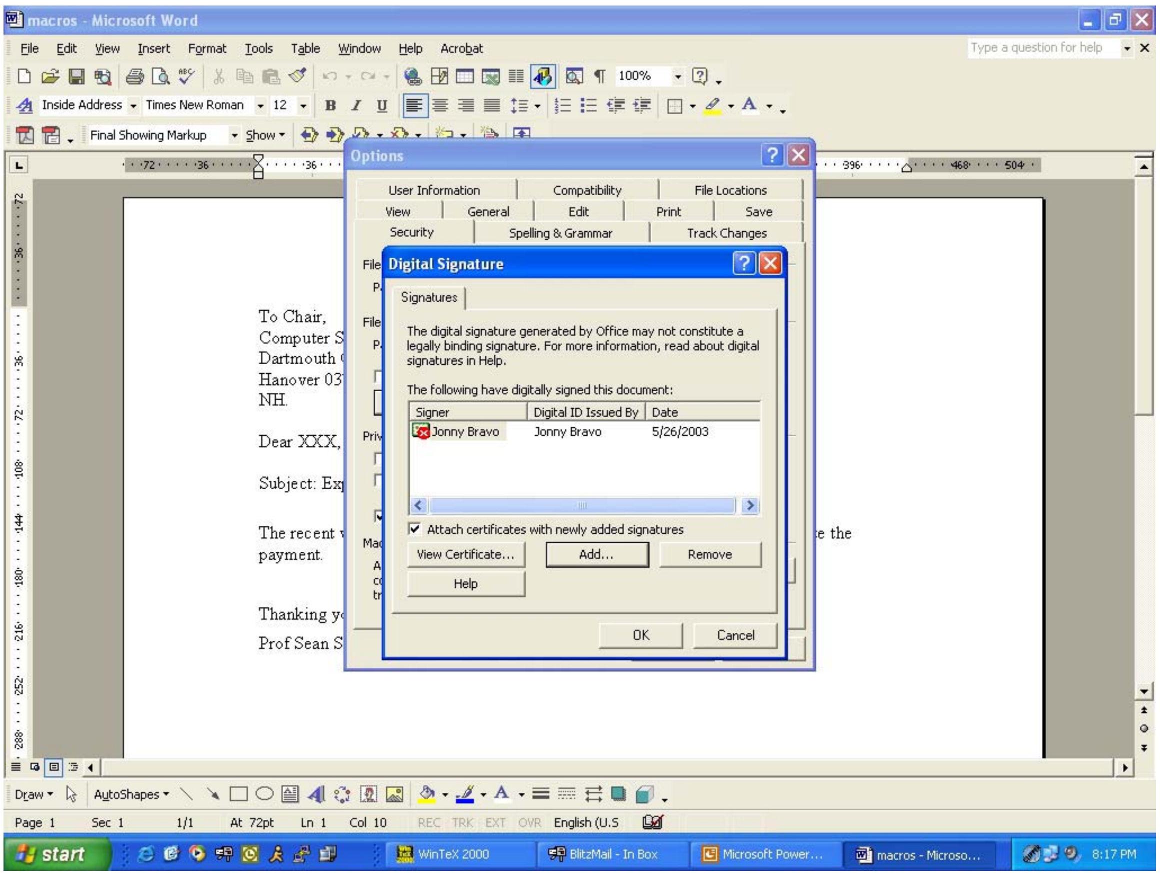
Figure 6.25: Adding a signer step 1



Select the certificate you want to use.

Issued to	Issued by	Expiration Date
Kunal Kain	VeriSign Class 1 CA In...	7/21/2003
Kunal Kain	Dartmouth CertAuth1	5/11/2004
Kunal Kain	Dartmouth CertAuth1	5/21/2004
Jonny Bravo	Jonny Bravo	1/1/2009
Jonny Bravo	Jonny Bravo	1/1/2009

Figure 6.26: Adding a signer step 2



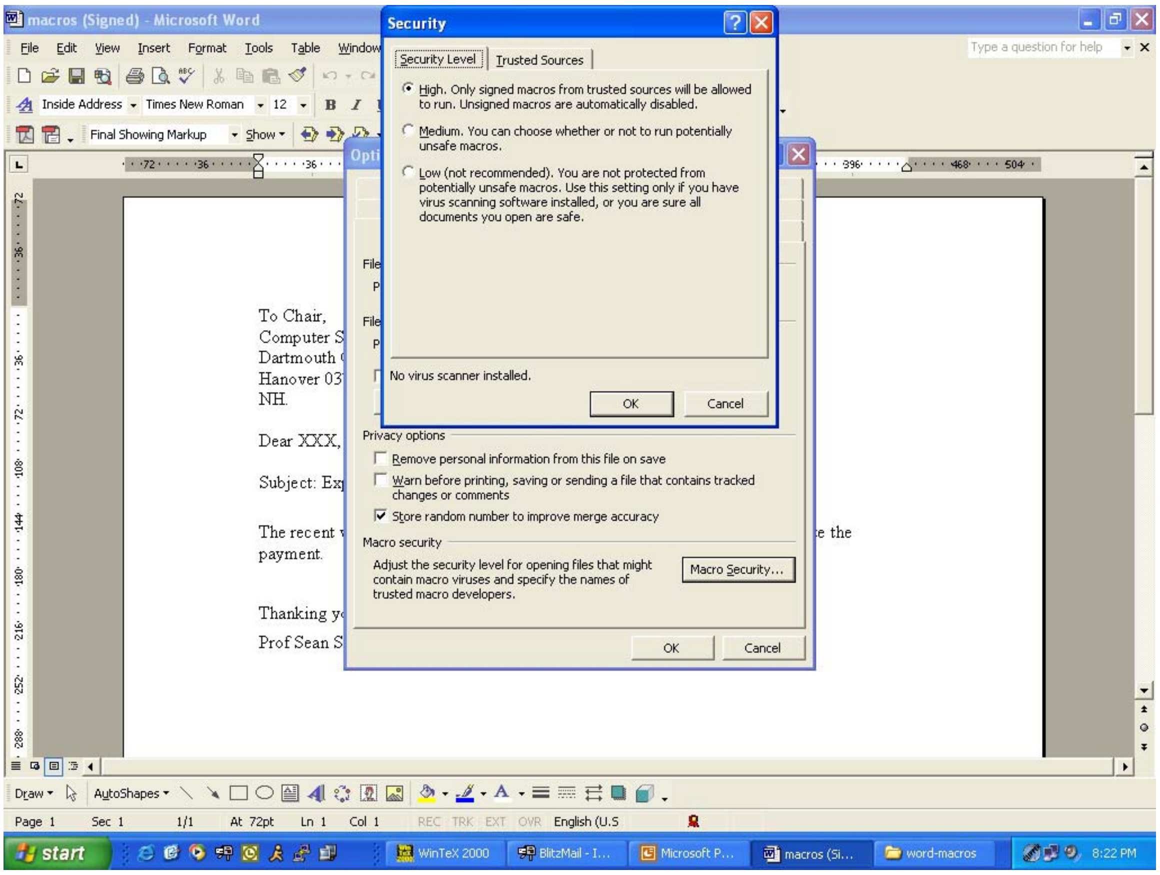
The digital signature generated by Office may not constitute a legally binding signature. For more information, read about digital signatures in Help.

The following have digitally signed this document:

Signer	Digital ID Issued By	Date
Jonny Bravo	Jonny Bravo	5/26/2003

☑ Attach certificates with newly added signatures

Figure 6.27: Adding a signer step 3



Security Level | Trusted Sources

(●) High. Only signed macros from trusted sources will be allowed to run. Unsigned macros are automatically disabled.

( ) Medium. You can choose whether or not to run potentially unsafe macros.

( ) Low (not recommended). You are not protected from potentially unsafe macros. Use this setting only if you have virus scanning software installed, or you are sure all documents you open are safe.

No virus scanner installed.

Privacy options

☐ Remove personal information from this file on save

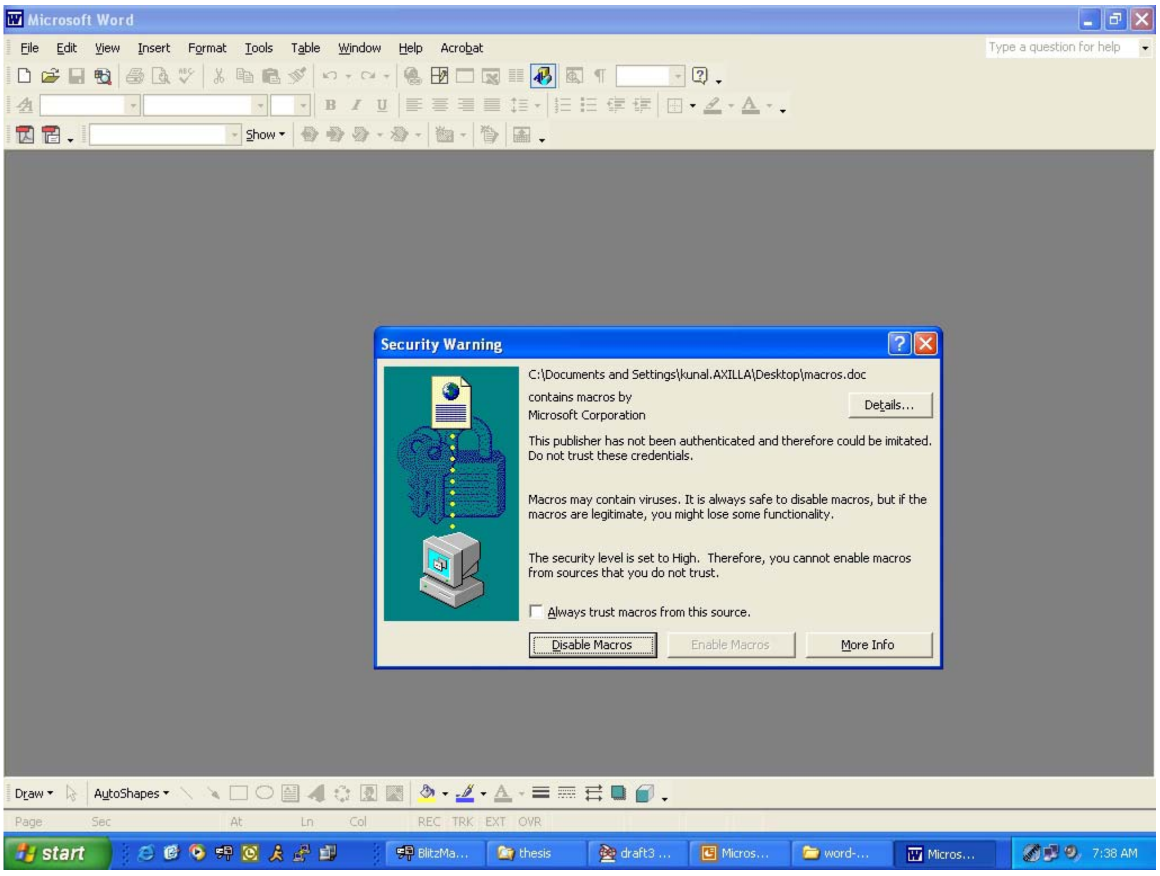
☐ Warn before printing, saving or sending a file that contains tracked changes or comments

☑ Store random number to improve merge accuracy

Macro security

Adjust the security level for opening files that might contain macro viruses and specify the names of trusted macro developers.

