Defence Research and Development Canada Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Investigation on Vulnerabilities of Pre-boot and Post-boot Authentication

Xinzhi Liu and Lijun Wang

Defence R&D Canada – Ottawa
CONTRACT REPORT
DRDC Ottawa CR 2006-083
March 2006

Canada

Investigation on Vulnerabilities of Pre-boot and Post-boot Authentication

Xinzhi Liu
Lijun Wang

Prepared by:

University of Waterloo
Department of Applied Mathematics
200 University Avenue West
Waterloo, Ontario N2L 3G1

Contract Number: W7714-5-0943
Contract Scientific Authority: Q. Xiao 613-998-1245

Defence R&D Canada – Ottawa

Contract Report
DRDC Ottawa CR 2006-083
March 2006

Abstract

This proposal provides an assessment and clear understanding of pre-boot and post-boot authentication for the future development of biometric authentication system in high security applications. We first give a technology review of current PC boot process, present a brief summary of mainly existing boot techniques and analyze the advantages and disadvantages of pre-boot and post-boot strategies. A brief summary of pre-boot and post-boot processes is then given, which presents the comparisons among different technologies.

Next, we give an insight view of authentication technologies for pre-boot and post-boot stages respectively, with a particular focus on new biometric techniques. The security differences between pre-boot and post-boot authentication are studied and a comparison among different strategies is presented. Based on the disadvantages and limitations of current authentication techniques, several alternative authentication solutions are proposed, especially for the wireless devices in ad hoc networks.

Résumé

Cette proposition présente une évaluation et une description claire de l'authentification pré- et post-démarrage en vue du développement ultérieur d'un système d'authentification biométrique dans les applications de haute sécurité. Nous commençons par une revue technologique des processus actuels de démarrage des PC, présentons un bref résumé des principales techniques de démarrage existantes et analysons les avantages et les inconvénients des stratégies pré- et post-démarrage. Nous proposons ensuite un bref résumé des processus pré- et post-démarrage, avec un comparatif des différentes technologies.

Nous présentons ensuite un aperçu des technologies d'authentification pré- et post-démarrage respectivement, en nous intéressant plus particulièrement aux nouvelles techniques biométriques. Nous étudions les différences, du point de vue de la sécurité, entre l'authentification pré-démarrage et post-démarrage, et présentons un comparatif des différentes stratégies. Diverses solutions d'authentification, qui prennent en compte les inconvénients et les limitations des techniques d'authentification actuelles, sont proposées, tout particulièrement à l'égard des appareils sans fil dans les réseaux ad hoc.

This page intentionally left blank

Executive summary

With the rapidly increasing use of networks of computers for the transfer of highly sensitive data or message such as commands for controlling operations of deadly weapons, commercial electronic fund transfer, electronic business transactions etc., the need for establishing a reasonable level of assurance of authentication is becoming of paramount importance.

The objective and purpose of this work is to provide an assessment and clear understanding of pre-boot and post-boot authentication for the future development of biometric authentication system in high security applications. We first give a brief summary of pre-boot and post-boot processes, review the current techniques for PC boot and analyze the advantages and disadvantages of pre-boot and post-boot strategies. Authentication issue is then studied for pre-boot and post-boot stages respectively. Current authentication technologies are reviewed and evaluations for different methods are presented.

With extensive use of wireless network, authentication in the field of military is getting more and more important since the message sent through the wireless network is easier to be intercepted, stolen and even changed. Based on the unique features of wireless network and the limitations of current authentication techniques, several alternative authentication solutions are proposed, especially for the wireless devices in ad hoc networks.

This proposal provides an insight view of current authentication technologies and compares existing authentication systems on different security levels. With the comparison of different authentication technologies, users can choose most suitable method according to the requirement of their systems. Some authentication problems usually neglected by researchers are discussed here and feasible solutions are proposed to improve the security.

Liu, X., Wang, L. 2006. Investigation on Vulnerabilities of Pre-boot and Post-boot Authentication. DRDC Ottawa CR 2006-083. University of Waterloo.

Sommaire

Alors que l'utilisation des réseaux d'ordinateurs pour le transfert de messages ou de données très sensibles comme des commandes pour la conduite d'armes mortelles, le transfert commercial de fonds électroniques, les transactions commerciales électroniques, etc. se répand de plus en plus, la nécessité d'établir un niveau raisonnable d'assurance dans l'authentification prend une importance primordiale.

L'objectif et le but de ce travail consistent à fournir une évaluation et une description claire de l'authentification pré-démarrage et post-démarrage en vue du développement ultérieur d'un système d'authentification biométrique dans les applications de haute sécurité. Nous commençons tout d'abord par présenter un bref sommaire des processus pré-démarrage et post-démarrage, nous examinons les techniques actuelles pour le démarrage d'un PC et analysons les avantages et les inconvénients respectifs des stratégies pré- et post-démarrage. Nous étudions ensuite le problème de l'authentification dans les étapes pré- et post-démarrage, respectivement. Nous examinons les technologies d'authentification et évaluons les différentes méthodes.

Alors que l'utilisation des réseaux sans fil se répand de plus en plus, l'authentification dans le domaine militaire prend de plus en plus d'importance, puisque les messages transportés dans ce genre de réseau sont plus faciles à intercepter, à dérober et même à modifier. Diverses solutions d'authentification, qui tiennent compte des caractéristiques particulières des réseaux sans fil et des limitations des techniques d'authentification actuelles, sont proposées, tout particulièrement à l'égard des appareils sans fil dans les réseaux ad hoc.

