6.857 Computer and Network Security	**Fall Term, 1997**

Lecture 20 : November 13, 1997

Lecturer: Ron Rivest *Scribe: Lydia Sandon*

1 Today's Topics

- In the news
- Certification
  - Introduction
  - PGP
  - X509

2 In the news

The following are some interesting bugs and related issues from the news this week:

Intel Pentium Bug

Running the instruction `F0 0F C7 C8` will currently crash any Intel Pentium, regardless of operating system or software from which it is called. The instruction, however, has no documented use, so this bug is unlikely to affect most people.

While there are no security attacks because of this bug, there are denial of service attacks. If you download an ActiveX control, it can crash your machine utilizing this bug.

The bug may possibly have been put there as a trap door, but it was most likely just a bug that random checking of instructions never came upon. It is agreed that the floating-point division bug was a much worse problem, as was the one in which dividing by 0 would put you in priviledged mode. Intel says they are going to fix this one very quickly.

Microsoft Internet Explorer Bug

Apparently, in resource URLs (`res://`), IE doesn't check for array bounds. So past the first 256 characters, you can write anything you want, and it will be written to disk. So you could write an entire virus starting at the 257th character of such a URL, and it would execute on the user's machine.

Sun Microsystems Falsified Java Benchmarks (not a bug)

Recently, it was found that Sun Microsystems falsified their benchmarks in some way as to prepare the system to run the particular timing tests very fast. This isn't a bug, but an interesting point.

3 Certification

In the real world, we don't just have Alice and Bob dealing with each other's public keys. We have to add more people to the system, and along with the people come scaling problems. What operations are needed to make public keys work in practice?

- Give me Alice's public key
  - also: Encrypt this message with Alice's public key
  - key lookup based on name
- Who signed this message
  - name lookup based on key
  - associate name with a public key

The large namespace makes these operations difficult to implement. Also, the names should be unique, recognizable by human beings, and easy to use. If such a naming scheme is difficult to use, it becomes a security issue.

Certification attempts to bind names (such as unique email addresses) to keys. A certificate might be a signed document that says that "rivest@mit.edu"'s key is n_{rivest} and e_{rivest}. Current systems to try to do this are PGP, X509, and SDSI.

3.1 Who has the authority to give away certificates?

There is no existing infrastructure for this, yet. There are also many ways to go about it, which are outlined here.

PGP uses a web of trust

- Relies on the principle that everyone is related by several degrees of separation.
- local, bottom-up
- Example (see figure): rivest@mit.edu signs a key for alice@mit.edu, thereby vouching for the validity of her key. She can then sign for whomever she wants, including:
  - a real user, bob@mit.edu (key K_3), or
  - a faked user, bill@whitehouse.gov (key K_{bill}), which she creates and maintains for sinister reasons.

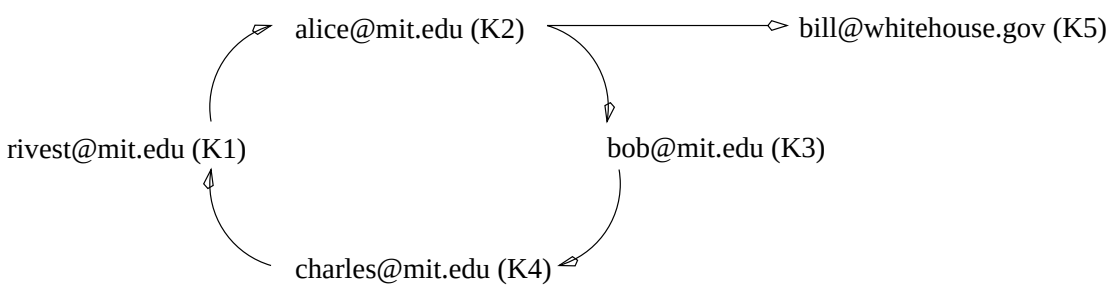

Figure 1: Sample PGP web of trust.

There are issues that arise with this system as follows:

- Is signing a key transitive? That is, because rivest@mit.edu signs alice@mit.edu's key, does that mean that he believes that any key she signs is valid? Currently, in such a system, you must find a path from you to the person with whom you want to communicate, and evaluate whether you trust that path. Path Server, by AT&T, is an online service to find the path between any two keys.

- Alice could, indeed, fake that key from Bill Clinton, and use it to fake email from him.
- In conclusion, it is a nice system, but it doesn't scale very well.