Figure 6.28: Setting macro security to High


Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Type a question for help
Show
Security Warning
Details...
Disable Macros
Enable Macros
More Info
Draw AutoShapes
Page Sec At Ln Col REC TRK EXT OVR
start
BlitzMa... thesis draft3 ... Micros... word-... Micros...
7:38 AM

C:\Documents and Settings\kunal.AXILLA\Desktop\macros.doc
contains macros by
Microsoft Corporation

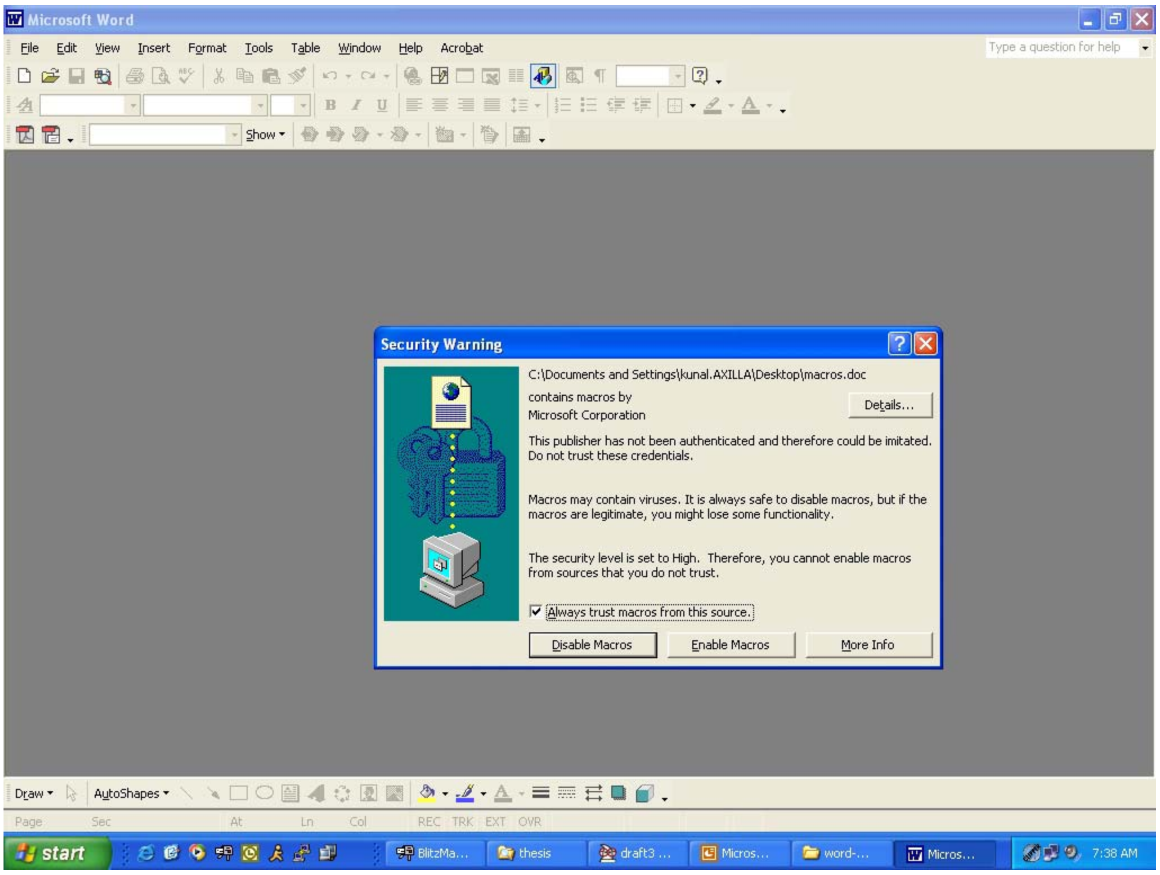
This publisher has not been authenticated and therefore could be imitated.
Do not trust these credentials.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

The security level is set to High. Therefore, you cannot enable macros from sources that you do not trust.

☐ Always trust macros from this source.

Figure 6.29: Warning window shown when the file is opened says it does not trust the certificate but says its from Microsoft


Security Warning

C:\Documents and Settings\kunal.AXILLA\Desktop\macros.doc

contains macros by

Microsoft Corporation

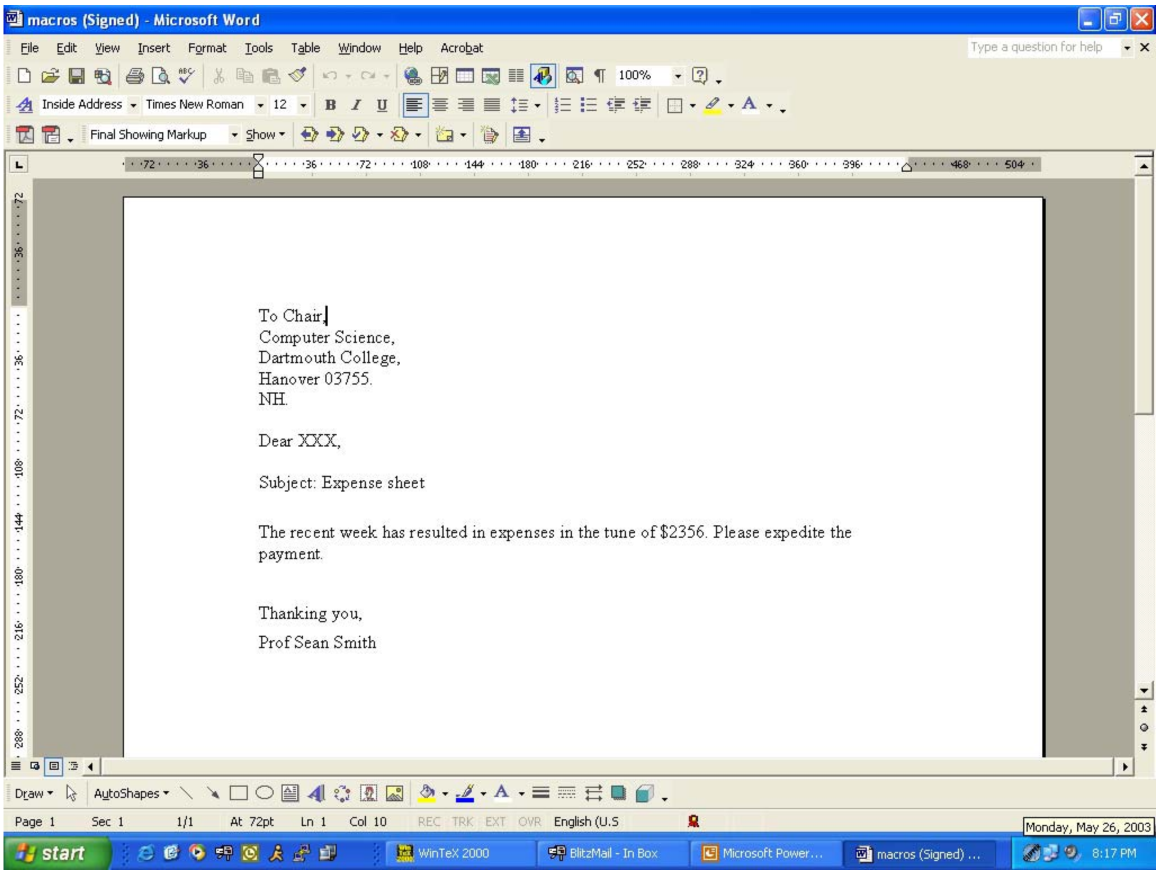
This publisher has not been authenticated and therefore could be imitated. Do not trust these credentials.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

The security level is set to High. Therefore, you cannot enable macros from sources that you do not trust.

☑ Always trust macros from this source.

Figure 6.30: Tricking the user into accepting the macro


macros (Signed) - Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Inside Address
Times New Roman
12
Final Showing Markup
Show
Page 1 Sec 1 1/1 At 72pt Ln 1 Col 10
English (U.S
Monday, May 26, 2003
8:17 PM

To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

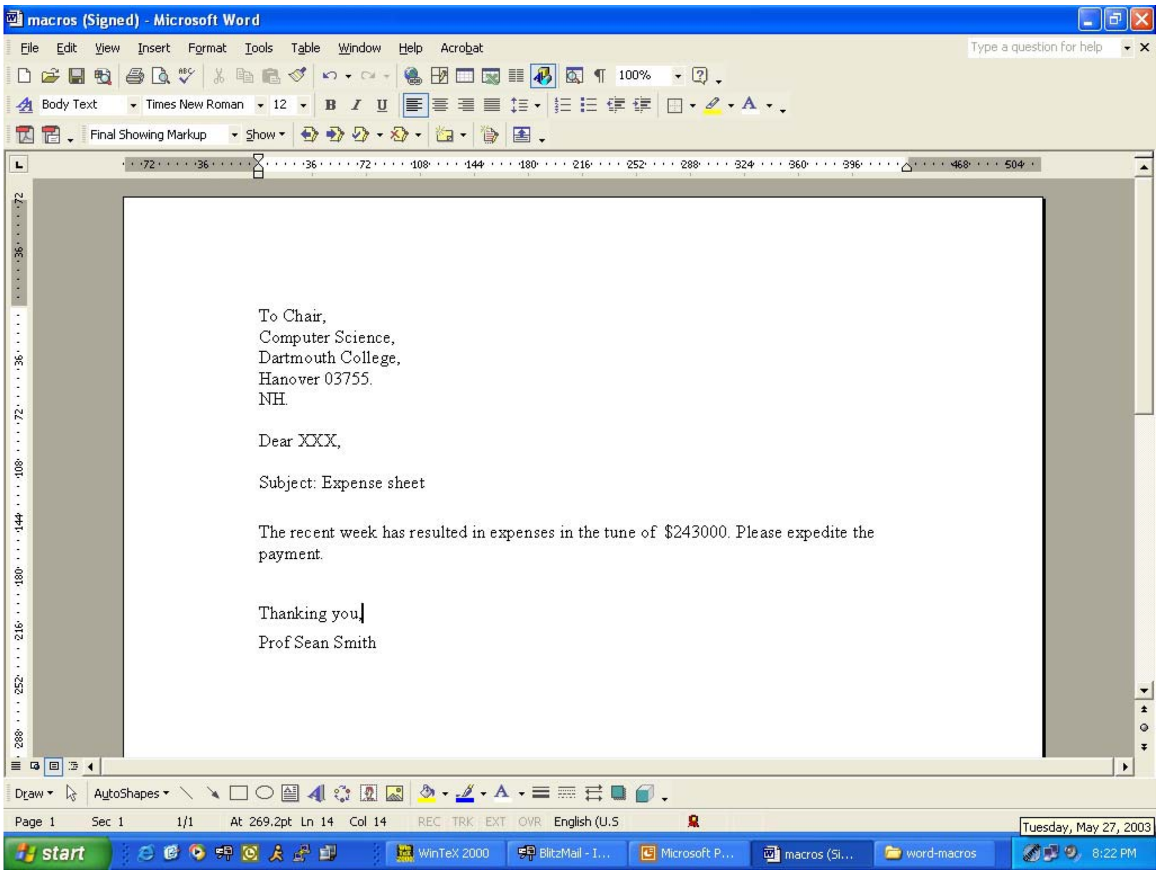
Subject: Expense sheet

The recent week has resulted in expenses in the tune of $2356. Please expedite the payment.