Cette proposition présente un aperçu des technologies d'authentification actuelles et compare les systèmes d'authentification existants à divers niveaux de sécurité. À partir de ce comparatif des différentes technologies d'authentification, l'utilisateur peut choisir la méthode la mieux adaptée aux besoins particuliers de son système. Nous étudions également ici un certain nombre de problèmes d'authentification qui sont souvent négligés par les chercheurs, et des solutions réalisables sont proposées afin d'améliorer la sécurité.

Table of contents

List of figures

List of tables

Technology overview of PC boot process

With the rapidly increasing use of networks of computers for the transfer of highly sensitive data or message such as commands for controlling operations of deadly weapons, commercial electronic fund transfer, electronic business transactions etc., the need for establishing a reasonable level of assurance of authenticity of a received message, before executing it, is becoming of paramount importance. Furthermore, if a received message can be demonstrate to be authentic, it can be conjectured that the message has not been modified en route i.e., it has reached the destination intact.

Authentication schemes protecting information systems can be classified into two general patterns: pre-boot authentication and post-boot authentication. Before talking about the authentication, we will begin with the PC boot process to see how the computer works in the stage.

Introduction

To boot (as a verb; also "to boot up") a computer is to load an operating system into the computer's main memory or random access memory (RAM). Once the operating system is loaded (and, for example, on a PC, you see the initial Windows or Mac desktop screen), it's ready for users to run applications. Sometimes you'll see an instruction to "reboot" the operating system. This simply means to reload the operating system (the most familiar way to do this on PCs is pressing the Ctrl, Alt, and Delete keys at the same time).

On larger computers (including mainframes), the equivalent term for "boot" is "initial program load" (IPL) and for "reboot" is "re-IPL". Boot is also used as a noun for the act of booting, as in "a system boot." The term apparently derives from bootstrap which is a small strap or loop at the back of a leather boot that enables you to pull the entire boot on. There is also an expression, "pulling yourself up by your own bootstraps," meaning to leverage yourself to success from a small beginning. The booting of an operating system works by loading a very small program into the computer and then giving that program control so that it in turn loads the entire operating system.

Booting or loading an operating system is different than installing it, which is generally an initial one-time activity. (Those who buy a computer with an operating system already installed don't have to worry about that.) When you install the operating system, you may be asked to identify certain options or configuration choices. At the end of installation, your operating system is on your hard disk ready to be booted (loaded) into random access memory, the computer storage that is closer to the microprocessor and faster to work with than the hard disk. Typically, when an operating system is installed, it is set up so that when you turn the computer on, the system is automatically booted as well. If you run out of storage (memory) or the operating system or an application program encounters an error, you may get an error message or your screen may "freeze" (you can't do anything). In these events, you may have to reboot the operating system.

How boot process works

In this section, we will explain in moderate detail what happens when a Linux system starts up [1]. As far as possible, we will try to separate features that are specific to the various Linux distributions from those that are generic. When this is not possible -- because the explanation would be too convoluted -- we will use the RedHat set-up as an example. In addition, we will focus on the Intel/PC platform, for the same reason.

To break the process into manageable pieces, we have broken it into four stages: the `firmware' stage, the `bootloader' stage, the `kernel' stage, and the `init' stage. These are my names, and are not necessarily used by other Linux users. Moreover, it isn't always easy to separate the `firmware' stage from the initial operations of the bootloader. On the PC platform, the firmware is so unintelligent that a separate (software) bootloader is required. On other platforms, notably Sparc machines, the firmware is quite sophisticated, and may be able to load a kernel directly.

Stage1 (firmware stage)

The purpose of a bootloader is to get at least part of the operating system kernel into memory and running. After that, the kernel can take over the process. However, unless the bootloader is in firmware, to run the bootloader we must first retrieve it, from disk or wherever else it is stored. The purpose of the firmware stage, therefore, is to get a bootloader into memory and run it.

On the Intel/PC platform, the firmware stage (which does not depend on the operating system) is governed by the BIOS (Basic Input/Output System). Most modern PCs (and other types of computer, of course) can boot from floppy disk, hard disk, or CD-ROM. It is common for Sparc-based systems to have built-in network bootloaders in firmware but, at present, this is unusual in the PC world. The BIOS typically provides a mechanism by which the operator can choose the devices that will be used to boot, and it will probably be prepared to try more than one if necessary. The process is slightly different for the different media types.

Bootloader on floppy disk or hard disk

This is usually the simplest situation. On a floppy disk, the first sector is reserved as the boot sector. It must contain executable program code. The BIOS loads the boot sector into memory and then runs it. This process is largely the same whatever the hardware platform.

The situation is similar for PC hard disks, except that it is conventional to divide the hard disk into partitions, and to provide a boot sector for each partition. In the world of DOS, the boot sector was, and remains, combined with the partition table; the partition table controls how much space is allocated to each partition. In addition to the partition boot sectors there is an overall boot sector/partition table called the `master boot record' (MBR). When booting from a hard disk formatted this way, the PC BIOS loads the

MBR and executes it as a boot sector; the code in the MBR will then find which partition to boot from, and load and run the boot sector from that partition.

Linux has no need to follow the convention of partitioning that is meaningful to DOS/Windows, but if the hard disk is to be used with more than one operating system then it is a good idea to.

So, when booting from a hard disk the Linux bootloader can be placed in the MBR, or in a partition boot sector. In the latter case, it won't be the BIOS that will load the Linux bootloader, it will be the bootloader on the master boot record. Whether the boot disk is a hard disk or a floppy disk, the first stage of the boot process finds a boot sector, which will contain the Linux bootloader, and runs it.