Hierarchical systems

There could be any number of hierarchical systems, such as the one pictured in Figure 2 below, which could be used not only for signing certificates, but also for providing authorization to certain services:

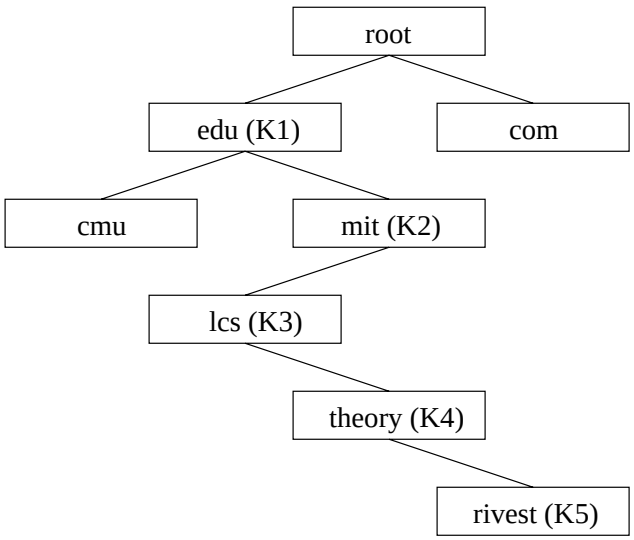

Figure 2: A sample hierarchy.

In this case, the root is responsible for signing the certificates of the .edu, .com, .net, and .org nodes, as well as any other that report directly to it. The .edu node, in turn, signs the certificates of mit.edu, cmu.edu, and all of the other .edu domains. In turn, mit.edu signs lcs.mit.edu, media.mit.edu, etc. This goes down the chain until an individual person (here, rivest@mit.edu) has his certificates validated (this is a slow process).

X509 uses a different hierarchy, where the levels start at "country," "state," "organization," "division," and continue down to "name." In the previous example hierarchy, the levels were more clearly linked to DNS organizations. A sample key for Professor

Rivest might be "Country=US, State=MA, Org=MIT, ... Name=Rivest." It is a similar structure with a clear outline of authority. However, there are some problems with it:

- What if there is more than one email address to a person?
- What if one person belongs to more than one organization?
- Too much trust is placed on the top nodes (if the root node is compromised, the whole system is),
- Maintenance (ten percent of the country changes jobs every year),
- Names are long, when they should be
  - unique, and
  - readable by human beings.
- Some possibly unwarranted trust in low-level nodes, and
- losing keys, moving, departmental name change.

How do you deal with change?

- Expiration dates on certificates (in any of the cases),
- Signed forwarding addresses (in the case of moving, etc), and
- Revocation (in the case of key compromise, move, change of organization, etc).

Dealing with revocation is a particular challenge.

- It is possible to maintain a system with no certificates which uses online certification when necessary.
- Revocation lists
  - Each certification authority has a list of all of the certificates that have been revoked,

  - Issued periodically and signed by the certification authority. But how do you know that you have the most recent revocation list? You can put an expiration date on each one that says something along the lines of, "This revocation list is good until the next of its kind is released on 1 December 1997."

- Silvio Micali presents another approach (see below).

The approach used by Silvio Micali is as follows: a certificate contains a name, key, date, and values called x_0 and y_0. There is also a known hashing function. If someone queries a server to find out whether a particular certificate for Bob is still is still valid. The following is returned to him.

```
 ---------
| certificate:
|
|   name
|   key
|   x0        <--(h)--x1 <--(h)--x2 <--(h)--x3 <--(h) ... <--(h)--x365
|   y0        <--(h)-- y1
|   date
 ---------
```

Figure 3: Silvio Micali's approach to certification.

In this case, the fields within the dotted lines are what Bob presents as his certificate. The server sends back a value. If the server sends back y_1, then the ticket is revoked (where y_1 is the hash of y_0). If the server sends back an xi, then the certificate was still valid i days after creation, and x_i is x_0 hashed i times with the same known algorithm.

3.2 What prevents you from giving away your private key?

All of the techniques used for public-key cryptography assume that a private key stays private; specifically, that no one will voluntarily give away their private key. People are worried about

- liability,

- legal contracts,
- loss of reputation (assuming key is associated with name), and
- detecting double-use (stolen v. given away).