Thanking you,

Prof Sean Smith

Figure 6.31: Document viewed on *Monday, May 26, 2003*


macros (Signed) - Microsoft Word
File Edit View Insert Format Tools Table Window Help Acrobat
Body Text Times New Roman 12
Final Showing Markup Show
Page 1 Sec 1 1/1 At 269.2pt Ln 14 Col 14 REC TRK EXT OVR English (U.S
Tuesday, May 27, 2003
start WinTeX 2000 BlitzMail - I... Microsoft P... macros (Si... word-macros 8:22 PM

To Chair,
Computer Science,
Dartmouth College,
Hanover 03755.
NH.

Dear XXX,

Subject: Expense sheet

The recent week has resulted in expenses in the tune of $243000. Please expedite the payment.

Thanking you,

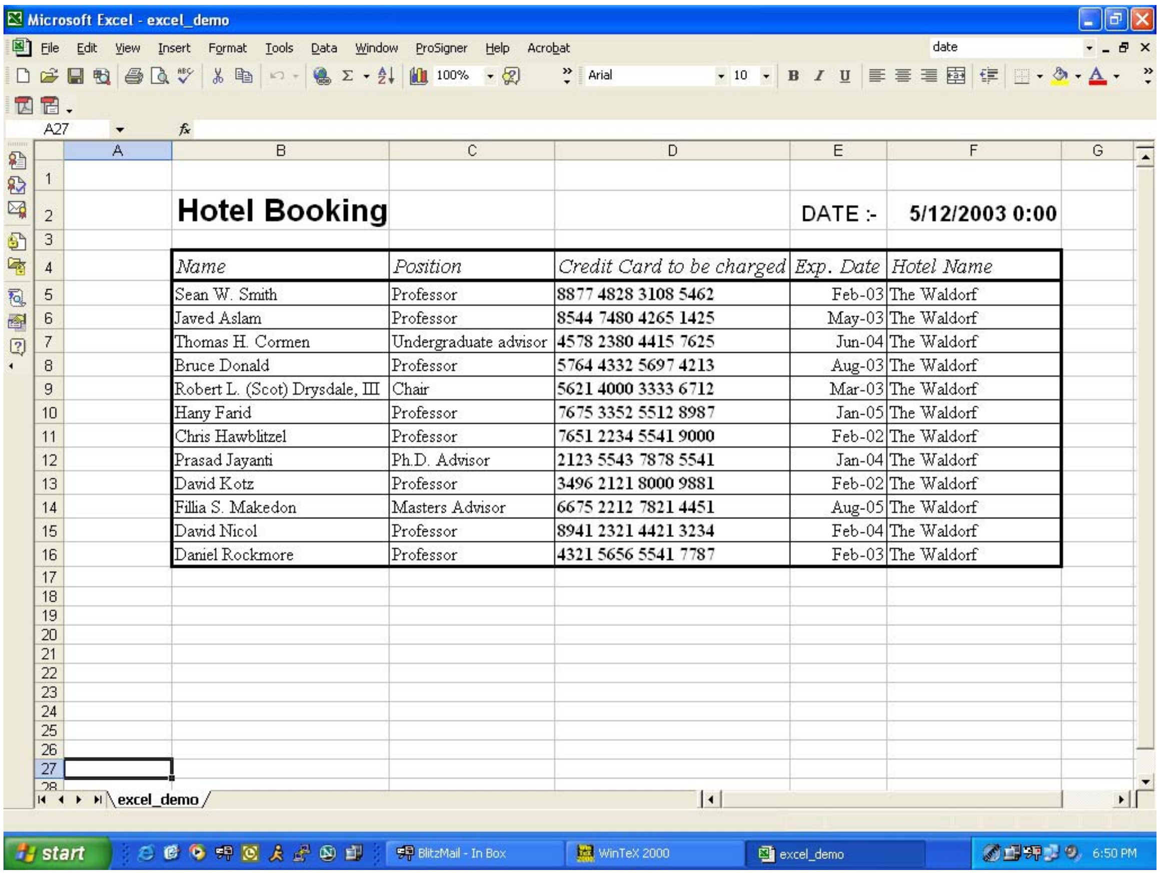
Prof Sean Smith

Figure 6.32: Document viewed on *Tuesday, May 27, 2003* the mark at the bottom signifies that the document content has not changed

6.2 An attack on Microsoft Excel

Figure 6.33 shows a simple Excel document sent by the department and signed by the people to pay for an overnight stay for a conference. It is easy for the attacker to conduct attacks using the *Now()* function to carry out an attack as demonstrated in Section 4.2.2. In the attacks due to some devious code Prof. Sean Smith ends up paying for everyone's stay at the hotel.

In Figure 6.34, dynamic content is inserted in each of the credit card number cells using the NOW() function call. Figures 6.35, 6.36 and 6.37 show the steps in the digital signatures process using E-Lock *Prosigner*. When verified on *May 12, 2003*, the document shows (Figure 6.38) the various credit card numbers and expiration dates and when verified on *May 13, 2003*, the document shows Prof. Sean Smith paying for everyone (Figure 6.39).

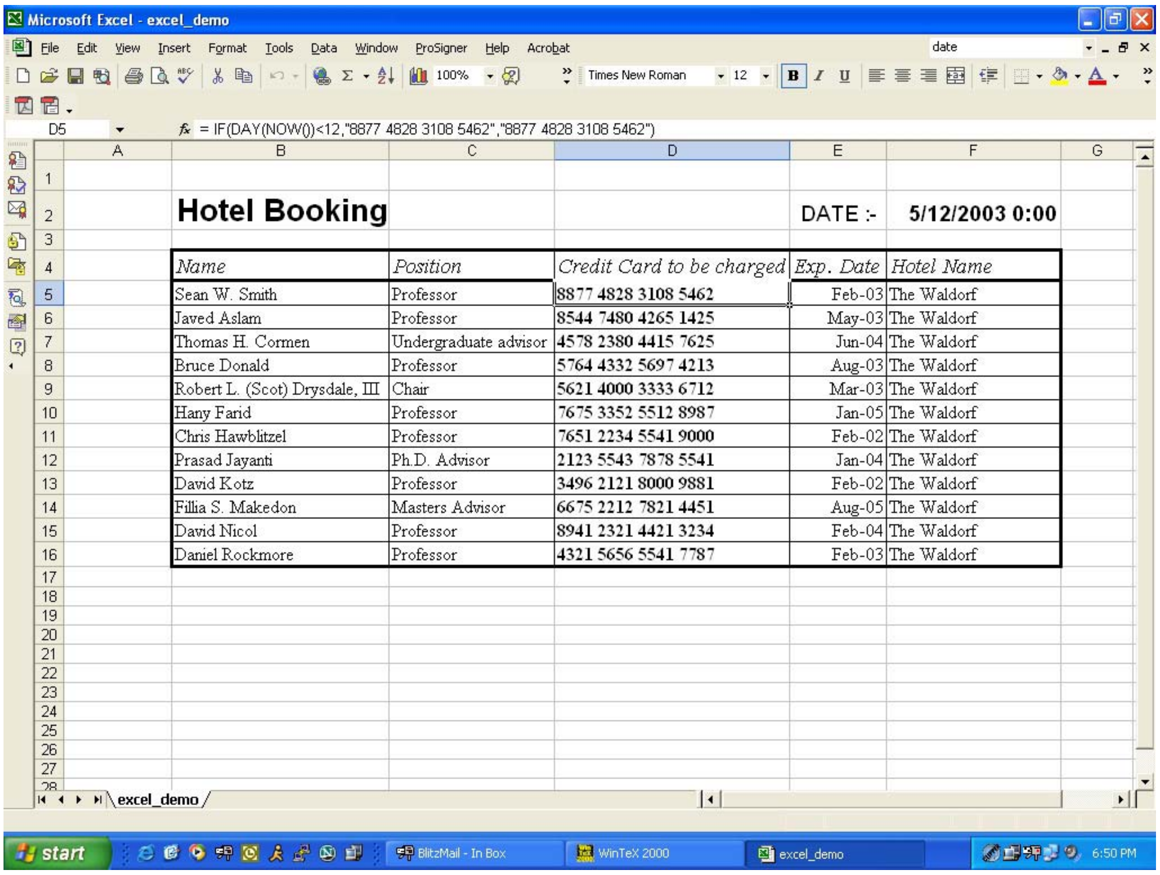


Microsoft Excel - excel_demo
File Edit View Insert Format Tools Data Window ProSigner Help Acrobat
date
100%
Arial
10
A27
excel_demo
start
BlitzMail - In Box
WinTeX 2000
excel_demo
6:50 PM

Hotel Booking DATE :- **5/12/2003 0:00**

Name	*Position*	*Credit Card to be charged*	*Exp. Date*	*Hotel Name*
Sean W. Smith	Professor	8877 4828 3108 5462	Feb-03	The Waldorf
Javed Aslam	Professor	8544 7480 4265 1425	May-03	The Waldorf
Thomas H. Cormen	Undergraduate advisor	4578 2380 4415 7625	Jun-04	The Waldorf
Bruce Donald	Professor	5764 4332 5697 4213	Aug-03	The Waldorf
Robert L. (Scot) Drysdale, III	Chair	5621 4000 3333 6712	Mar-03	The Waldorf
Hany Farid	Professor	7675 3352 5512 8987	Jan-05	The Waldorf
Chris Hawblitzel	Professor	7651 2234 5541 9000	Feb-02	The Waldorf
Prasad Jayanti	Ph.D. Advisor	2123 5543 7878 5541	Jan-04	The Waldorf
David Kotz	Professor	3496 2121 8000 9881	Feb-02	The Waldorf
Fillia S. Makedon	Masters Advisor	6675 2212 7821 4451	Aug-05	The Waldorf
David Nicol	Professor	8941 2321 4421 3234	Feb-04	The Waldorf
Daniel Rockmore	Professor	4321 5656 5541 7787	Feb-03	The Waldorf