Bootloader on CD-ROM

The ability to boot from a CD-ROM has been commonplace on most platforms for some years. On some platforms a bootable CD-ROM has the same structure as a bootable hard disk: a boot sector followed by a load of data. A structure like this is unworkable for PCs, owing to limitations in the BIOS specification. Most modern PCs are, however, able to boot from a CD-ROM formatted according to the El Torito specification. This process is far more complex than it ought to be. Because the BIOS can't cope with a full-sized bootable Linux filesystem on a CD-ROM, El Torito requires that the CD-ROM be provided with an additional bootable filesystem. This filesystem is considered to be `outside' the normal data area of the CD-ROM, and won't be visible if the CD-ROM is mounted as a filesystem in the usual way. In fact, although the CD-ROM itself will normally be formatted with an ISO9660 filesystem, the El Torito bootable image can be of any filesystem type. In practise, the bootable image will be formatted as a floppy disk: a boot sector followed by a filesystem. When booting from the CD-ROM, the BIOS finds the bootable filesystem image, loads the boot sector, and makes the rest of the image available through BIOS calls just as it does for a floppy disk. As far as the bootloader is concerned, therefore, the BIOS treats a bootable CD-ROM as an ordinary CD-ROM with an `embedded' bootable floppy disk. Booting from CD-ROM is therefore just like booting from a floppy disk in practise. With Linux, this embedded floppy disk is usually formatted with an ***ext2*** filesystem. As with a floppy disk, this filesystem will either become the root filesystem for the next phase of the boot process, or will supply a new, compressed filesystem which will be loaded into memory as a `ramdisk' (see below).

The diagram below shows the structure of a typical Linux bootable CD-ROM (but this isn't the only way to do it). The areas aren't to scale, of course: the volume descriptors, etc., are only one sector in length, but the filesystems will be many thousands of sectors. Notice that there is a complete ***ext2*** filesystem in the boot filesystem image, along with the boot sector. The boot sector will normally contain LILO code (see below). The filesystem contains the kernel and the initial ramdisk (see below), and the initial ramdisk in turn contains an ***ext2*** filesystem which will become the root filesystem.

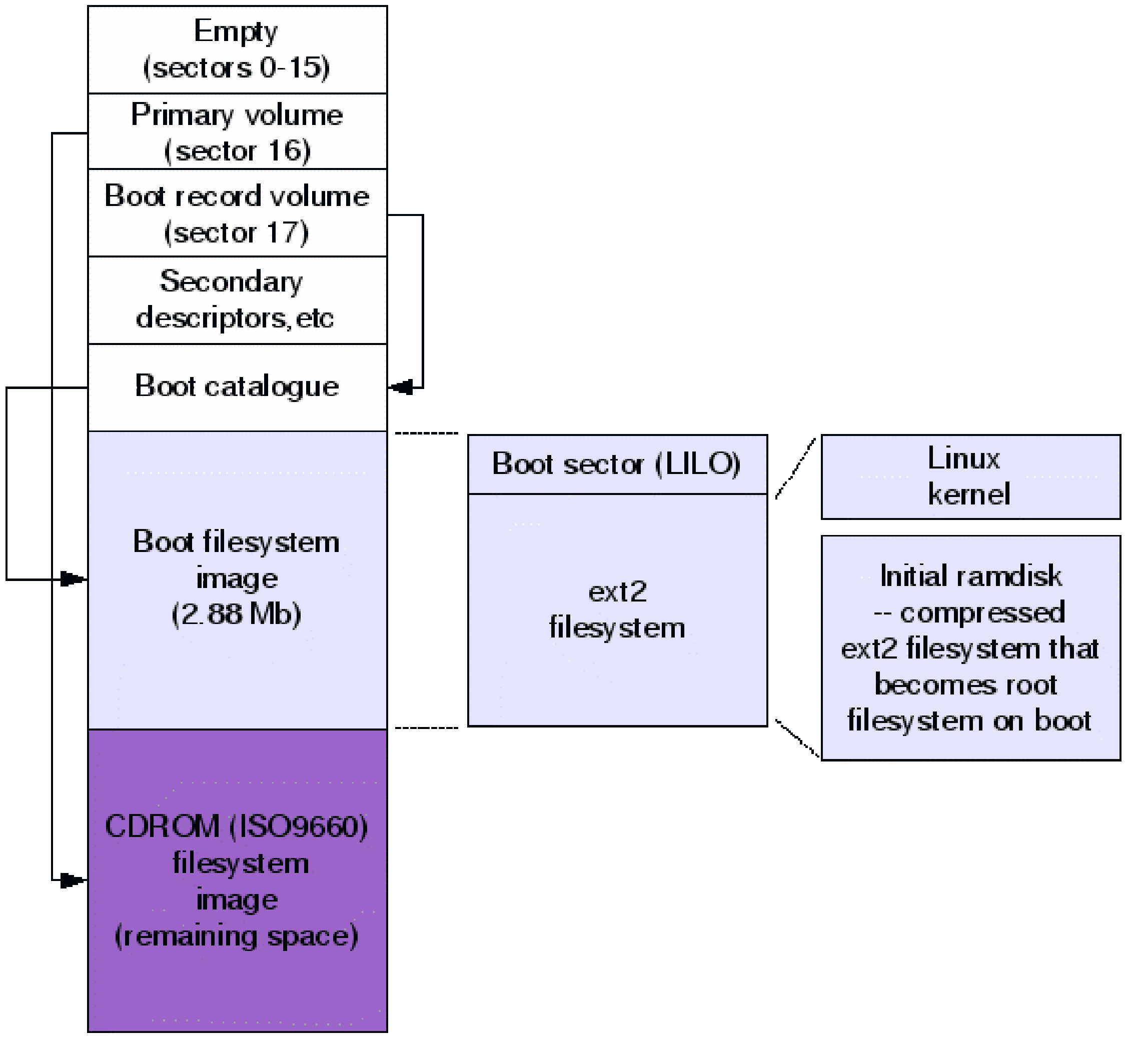

Empty (sectors 0-15)
Primary volume (sector 16)
Boot record volume (sector 17)
Secondary descriptors, etc
Boot catalogue
Boot filesystem image (2.88 Mb)
CDROM (ISO9660) filesystem image (remaining space)
Boot sector (LILO)
ext2 filesystem
Linux kernel
Initial ramdisk -- compressed ext2 filesystem that becomes root filesystem on boot

Figure 1. Structure of Linux bootable CD-ROM

Bootloader retrieved from network

The problem with booting from a network is that the functionality must be supplied in firmware, because if there is no hard disk, there is no practical place to load network-boot software from. Most PCs do not contain firmware this sophisticated, although some network adaptors have this functionality. Sparc-based workstations generally do have network boot functionality -- in the OpenBoot firmware, and it is quite comprehensive. Note that there is nothing to stop a PC getting a bootloader with network capabilities from, say, a hard disk or CDROM and then using this to complete the boot process over the network. However, this is not network booting in the sense I am describing here.

To get a bootloader via the network, the workstation must first of all decide where to get it from. This may be configurable at the firmware level or, more often, the workstation will issue a broadcast, and then select a boot server from the replies. Sun Sparc systems typically make a RARP request, broadcasting their hardware MAC address (`Ethernet address'). The reply from the server will contain the IP number assigned to the workstation, and that of the server itself. The workstation then uses the server's IP as the target for a TFTP download. Whether this download retrieves a network-aware bootloader, or a whole kernel, varies from one system to another. Some Sparc systems are able to TFTP a Linux kernel and load it, other require the retrieval of a network-aware bootloader which then retrieves the kernel (this is how Linux can be made to run on the Sun Javastation network appliance, which has somewhat stunted firmware).

Stage 2 (bootloader stage)

So we've got a bootloader into memory, from disk or network, and it can be executed. Its job will be to get the kernel into memory, again either from disk or network, and execute it. The bootloader will have to supply various vital pieces of information to the kernel, crucially the location of its root filesystem.

There are a number of bootloaders available for Linux: on the Intel/PC platform we have LILO and GRUB; on Sparc we have SILO. LILO is probably the best known, and has existed since the earliest days of Linux. SILO is essentially the Sparc port of LILO. GRUB is a much more sophisticated proposition.

LILO

LILO is a very rudimentary, single-stage bootloader. It has little or no knowledge of Linux, and does not understand the structure of any filesystem. Instead, it reads from the disk using BIOS calls, supplying numerical values for the locations on disk of the files it needs. Where does it get these values from? It has no way to figure them out at run-time, so the LILO installer has to supply them in the form of a `map' file. The LILO installer is a utility called lilo; this utility reads a configuration file and builds the map file from it. The location of the map file is then supplied to the boot sector that lilo installs.

The bootloading process with LILO thus looks something like this.

- The fireware loads the LILO bootsector and executes it

- LILO loads its map file using BIOS calls. Using the map file it finds the location of the boot message, which it displays to the console, followed by a prompt.

- The user selects which kernel to boot -- if there's more than one -- at the prompt

- LILO loads the kernel using BIOS calls, based on information in the map file it loaded earlier

- (optional) LILO loads the initial ramdisk (see below)

- LILO executes the kernel, indicating where it can find its root filesystem and (if necessary) initial ramdisk

A problem with LILO is that it can be quite tricky to use it for creating a boot sector for a system different to the one running the LILO installer (lilo). The LILO configuration file (usually /etc/lilo.conf) takes the names of files and devices as its inputs, but these names are never passed through to the boot sector being created. The files and devices referenced are simply analysed for their numerical offsets. For example, if lilo.conf contains the line

```
root=/dev/cdrom
```

and /dev/cdrom is a symbolic link to the real device file (perhaps /dev/hdc), it is important to understand that all lilo will store is the major and minor device identifiers of /dev/hdc. It is easy to imagine that if the bootable filesystem you are building contains a file called /dev/cdrom, and that is a link to, say, /dev/hdd, then the root filesystem will be found on /dev/hdd. But it won't; LILO does not understand filesystems, and the names in the configuration file are simply rendered down to device IDs and file sector locations.

GRUB

GRUB is a very different bootloader from LILO. It has a two-stage or three-stage operation, and has network boot capabilities (of course, the network boot facilities don't give you a way to get GRUB itself loaded: you'll still need network boot firmware).

The additional sophistication of GRUB means that it can't easily fit into a single boot sector. It therefore uses a multiple-stage process to load successively larger amounts into memory. In so doing it becomes able to understand filesystems, so the kernel itself, and the other files GRUB uses, can be specified dynamically at boot time; there is no need for explicit numerical maps such as the ones that LILO uses.

In brief, the GRUB boot process looks like this,

- Stage 1: the firmware loads the GRUB boot sector into memory. This is a standard (512 byte) boot sector and, thus far, the process is the same as for lilo. Encoded in the boot sector are the numerical disk block addresses of the sectors that make up the implementation of the next stage. GRUB then loads the blocks that are required for the next stage using BIOS calls.

- Stage 1.5 (this name reflects the fact that, strictly speaking, it is optional; its purpose is to load the code that recognizes real filesystems, and GRUB can be set to use numerical block offsets just like LILO): the code for stage 2 is loaded using BIOS calls, but with knowledge of the filesystem. Typically this code is in the file /boot/grub/stage2. On my system this program is about 120 kB in size; clearly we can offer far more sophisticated functionality in a program of this size than in the 5000-or-so bytes of LILO. The fact that GRUB loads its second stage as a *file*, and not as a list of disk sectors, is the key to its power; LILO can't do this, so you can't do much with it at boot time.

- Stage 2: GRUB puts up a menu of defined boot options, and exposes a command-line to the operator. The command line can be used to load arbitrary files as kernels and ramdisks (because stage 2 understands filesystems). Each boot option in the GRUB configuration file is expressed in terms of GRUB command-line operations.