Figure 6.33: A simple Microsoft Excel document

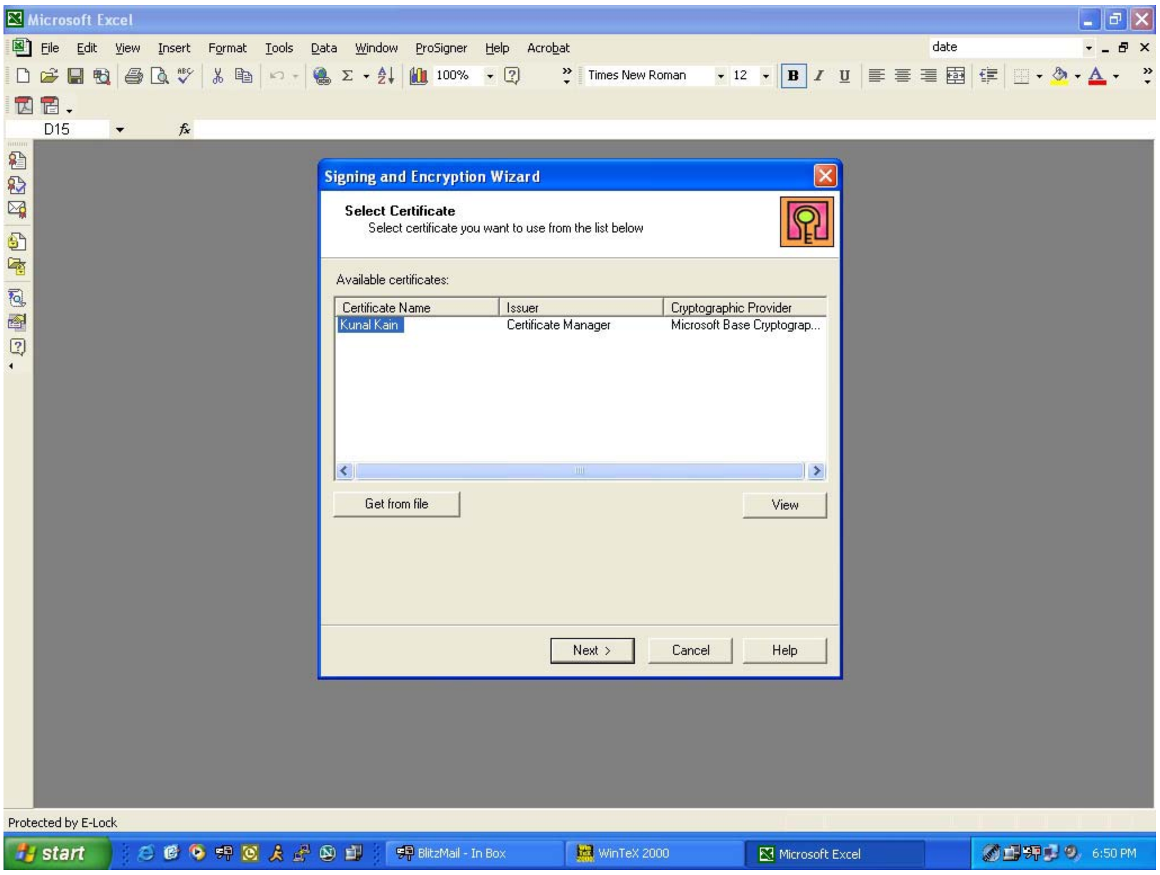

Microsoft Excel - excel_demo
File Edit View Insert Format Tools Data Window ProSigner Help Acrobat
Times New Roman 12
excel_demo

D5 = IF(DAY(NOW())<12,"8877 4828 3108 5462","8877 4828 3108 5462")

Hotel Booking DATE :- **5/12/2003 0:00**

Name	*Position*	*Credit Card to be charged*	*Exp. Date*	*Hotel Name*
Sean W. Smith	Professor	8877 4828 3108 5462	Feb-03	The Waldorf
Javed Aslam	Professor	8544 7480 4265 1425	May-03	The Waldorf
Thomas H. Cormen	Undergraduate advisor	4578 2380 4415 7625	Jun-04	The Waldorf
Bruce Donald	Professor	5764 4332 5697 4213	Aug-03	The Waldorf
Robert L. (Scot) Drysdale, III	Chair	5621 4000 3333 6712	Mar-03	The Waldorf
Hany Farid	Professor	7675 3352 5512 8987	Jan-05	The Waldorf
Chris Hawblitzel	Professor	7651 2234 5541 9000	Feb-02	The Waldorf
Prasad Jayanti	Ph.D. Advisor	2123 5543 7878 5541	Jan-04	The Waldorf
David Kotz	Professor	3496 2121 8000 9881	Feb-02	The Waldorf
Fillia S. Makedon	Masters Advisor	6675 2212 7821 4451	Aug-05	The Waldorf
David Nicol	Professor	8941 2321 4421 3234	Feb-04	The Waldorf
Daniel Rockmore	Professor	4321 5656 5541 7787	Feb-03	The Waldorf

Figure 6.34: Inserting dynamic content using the NOW() function call


Microsoft Excel
File Edit View Insert Format Tools Data Window ProSigner Help Acrobat
date
100%
Times New Roman
12
D15
Signing and Encryption Wizard

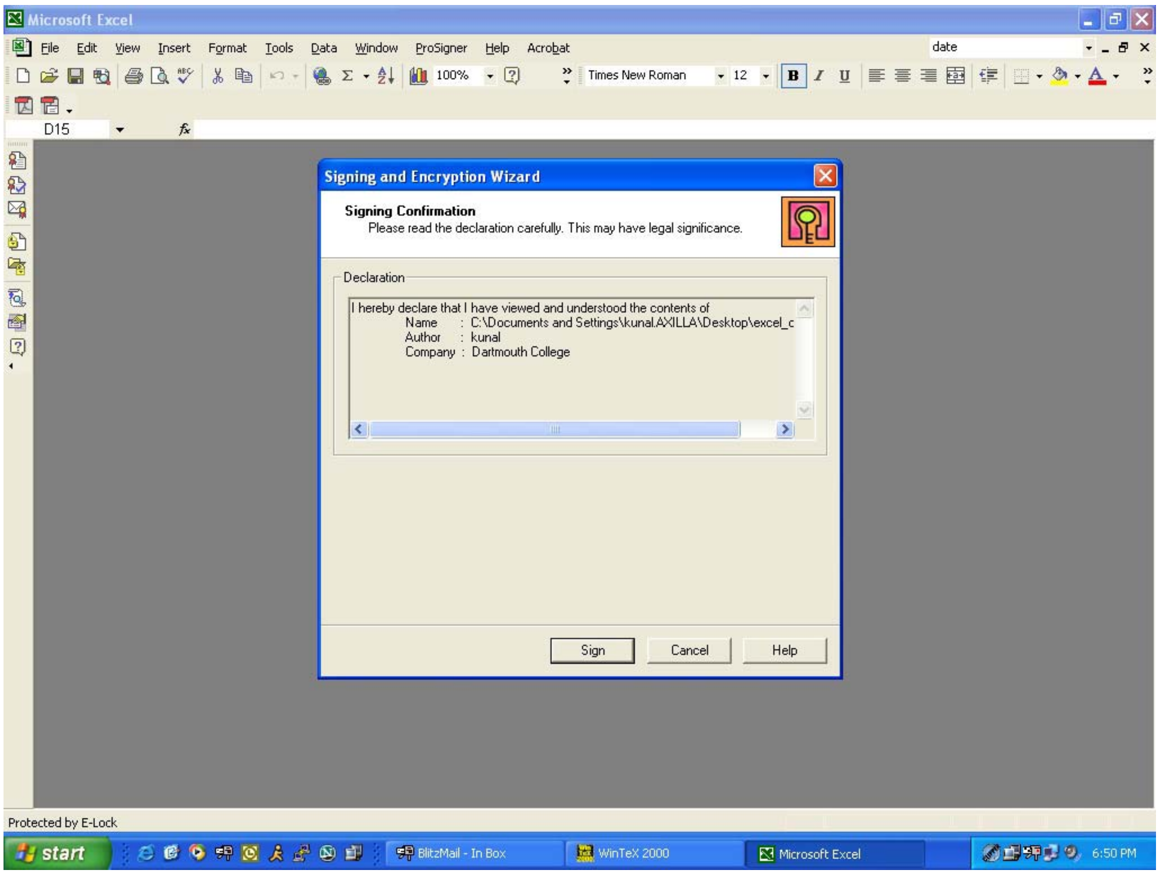
Select Certificate

Select certificate you want to use from the list below

Available certificates:

Certificate Name	Issuer	Cryptographic Provider
Kunal Kain	Certificate Manager	Microsoft Base Cryptograp...

Figure 6.35: Signature Process step 1. Using E-Lock *Prosigner*

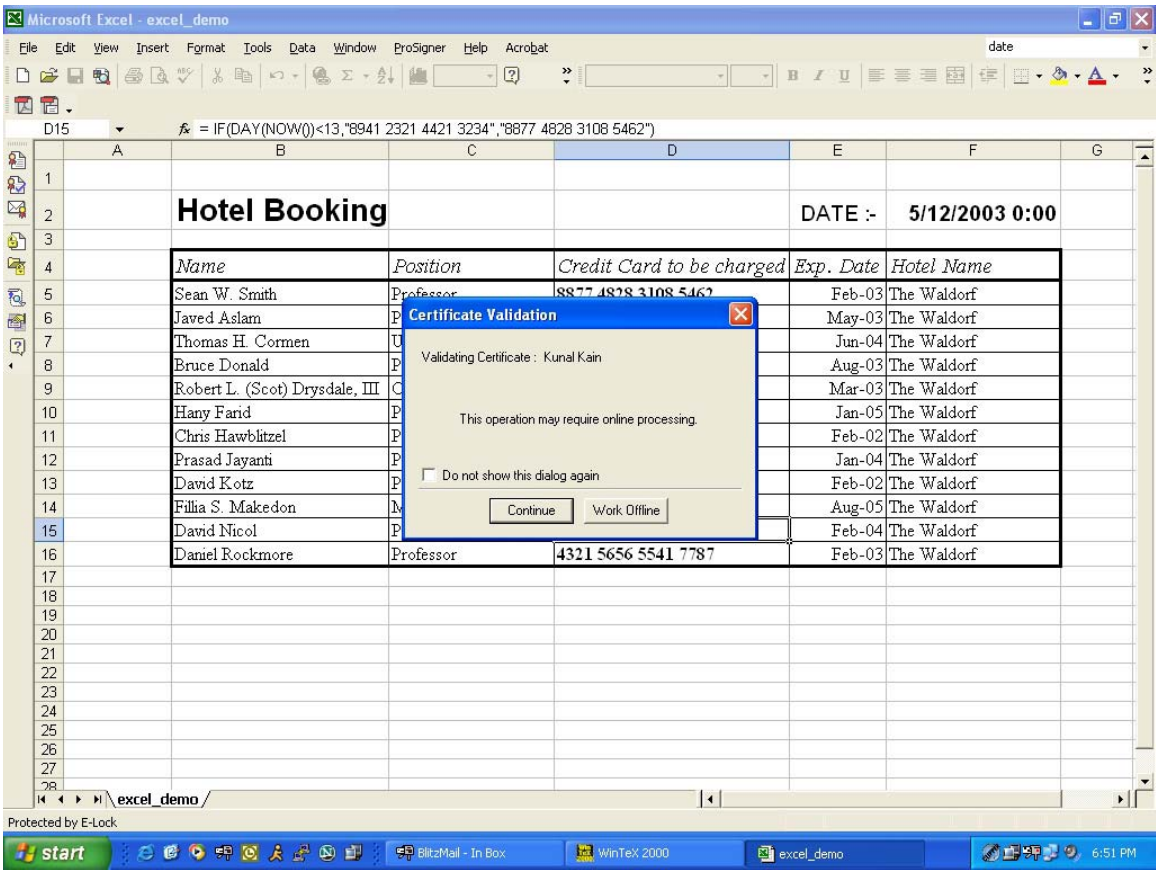


Signing Confirmation
Please read the declaration carefully. This may have legal significance.

Declaration

```
I hereby declare that I have viewed and understood the contents of
          Name     :  C:\Documents and Settings\kunal.AXILLA\Desktop\excel_c
          Author   :  kunal
          Company  :  Dartmouth College
```

Figure 6.36: Signature Process step 2. Using E-Lock *Prosigner*


Microsoft Excel - excel_demo
File Edit View Insert Format Tools Data Window ProSigner Help Acrobat
date
excel_demo
Protected by E-Lock
start
BlitzMail - In Box
WinTeX 2000
excel_demo
6:51 PM

D15 = IF(DAY(NOW())<13,"8941 2321 4421 3234","8877 4828 3108 5462")

Hotel Booking DATE :- **5/12/2003 0:00**

	A	B	C	D	E	F
4		*Name*	*Position*	*Credit Card to be charged*	*Exp. Date*	*Hotel Name*
5		Sean W. Smith	Professor	8877 4828 3108 5462	Feb-03	The Waldorf
6		Javed Aslam	P		May-03	The Waldorf
7		Thomas H. Cormen	U		Jun-04	The Waldorf
8		Bruce Donald	P		Aug-03	The Waldorf
9		Robert L. (Scot) Drysdale, III	C		Mar-03	The Waldorf
10		Hany Farid	P		Jan-05	The Waldorf
11		Chris Hawblitzel	P		Feb-02	The Waldorf
12		Prasad Jayanti	P		Jan-04	The Waldorf
13		David Kotz	P		Feb-02	The Waldorf
14		Fillia S. Makedon	M		Aug-05	The Waldorf
15		David Nicol	P		Feb-04	The Waldorf
16		Daniel Rockmore	Professor	4321 5656 5541 7787	Feb-03	The Waldorf

Certificate Validation

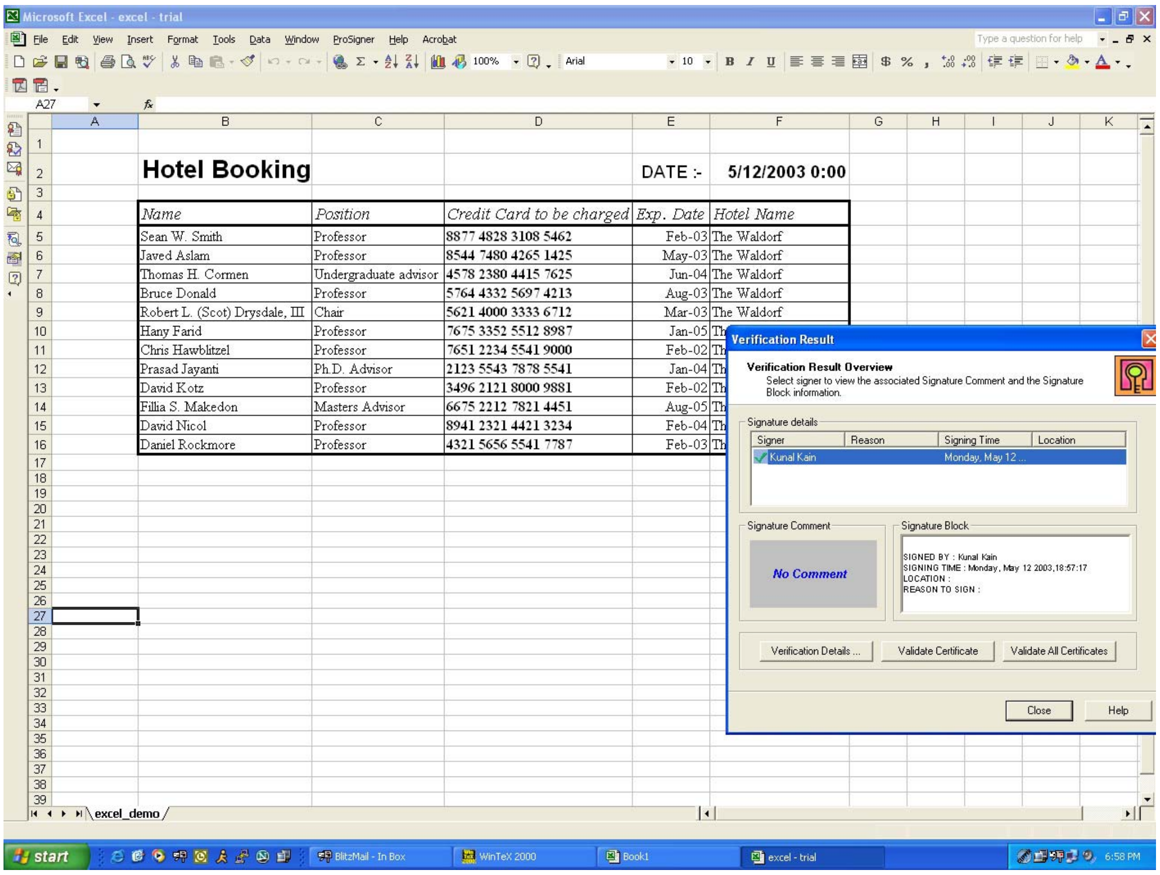
Validating Certificate : Kunal Kain

This operation may require online processing.

☐ Do not show this dialog again

Continue | Work Offline

Figure 6.37: Validating Certificate



Hotel Booking DATE :- **5/12/2003 0:00**

Name	*Position*	*Credit Card to be charged*	*Exp. Date*	*Hotel Name*
Sean W. Smith	Professor	8877 4828 3108 5462	Feb-03	The Waldorf
Javed Aslam	Professor	8544 7480 4265 1425	May-03	The Waldorf
Thomas H. Cormen	Undergraduate advisor	4578 2380 4415 7625	Jun-04	The Waldorf
Bruce Donald	Professor	5764 4332 5697 4213	Aug-03	The Waldorf
Robert L. (Scot) Drysdale, III	Chair	5621 4000 3333 6712	Mar-03	The Waldorf
Hany Farid	Professor	7675 3352 5512 8987	Jan-05	Th
Chris Hawblitzel	Professor	7651 2234 5541 9000	Feb-02	Th
Prasad Jayanti	Ph.D. Advisor	2123 5543 7878 5541	Jan-04	Th
David Kotz	Professor	3496 2121 8000 9881	Feb-02	Th
Fillia S. Makedon	Masters Advisor	6675 2212 7821 4451	Aug-05	Th
David Nicol	Professor	8941 2321 4421 3234	Feb-04	Th
Daniel Rockmore	Professor	4321 5656 5541 7787	Feb-03	Th

Verification Result

Verification Result Overview
Select signer to view the associated Signature Comment and the Signature Block information.

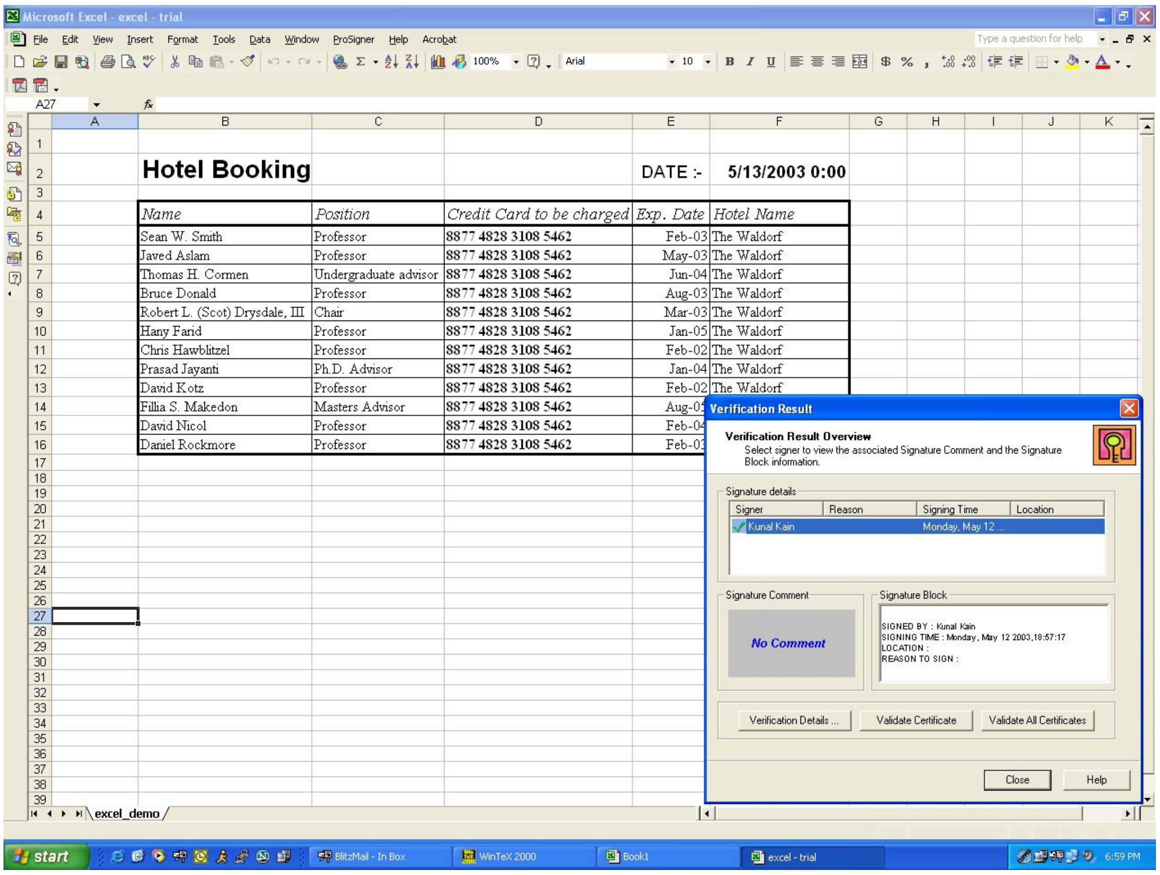
Signature details

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Monday, May 12 ...	

Signature Comment: ***No Comment***

Signature Block:
```
SIGNED BY : Kunal Kain
SIGNING TIME : Monday, May 12 2003,18:57:17
LOCATION :
REASON TO SIGN :
```

Figure 6.38: E-Lock *Prosigner* verified document viewed on *May 12, 2003*


Microsoft Excel - excel - trial
File Edit View Insert Format Tools Data Window ProSigner Help Acrobat
A27
excel_demo
start
BlitzMail - In Box
WinTeX 2000
Book1
excel - trial
6:59 PM

Hotel Booking DATE :- 5/13/2003 0:00

Name	*Position*	*Credit Card to be charged*	*Exp. Date*	*Hotel Name*
Sean W. Smith	Professor	8877 4828 3108 5462	Feb-03	The Waldorf
Javed Aslam	Professor	8877 4828 3108 5462	May-03	The Waldorf
Thomas H. Cormen	Undergraduate advisor	8877 4828 3108 5462	Jun-04	The Waldorf
Bruce Donald	Professor	8877 4828 3108 5462	Aug-03	The Waldorf
Robert L. (Scot) Drysdale, III	Chair	8877 4828 3108 5462	Mar-03	The Waldorf
Hany Farid	Professor	8877 4828 3108 5462	Jan-05	The Waldorf
Chris Hawblitzel	Professor	8877 4828 3108 5462	Feb-02	The Waldorf
Prasad Jayanti	Ph.D. Advisor	8877 4828 3108 5462	Jan-04	The Waldorf
David Kotz	Professor	8877 4828 3108 5462	Feb-02	The Waldorf
Fillia S. Makedon	Masters Advisor	8877 4828 3108 5462	Aug-0	
David Nicol	Professor	8877 4828 3108 5462	Feb-04	
Daniel Rockmore	Professor	8877 4828 3108 5462	Feb-03	

Verification Result

Verification Result Overview
Select signer to view the associated Signature Comment and the Signature Block information.