- GRUB executes the commands entered by the operator, either from the configuration file or from the command line prompt. Typical commands are `kernel`, which loads a kernel into memory, `initrd`, which loads an initial ramdisk from a file, and `boot`.

The functionality offered by GRUB is quite similar to the OpenBoot firmware in Sun workstations, and includes the ability to retrieve kernels from a server using TFTP.

Multiple-boot machines

Because Linux was designed to be able to co-exist with other operating systems, the bootloader should be able to boot other operating systems on a hard disk as well as Linux. In practise this is relatively straightforward, as each of the other operating systems will have its own boot sector. All the Linux boot loader has to do is to locate the appropriate boot sector, and execute it. After that, the process will be under the control of the other system's bootloader. LILO, GRUB, and SILO all have this functionality.

Stage 3 (kernel stage)

By the time this stage begins, the bootloader will have loaded the kernel into memory, configured it with the location of its root filesystem, and loaded the initial ramdisk, if supplied. How we proceed from here depends to a large extent on whether we are using an initial ramdisk or not.

So why is an initial ramdisk such a big deal? Well, the concept arose from attempts to solve the problem of fitting a fully bootable Linux system onto a single floppy disk. The problem is that a Linux system that will boot as far as giving a shell, and offering a few basic utilities, needs about 8Mb -- far too much to fit onto a floppy. However, such a system will in practise compress down to about 2 Mb using gzip compression, so if the root filesystem could be compressed, we could get a working system in two standard floppies, or a single 2.88 Mb floppy.

Another problem that had to be solved was that of booting from a floppy disk and then mounting a root filesystem from a device other than an IDE drive. SCSI drives were particularly problematic: if the kernel was compiled to included all the necessary drivers, it would not fit onto a floppy disk. However, the initial ramdisk technique allows the drivers to be supplied as loadable modules, which can be compressed.

In outline, an initial ramdisk is a root filesystem that is unpacked from a compressed file. The boot loader will load the compressed version into memory, then the kernel uncompresses it and mounts it as the root filesystem. In this way we can get an 8 Mb root filesystem onto a 2.88 Mb file. Initial ramdisks are also useful on bootable CDROMs, because the bootable part of the CDROM is typically implemented as an `embedded' floppy disk.

Stage 3a (common kernel stage)

Whether or not we are using an initial ramdisk, the kernel will begin initializing itself and the hardware devices for which support is compiled in. The process will typically include the following steps.

- Detect the CPU and its speed, and calibrate the delay loop
- Initialize the display hardware
- Probe the PCI bus and build a table of attached peripherals and the resources they have been assigned
- Initialize the virtual memory management system, including the swapper `kswapd`
- Initialize all compiled-in peripheral drivers; these typically include drivers for IDE hard disks, serial ports, real-time clock, non-volatile RAM, and AGP bus. Other drivers may be compiled in, but it is increasingly common to compile as stand-alone modules those drivers that are not required during this stage of the boot process. Note that drivers must be compiled in if they are needed to support

the mounting of the root filesystem. If the root filesystem is an NFS share, for example, then drivers must be compiled in for NFS, TCP/IP, and low-level networking hardware

If we aren't using an initial ramdisk, then the next step is to mount the root filesystem. The kernel can then run the first true process from the root filesystem (strictly speaking, kswapd and its associates are not processes, they are kernel threads). Conventionally this process is /sbin/init, although the choice can be overridden by supplying the boot= parameter to the kernel at boot time. The init process runs with uid zero (i.e., as root) and will be the parent of all other processes.

Note that kswapd and the other kernel threads have process IDs but, even though they start before init, init still has process ID 1. This is to maintain the Unix convention that init is the first process.

Stage 3b (ramdisk kernel stage)

This stage is only relevant if we are using an initial ramdisk. In this case, the kernel won't involve init, but will proceed as follows.

- The kernel unpacks the compressed ramdisk into a normal, mountable ramdisk

- It then mounts the uncompressed ramdisk as a root filesystem. The original ramdisk memory is freed. It should be obvious that the kernel must have drivers compiled in to support whatever filesystem is in the ramdisk, as it won't be able to load any modules until the root filesystem is visible.

- The kernel then runs an initialization process. This process will, in general, not be the standard unix `init`, but a script that will mount the real root filesystem and then launch the next stage of the boot process. Conventionally this script is called `/linuxrc` but it can be specified to the kernel using the `init` parameter.

- `/linuxrc` does whatever it needs to, in order to make the real root filesystem available, probably including loading some modules. It then mounts the new root filesystem over the top of the ramdisk filesystem.

- Conventionally `/linuxrc` then spawns the `real' `init` process. It will typically do this using the `exec` command so that `init` ends up as process number 1, rather than 2.

/linuxrc need not mount a new root filesystem over the top of the ramdisk root, nor need it load init. These activities are simply conventions. For example, in order to boot a full Linux system from a CD-ROM, a workable proposition is to retain the initial ramdisk as the root filesystem, and have /linuxrc mount the CD-ROM at, say, /usr. This allows the root filesystem to be read-write; if we mounted the CDROM at /, the root filesystem would be read-only, and we would have to create a separate ramdisk and have a bunch of symbolic links from the CD-ROM to parts of that ramdisk.

Similarly, a `rescue' disk -- floppy or CD-ROM -- would probably not want to invoke init, but simply put up a root shell.

If we are using /linuxrc to prepare a root filesystem, it is a good idea to minimize the amount of initialization code in it. This is not because it won't work, but because the correct place for initialization is in the start-up script spawned by init. Doing initialization here, and not in /linuxrc enables us to ensure that the same initialization code is available whether or not an initial ramdisk is in use.

Stage 4 (init stage)

By now the kernel is loaded, memory management is running, some hardware is initialized, and the root filesystem is in place. All subsequent operations are invoked -- directly or indirectly by init. This process takes its instructions -- again by default -- from the file /etc/inittab. inittab specifies at least three important pieces of information.

- the `runlevel' to enter at startup
- a command to run to perform basic system initialization (conventionally this is `/etc/rc.sysinit`)
- the commands to run on entry to and exit from particular runlevels.

The order of operations is that the initialization command (rc.sysinit) is run first, then the runlevel scripts. The division of work between rc.sysinit and the runlevel scripts is entirely a convention. If you are building a custom Linux system you don't have to follow this convention. In fact, you don't even have to run init if it doesn't do what you need.

Stage 4a (rc.sysinit)

This script or executable is responsible for all the one-off initialization of the system. Linux distributions differ in the distribution of work between this script and the runlevel scripts but, in general, the following initialization steps are likely to be carried out here.

- Configure the system clock from the hardware clock
- Set up keymappings for the console(s)
- Mount the `/proc` filesystem
- Set up swap space (if there is any)
- Mount and check `local' (i.e., non-network) filesystems

- Run `depmod` to initialize the module dependency tree. This is important because it makes it possible for `modprobe` to work. The kernel's module auto-loader refers to modules by name, not by filename. It also expects that when it tries to load a module by name, any modules on which it depends can also be loaded by name. In a custom boot set-up, you may prefer to load all your modules by filename, and not compile in the auto-loader at all. This speeds the boot process considerably. However, you'll lose the flexibility of dynamically loading and unloading modules for hot-plug devices.

- Initialize and configure network interfaces. This step usually has to come after the `depmod` step or its equivalent, because the network drivers are likely to be loaded as modules.

- Load drivers for USB, PCMCIA, sound, etc. Again, these steps probably load or reference modules.

Stage 4b (runlevel scripts)

Let's assume that we will be entering runlevel 5 which, by convention, gives us a graphical login prompt under the X server. A typical inittab will have entries like this:

```
15:5:wait:/etc/rc.d/rc 5
```

The first line says that on entry to runlevel 5, invoke a script called rc, passing the argument `5'. The second line says that on entry to runlevel 5, run the script /etc/X11/prefdm -nodaemon. This latter script is somewhat beyond the scope of this article, being in the realm of X display management. In outline, prefdm is a script inserted by the RedHat installer. It contains code that will launch the X display manager selected by the user, either at install time or using a configuration utility. The reason it works this way is so that configuration utilities don't have to mess about with inittab, which is a bad file to mess up if you want your system to keep working. The X display manager will typically invoke the X server (i.e., the graphical display) on the local machine and give you a login prompt.

But back to the `real' boot process... The script rc runs the start scripts in a directory for the runlevel given in inittab. Usually, runlevel *N* will correspond to a directory /etc/rc.d/rcN.d. As we've decided to enter runlevel 5, the relevant directory is /etc/rc.d/rc5.d. This directory will contain a (possibly large) number of scripts with names beginning with `S' or `K' followed by two digits, e.g., S12syslog. The digits denote the order in which the scripts are executed: The `S' scripts are executed in ascending numerical order on entry to the runlevel (i.e., at boot), and the `K' scripts are executed in descending order on exit (usually at shutdown). rc passes the argument `start' to each script at startup, and `stop' and shutdown. As a result, we don't really need both `S' and `K' scripts, because we can use the argument to determine whether we are starting or stopping. Thus it is a convention on Linux systems that the K scripts are simply symbolic links to their corresponding S scripts, and the S scripts do both startup and shutdown operations.

So, for example, when entering runlevel 5, somewhere near the beginning of the rc process we will execute

S12syslog start

On shutdown, somewhere towards the end of the shutdown process we will do

K12syslog stop

which is, in fact, an invocation of

S12syslog stop

Inside the script S12syslog -- and most of the other scripts in that directory -- you will find both initialization and finalization code. So what do these scripts do? Well, this depends on the runlevel, and the distribution, and any customizations you have made. A typical set of operations will included the following:

- Apply firewall settings to IP network interfaces
- Bring up non-IP networking (e.g., IPX, appletalk)
- Start the system logger
- Start the NFS portmapper, lock daemon, etc., and mount any NFS shares specified in `/etc/fstab`.
- Start the power management daemon
- Initialize the auto-mounter
- Initialize the PCMCIA subsystem, loading drivers and daemons both for the PCMCIA hardware itself, and any cards that are currently inserted
- Start up the inet daemon (`inetd` or `xinetd`) which will take care of accepting incoming network connections
- Start the printer daemon
- Start `cron`
- Start the X font server

The very last step in the boot process will be to run a script S99local. This is the conventional place to put machine-specific initialization. It is considered bad manners to customize any of the initialization scripts that are supplied as part of a Linux distribution, simply because other people who may have to manage the system will have expectations about what is in them. Making arbitrary changes here will defeat these expectations. However, everybody expects to see machine-specific configuration in S99local.

Gotchas

It should be clear that the boot process on a fully-featured Linux system is fairly complex. You can simplify it a great deal if you are building a custom Linux system, or if you just want your machine to start up faster. However, there are a few things to watch out for when constructing a custom boot process.