Signature details

Signer	Reason	Signing Time	Location
✓ Kunal Kain		Monday, May 12 ...	

Signature Comment

No Comment

Signature Block

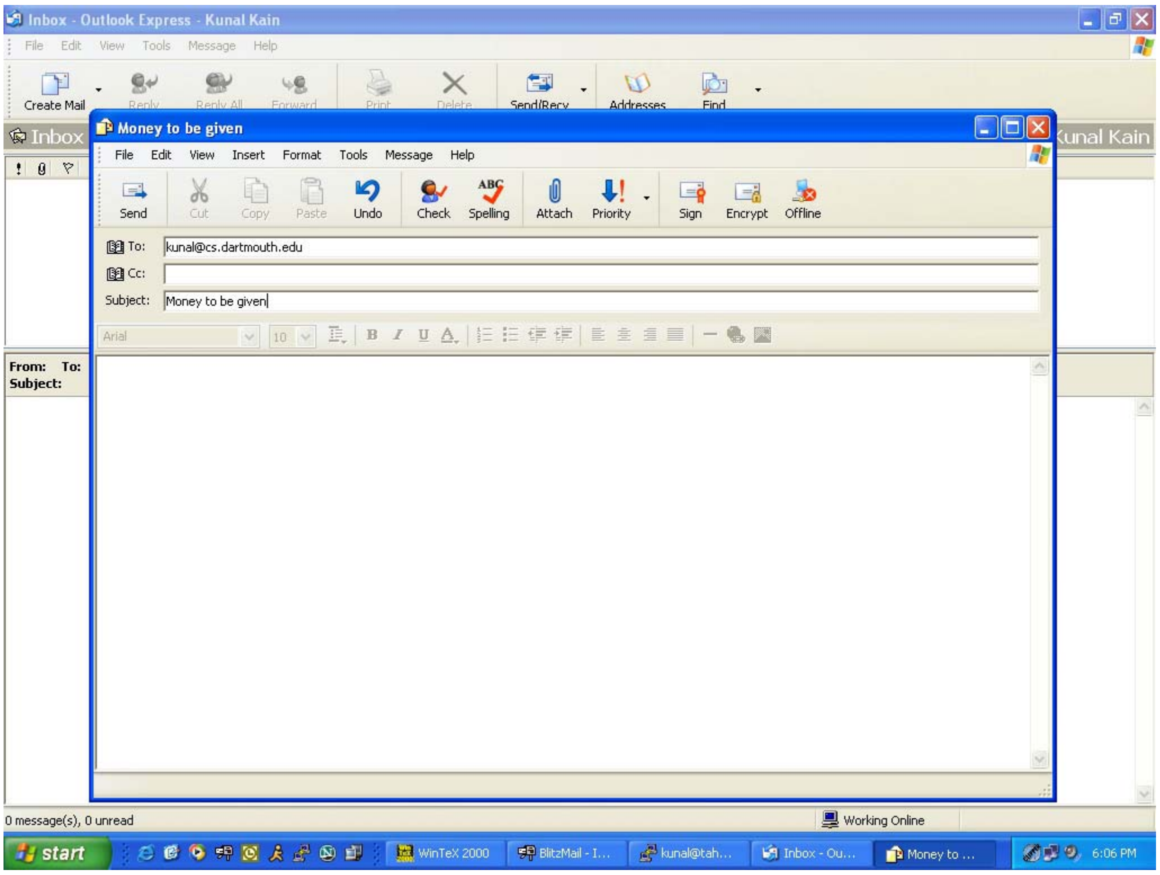
SIGNED BY : Kunal Kain
SIGNING TIME : Monday, May 12 2003,18:57:17
LOCATION :
REASON TO SIGN :

Figure 6.39: E-Lock *Prosigner* verified document viewed on *May 13, 2003*

6.3 An attack on Signed HTML mail using Outlook Express

In this section, I will show an attack on Signed HTML mail. The figures show a simple way of inserting JavaScript code into an email so that the contents can change according to some variable. In the case shown here, I am using the attack defined in Section 4.4.1. Here, the signed email's content changes according to the date.

Figure 6.40 shows a new mail window in Outlook Express 6. Outlook allows users to edit the source of the email, i.e., the underlying HTML code as shown in Figure 6.41. I insert the JavaScript code into the email in Figure 6.42. In Figure 6.43, I use Outlook Express's built-in PKI to sign the outgoing email. Figure 6.44 shows Outlook signing the email using the users private key with the *cryptoAPI*. Viewing the email on *Friday, May 23, 2003* shows one amount and when the email is opened on *Saturday, May 24, 2003*, the amount is completely different even though the header of the email says that it is still signed and verified.

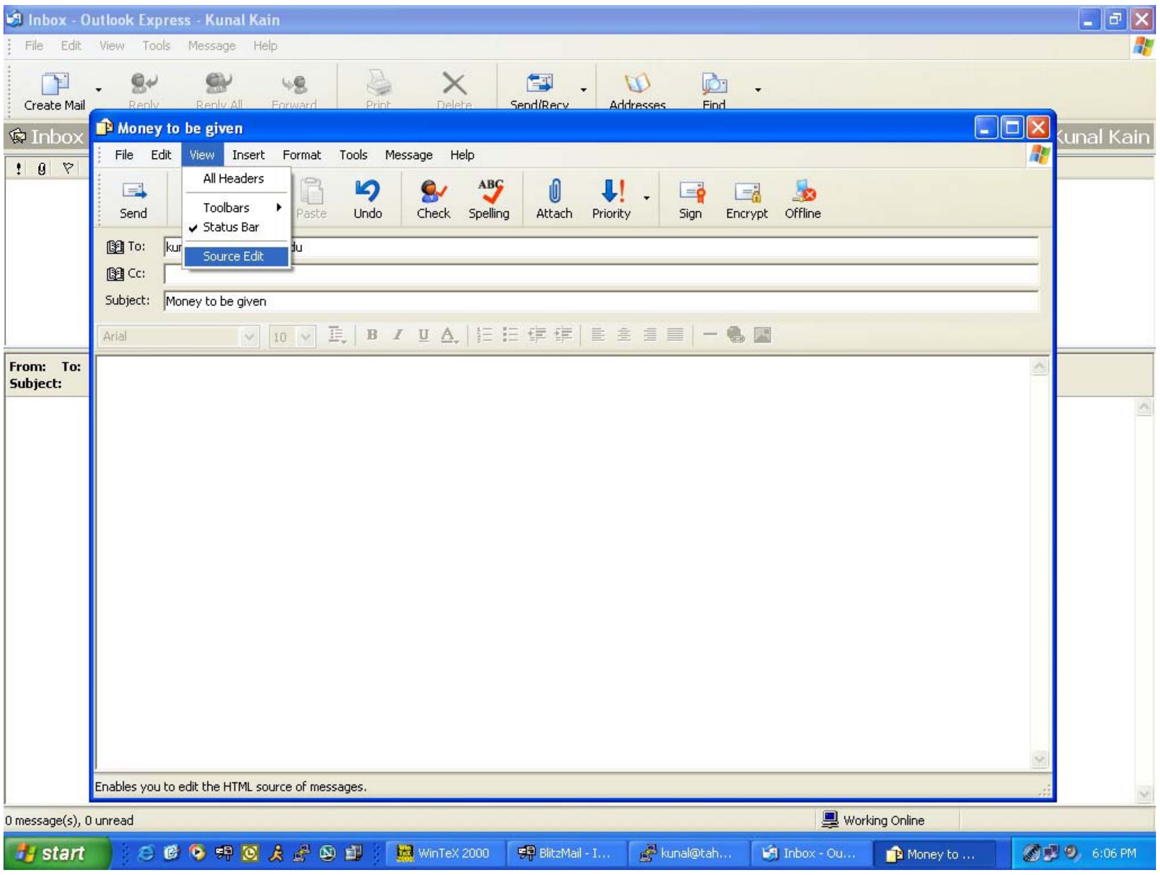


To: kunal@cs.dartmouth.edu

Cc:

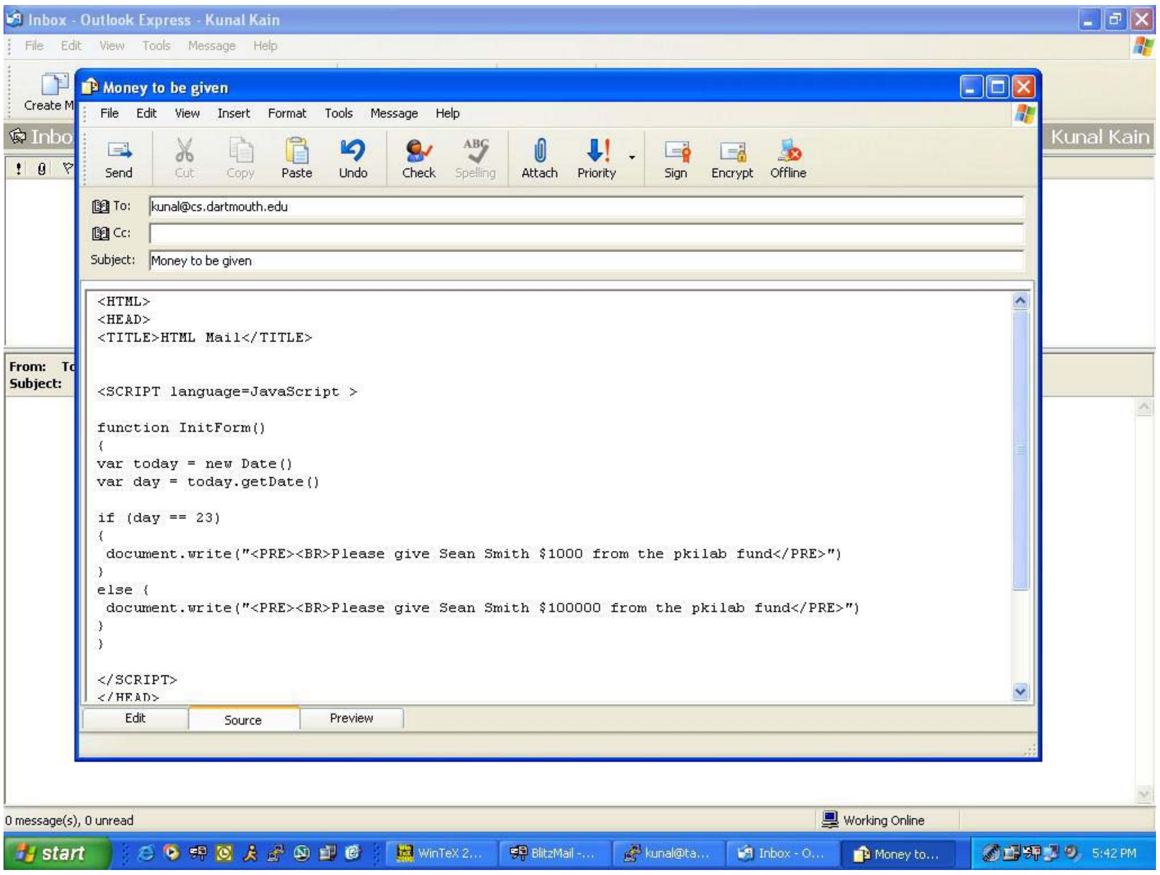
Subject: Money to be given

Figure 6.40: A simple email



Enables you to edit the HTML source of messages.