- The various stages of the boot process are quite well separated, particularly the bootloader stage and the kernel stage. What does this mean? Well, imagine a situation in which we download a network bootloader, which loads a kernel from a file server. The kernel then starts up and wants to initialize its network settings (IP number, etc). Now, the machine had to have an IP number during the bootloader stage, didn't it? Otherwise, how would it have been able to do network operations to fetch the kernel? So one might expect that the kernel could simply get its IP number from the bootloader. The problem is that Linux bootloaders don't know how to supply this information to the kernel. Why should they? The bootloader designers can't anticipate everything that the kernel might need to know in advance. Therefore the kernel must then have the machine find its IP number, etc., *again*, independently of what the bootloader may have done. In practise, it's probably going to get the same IP number, but that makes no difference. This causes problems for people who want to build a fully-diskless installations (like the Javastation example elsewhere on this site). Your network-boot firmware probably uses RARP or DHCP to find the machine's network settings, but that doesn't mean that you don't need to include the same support in the kernel when you build it: the kernel will have to do it again. When you come to mounting the root filesystem as an NFS mount, you need to make sure that you have a way to tell the kernel where the NFS server is (usually via kernel command-line parameters, but on some dumb systems you have to hard code them into the kernel before compilation). The kernel has no way to know whether the machine that replied to the RARP or DHCP request is going to be the one to supply the root filesystem.

- Another problem, which appears different but is in fact identical, is that of booting from SCSI devices. So, you have a PC or workstation that can boot from a SCSI CD-ROM drive. The firmware loads the boot sector, which initializes the bootloader, which loads the kernel. So far so good. Then the kernel takes over. It tries to mount the SCSI CD-ROM as a filesystem, but fails. Why? Because SCSI drivers aren't included in the kernel. It is wrong to believe that, because the system can read from a CD-ROM during boot, that the kernel will be able to read from the CD-ROM. The kernel won't use BIOS calls to read the CD-ROM, which is what the bootloader will probably do. The kernel will use the standard Linux VFS (virtual filesystem) infrastructure, which will communicate with the SCSI infrastructure, which will communicate with the low-level SCSI device driver, which will communicate with the hardware. To boot a kernel from a SCSI CD-

ROM, you need to make sure that all of these components are available to the kernel (in the form of modules), or are compiled in.

Some comments

BIOS (The Basic Input/Output System) is the lowest level interface between the computer and peripherals. The BIOS performs integrity checks on memory and seeks instructions on the Master Boor Record (MBR) on the floppy drive or hard drive.

The MBR points to the boot loader (GRUB or LILO: Linux boot loader). Boot loader (GRUB or LILO) will then ask for the OS label which will identify which kernel to run and where it is located (hard drive and partition specified). The installation process requires to creation/identification of partitions and where to install the OS. GRUB/LILO is also configured during this process. The boot loader then loads the Linux operating system.

The first thing the kernel does is to execute init program. Init is the root/parent of all processes executing on Linux. The first processes that init starts is a script /etc/rc.d/rc.sysinit. Based on the appropriate run-level, scripts are executed to start various processes to run the system and make it functional.

A few basic facts on how the (IBM compatible) PC boots itself.

a. The first thing your processor does when powered on, is to seek a standard address in memory, and execute whatever code it finds. This address is F000:FFF0h. All x86 Intel or clone chips are hard-wired to seek this address.

b. At memory address F000:FFF0h, there will be a jmp (jump) instruction to another address where a program is located. This program (in the BIOS), hard wired in ROM somewhere between 640k and 1 meg in memory will perform rudimentary hardware testing called Power On Self Test (POST).

c. BIOS will seek boot drive. (the one given priority in your setup (CMOS)). BIOS will get the first sector of this drive (cylinder 0, head 0, sector 1), load this sector into RAM and execute the code found therein. It is important to remember that this code will be executed, no matter what it says to do. If it is virus code, it will execute, because there is no antivirus software running yet. The OS hasn't loaded.

d. The first sector contains two items. First is the boot loader. The boot loader is a short program that seeks the bootable partition, and loads the first sector of the partition into memory, and jumps there to execute its code. Second, the first sector also has four data fields for partition info. This is the Partition Table. So, this first sector is sometimes called the Master Boot Record, and sometimes it is called the Partition Table, because it contains both of these items.

e. In the standard MSDOS MBR/boot loader, the code will consult the partition table, looking for the "bootable", or "active" flag on one of the partition entries.

The first sector of the bootable partition will be loaded into memory and executed. This sector is called the Boot Sector, and is often confused with the Master Boot Record, discussed above. It is also, especially in the past, a common place for virus code.

f. The code in the boot sector looks for the OS kernel and loads it into memory and executes it. Originally, the dos kernel had to occupy the first contiguous data sectors of the partition, because the boot sector code wasn't very sophisticated.

If you install a different OS than the dos/win type, or if you dual boot, the standard boot loader in the master boot record can be replaced by a boot loader that can give you a choice. The standard Dos boot loader, can only boot the partition marked bootable, but LILO, the Linux loader ignores the bootable flag, and load whatever OS you select.

Preboot Execution Environment

The Preboot Execution Environment (PXE) is an environment to bootstrap computers using a network interface card independently of available data storage devices (like hard disks) or installed operating systems.

The Preboot Execution Environment (PXE) is an industry standard client/server interface that allows networked computers that are not yet loaded with an operating system to be configured and booted remotely by an administrator. The PXE code is typically delivered with a new computer on a read-only memory chip or boot disk that allows the computer (a client) to communicate with the network server so that the client machine can be remotely configured and its operating system can be remotely booted. PXE provides three things:

1. The Dynamic Host Configuration Protocol (DHCP), which allows the client to receive an IP address to gain access to the network servers.

2. A set of application program interfaces (API) that are used by the client's Basic Input/Output Operating System (BIOS) or a downloaded Network Bootstrap Program (NBP) that automates the booting of the operating system and other configuration steps.