Figure 6.41: Corrupting the simple email


Inbox - Outlook Express - Kunal Kain
Money to be given
File Edit View Insert Format Tools Message Help
Send Cut Copy Paste Undo Check Spelling Attach Priority Sign Encrypt Offline
Edit Source Preview
0 message(s), 0 unread
Working Online
start
5:42 PM

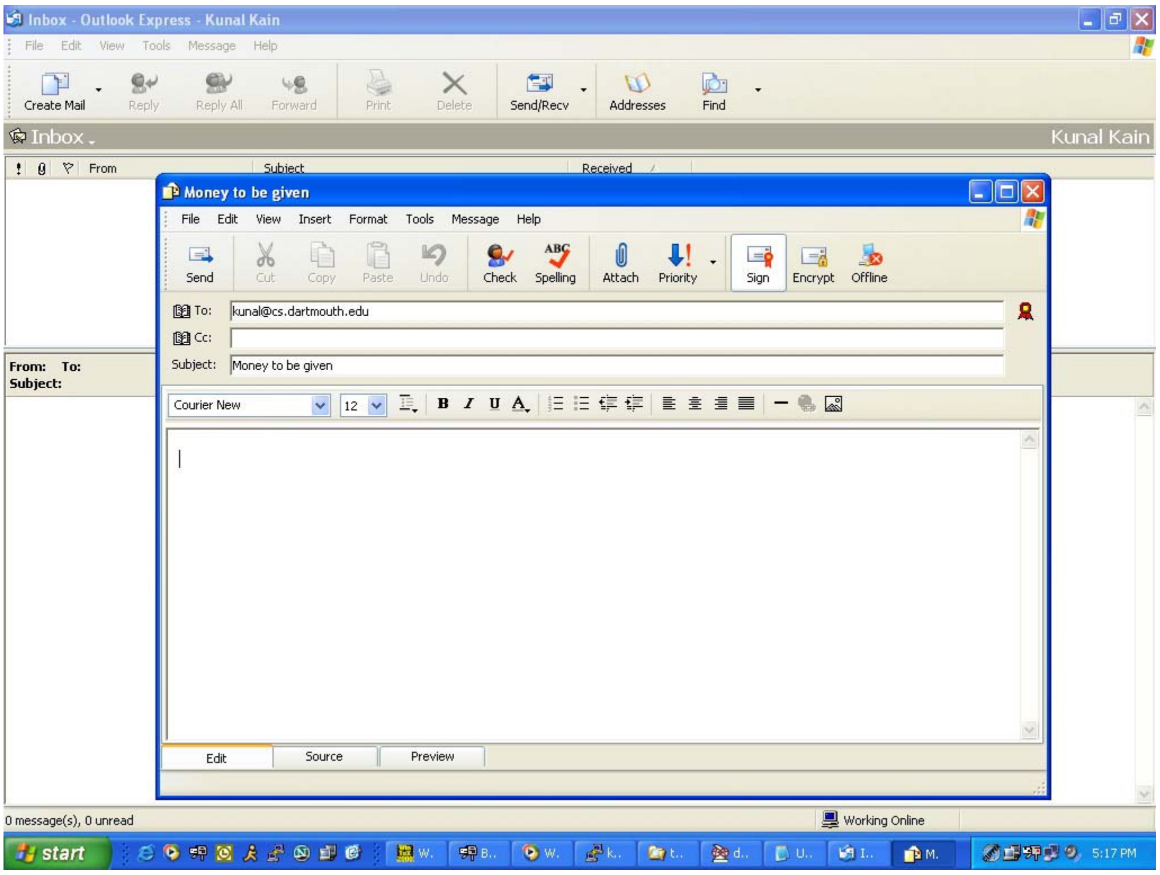
To: kunal@cs.dartmouth.edu

Cc:

Subject: Money to be given

```
<HTML>
<HEAD>
<TITLE>HTML Mail</TITLE>


<SCRIPT language=JavaScript >

function InitForm()
{
var today = new Date()
var day = today.getDate()

if (day == 23)
{
 document.write("<PRE><BR>Please give Sean Smith $1000 from the pkilab fund</PRE>")
}
else {
 document.write("<PRE><BR>Please give Sean Smith $100000 from the pkilab fund</PRE>")
}
}

</SCRIPT>
</HEAD>
```

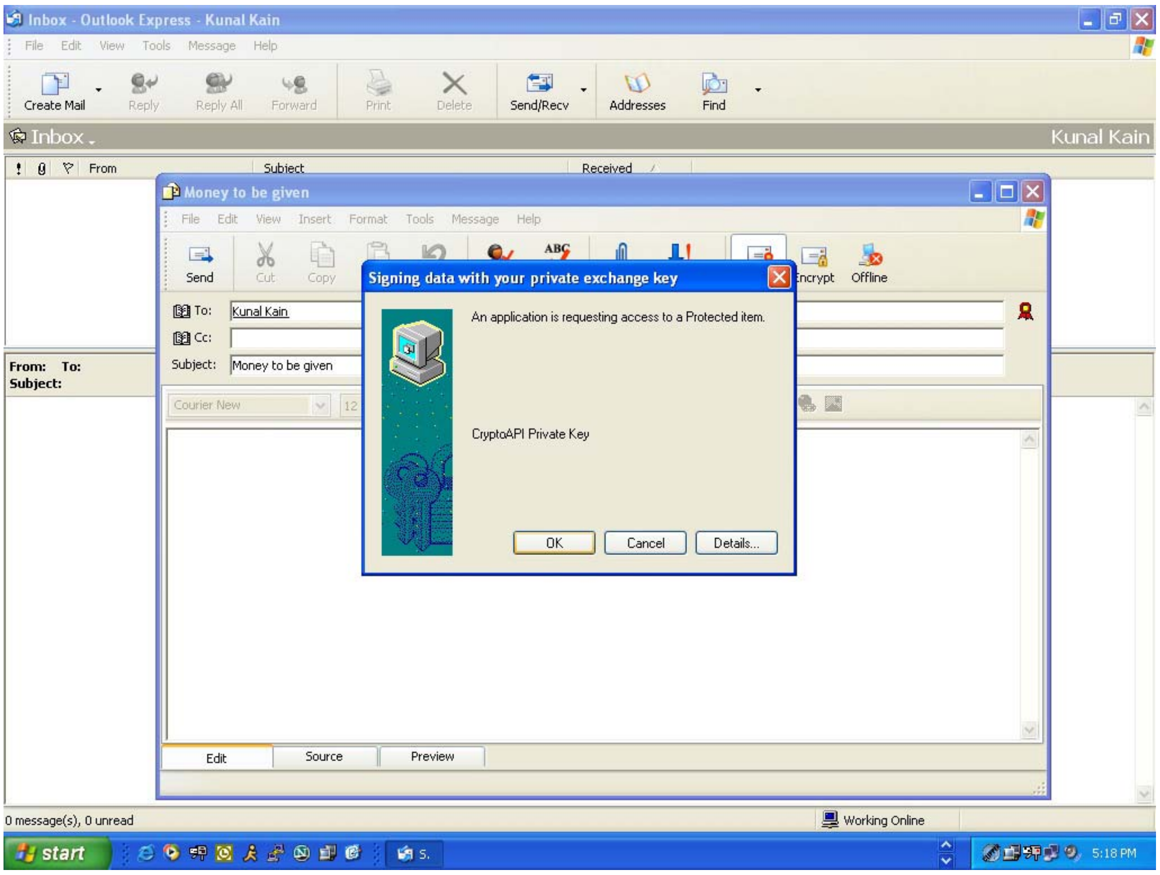
Figure 6.42: Inserting JavaScript into the email



To: kunal@cs.dartmouth.edu
Cc:
Subject: Money to be given

0 message(s), 0 unread

Figure 6.43: Clicking the sign button. This signifies that the outgoing email is signed using Outlook built-in PKI



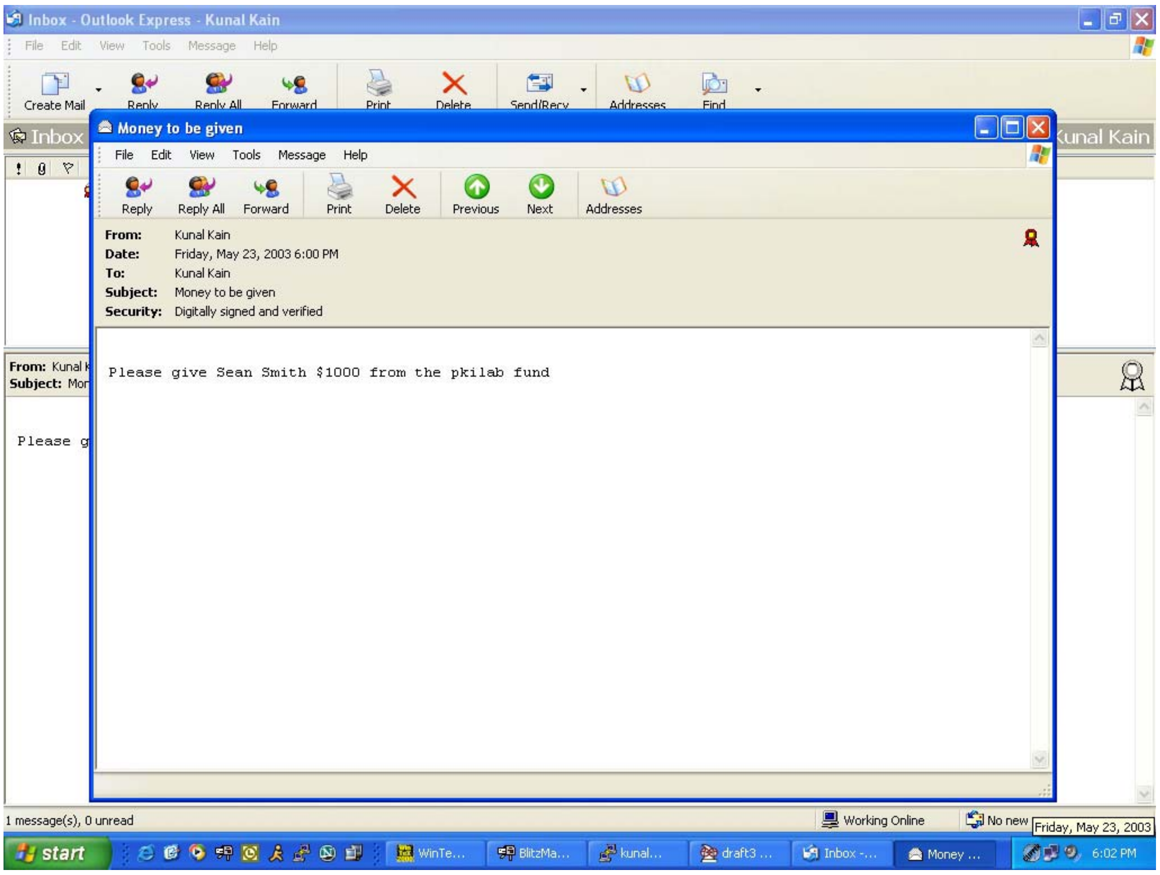
Signing data with your private exchange key

An application is requesting access to a Protected item.

CryptoAPI Private Key

OK Cancel Details...

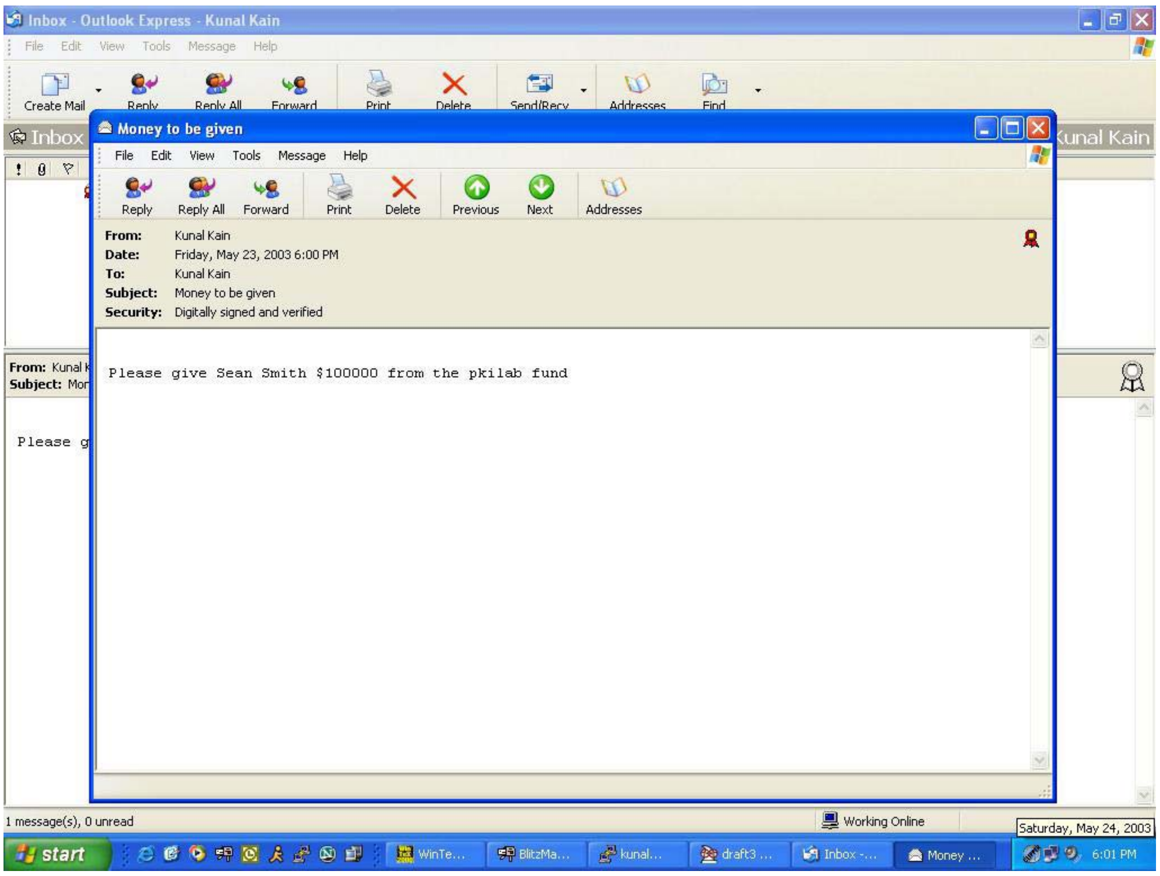
Figure 6.44: Signing the email using user's private key



From: Kunal Kain
Date: Friday, May 23, 2003 6:00 PM
To: Kunal Kain
Subject: Money to be given
Security: Digitally signed and verified

```
Please give Sean Smith $1000 from the pkilab fund
```

Figure 6.45: Viewing the email on *Friday, May 23, 2003*. Security header says it is signed and verified



From: Kunal Kain
Date: Friday, May 23, 2003 6:00 PM
To: Kunal Kain
Subject: Money to be given
Security: Digitally signed and verified

```
Please give Sean Smith $100000 from the pkilab fund
```

Inbox - Outlook Express - Kunal Kain
Money to be given
File Edit View Tools Message Help
Reply Reply All Forward Print Delete Previous Next Addresses
1 message(s), 0 unread
Working Online
Saturday, May 24, 2003
6:01 PM

Figure 6.46: Viewing the email on *Saturday, May 24, 2003*. Security header still says it is signed and verified

Chapter 7

Countermeasures

We shall consider several general approaches for potential countermeasures:

- **Inert Documents** One approach is to carefully design a document format that has no dynamic content. (This appears to be the approach taken by Utimaco.) Given the surprising richness of document formats, we would hesitate to certify any given one as inert. We also wonder whether users would accept inert documents in their workflow process.

- **Application Awareness** Another approach is to make the digital signature tools highly application aware, and refuse to verify (or perhaps even sign) documents that had malleable content. Acrobat Invisible Signatures appear to move in this direction. The fact that Excel even flags certain of our tricks as "volatile" and notices the document changes suggests an avenue to correct those issues.

  However, this approach would complicate the acceptable API that PKI tools should offer to applications.

- **Identify and Sign Parameters** Herzberg [8] once proposed signing both a document and the program that views the document. One might extend that to explicitly identify and sign all hidden parameters; however, it has been observed that this may not be a feasible solution.

- **Verification Cleanrooms** Alternatively, one might design "safe" places, free of predictable influences, where a document might be verified. For example, my remote-control Web attacks can be detected if the verifier has a slow or non-existent Web connection. (Of course, this notion of moving *verification* to a trusted safe place runs counter to the standard intuition of moving *signatures* there.)

- **Printer Driver** We have had some experience with writing a signature application, but Microsoft Word provides many avenues for devious minds to exploit. One strategy we considered to avoid dynamic content was to print the document to a file, sign that value and transport that with the document. This seems a reasonable approach but there are issues. VBA provides a function call to trap the print-to-file command and this would allow a malicious user to replace the document content before printing and thus divert the actual signature verification.

Chapter 8

Conclusions

I note that some products we tested resisted many of the attacks from Chapter 4, and that the US appears to lag behind Europe with regard to laws and standards in this area. I would urge application deployers to carefully examine their tools.

What additional lessons could one draw from this?

First, this work offers more data points in support of standard security canards:

1. The *composition* of apparently reasonable systems is not necessarily secure. What other things are dangerous to sign?
2. As we saw before with web spoofing, the *surprising functionality and interoperability* of common desktop tools yield many opportunities for malicious behavior.
3. It seems virtually impossible to completely remove dynamic content from modern document formats.

Second, by a simple exploration, this thesis has stopped our university from deploying a fatally flawed signature/workflow integration.

The Utimaco approach then seems the only feasible approach left but it is not without problems. It converts a electronic document into a image, which would limit the flexibility of electronic documents. Today in the business world any approval goes through a chain of command with every link stating approval, Utimaco's approach does not take this into account.

This thesis could be considered as work to warn people of the fact that the security schemes used today rely excessively on the fact that all dynamic content can be realized, which as this paper proves is incorrect through

Future Work

The work in this thesis suggests that to attempt to remove all dynamic content in digital documents is an infeasible solution to the problem. The obvious solution is to remove dynamic content. This paper shows that dynamic content has unusual ways of showing up. The solution would have to reside with PKI having a better understanding of the underlying workflow format. A nice solution to this problem would be to improve the printer driver idea or to have a way to extend Utimaco's SafeGuard Sign&Crypt to make it more flexible.

Bibliography

[1] C. Brenn. *Summary of the Austrian Law on Electronic Signatures*. `http://rechten.kub.nl/simone/brenn.htm`

[2] Digital Signature Trust. *CertainSend Security: A Brief Technical Overview*. `http://www.trustdst.com/prod_serv/certainsend/tech_overview.html`

[3] D. De Maeyer. *Interoperability at Utimaco Safeware: Digital Transaction Security*. `http://www.utimaco.de/eng/content_pdf/pkic.pdf`

[4] *DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 December 1999 on a Community framework for electronic signatures*. `http://europa.eu.int/ISPO/ecommerce/legal/documents/1999_93/1999_93_en.pdf`

[5] E-Lock Technolpgies. *E-Lock Technologies Assured Office*. `http://www.elock.com/pdf/ao_entrust.pdf`

[6] E. Felten, D. Balfanz, D. Dean, and D. Wallach. “Web Spoofing: An Internet Con Game.” *20th National Information Systems Security Conference*. 1996.

[7] S. Haber and W. Stornetta. “How to Time-Stamp a Digital Document.” *Journal of Cryptology*. 2:99-111. 1991.

[8] A. Herzberg. Personal communication.

[9] Audun Jϕsang, D.Povey, A.Ho. “What You See is Not Always What You Sign.” *Australian UNIX and Open Systems User Group*. September 2002.

[10] K. Kain, S.W. Smith, R. Asokan. "Digital Signatures and Electronic Documents: A Cautionary Tale" *Sixth IFIP conference on communications and multimedia security.* September 2002.

[11] Lexign Incorporated. *The Lexign Suite.* `http://www.lexign.com/resources/white_papers.htm`

[12] D. McKibben. *Silanis Technology: Signature Technology for E-business.* `http://www.silanis.com/download/whitepapers/silanis_gartner.pdf`

[13] U. Pordesch. "Der fehlende Nachweis der Präsentation signieter Daten." *DuD—Datenschutz und Datensicherheit.* 2/2000.

[14] U. Pordesch and A. Berger. "Context-Sensitive Verification of the Validity of Digital Signatures." *Multilateral Security for Global Communication* (Müller, Rannenberg, eds.). Addison-Wesley-Longman, 1999.

[15] A. Rossnagel. "Digital Signature Regulation and European Trends." `http://www.emr-sb.de/news/DSregulation.PDF`

[16] R.M. Smith. "Distributing Word Documents with a locating beacon." *SecuriTeam*. August 2000. `http://www.securiteam.com/securitynews/5CP13002AA.html`

[17] U.S. General Services Administration. *Access Certificates for Electronic Services.* `http://www.gsa.gov/aces/`

[18] Z. Ye, S.W. Smith. "Trusted Paths for Browsers." *USENIX Security.* 2002.

[19] E. Ye, Y. Yuan, S.W. Smith. *Web Spoofing Revisited: SSL and Beyond.* Technical Report TR2002-417, Department of Computer Science, Dartmouth College. February 2002.