3. A standard method of initializing the PXE code in the PXE ROM chip or boot disk.

The PXE process consists of the client notifying the server that it uses PXE. If the server uses PXE, it sends the client a list of boot servers that contain the operating systems available. The client finds the boot server it needs and receives the name of the file to download. The client then downloads the file using Trivial File Transfer Protocol (Trivia File Transfer Protocol) and executes it, which loads the operating system. If a client is equipped with PXE and the server is not, the server ignores the PXE code preventing disruption in the DHCP and Bootstrap Protocol (BP) operations.

The advantages of using PXE include:

1. The client machine does not necessarily need an operating system or even a hard disk.

2. The client machine can be rebooted in the event of hardware or software failure. This allows the administrator to diagnose and perhaps fix the problem.

3. Since PXE is vendor-independent, new types of computers can easily be added to the network.

Post booting of different OS

In this part we will give a comparison between the post-booting processes of three Operating Systems: Mac OS X, FreeBSD and Gentoo Linux. We will not describe the entire booting process, but limit the explanation to the last stage: getting the system up and running after the kernel has been loaded.

After init is started, almost everything is different between FreeBSD, Gentoo Linux and Mac OS X. One of the biggest differences is Gentoo using explicit runlevels, whereas the other two almost only have single and multi-user mode. This is one of the differences between SystemV and BSD style init.

Another notable difference is the way the rc scripts are stored on disk and the way they are run. FreeBSD has a configuration file containing knobs which are read by the rc script to determine the actions to take. Mac OS X has startup items which are all run, Gentoo has per-runlevel symbolic links to the rc scripts which will run. FreeBSD has a static repository of rc scripts, Gentoo and Mac OS X have a dynamic repository.

Another difference is the configuration of init itself. On Gentoo, this is done using the inittab file, containing the value of the default runlevel, the scripts to run at each runlevel and the tty-configuration. On FreeBSD and Mac OS X, there's no inittab, the system simply boots into multi-user mode unless otherwise specified in the bootloader and the tty-configuration is done using "/etc/ttys".

Network boot technology

Common booting devices are hard disk, CDROM and floppy diskette. All PC BIOS support them. Latest BIOS will allow USB device and network interface card as the primary booting device. Using the network interface card for booting is known as network boot. Different technologies are proposed on this area. EtherBoot creates boot strap program image, which is burned into ROM/EPROM to plug on socket of Ethernet network interface. A customized boot image is needed to prepare by the supplied utilities program in order for EtherBoot to load and execute. This boot image may be the combination of kernel and initrd or simply the Linux kernel itself. Netboot is another approach. Differently it uses standard DOS drivers (either Packet or NDIS-2) rather need to develop the network interface driver in the case likes EtherBoot. For similarity, the "netbootable" image is needed to prepare with the utility program with the NetBoot package. Another approach is PXE (Preboot Execution Environment). It is new standard proposed by Intel for Intel Wired for Management. As PXE is widely accepted and supported, it is network technology used in SLIM.

PXE provides varies of network connection to servers prior to loading an OS. A number of standard IP protocols such as DHCP and TFTP are support in PXE network connection. Almost latest PC system integrated with NIC in market supports PXE. For Linux, the common boot loader use with PXE is pxelinux, which is a variant of syslinux. Differently to Netboot, EtherBoot, pxelinux does not have initial boot strap code that is supposed to be part of PXE network code. The pxelinux simply communicates with PXE boot code.

Typical network boot sequence is simple. Firstly codes (PXE, Netboot, EtherBoot) in ROM of network interface contacts BOOTP/DHCP server to allocate an IP address. Then it obtains the location of the boot image. Next it uses TFTP to download the boot image, and finally jump to execution point of the boot image.

Understanding and insight into authentication

With the rapidly increasing use of networks of computers for the transfer of highly sensitive data or message such as commands for controlling operations of deadly weapons, commercial electronic fund transfer, electronic business transactions etc., the need for establishing a reasonable level of assurance of authenticity of a received message, before executing it, is becoming of paramount importance. Furthermore, if a received message can be demonstrate to be authentic, it can be conjectured that the message has not been modified en route i.e., it has reached the destination intact.

Introducing authentication

Authentication is the act of validating the unique user before allowing access to a computer or computing resource [9].

The word "authenticate" is derived from the Latin ***authenticus*** which means to make valid and effective by proof. Authentication is the action or process for verification of genuineness of a document or identity of a person or process by using a unique attribute assigned to it in order to check the legitimacy of entrance (i.e., access control) to a protected resource. So authentication may refer to a message, a person or a process. In a computer system or computer communication network, authentication is an important part of data security.

If authentication refers to a message then it means to ensure that the message is genuine, that is, it has arrived exactly as it was sent and has come from such a source which has the authority to send it. If authentication refers to a person then it means verifying the identity of that individual in a reliable, unforgeable and co-operative manner.

The authentication problem is similar to a "pattern recognition" problem from the point of view that recognizing an object needs a supply of a sufficient number of points (identifiers) of different intensity level and this is what occurs during an authentication process based on matching of finger prints or hand geometry.

For identifying a user, publicly known information regarding that user (e.g., user name, surname, date of birth etc.) can be used whereas for authenticating the user some time-variant and/or time-invariant attributes, which are unique to that user, can be used. The unique (i.e., individual) characteristics of a use may include such terms as secret (privately selected) passwords (assumed to be time-variant) or personal physical characteristics e.g., fingerprint, voiceprint, keyboard tying speed etc. (assumed to be invariant over a certain period of time).

The term ***authentication*** refers to either the content of a message or the identity of an individual. When it refers to a message, it means to ensure that the message is genuine i.e., it has arrived exactly as it was sent and originated at the stated source. And when it refers to an individual, it means verifying the identity of that individual.

The ***environment*** where authentication takes place can be either mutually suspicious or one-way. In case of mutually suspicious environment, none of the communicants are assumed to be trustworthy and hence both sender and receiver have to supply sufficient identifiers/authenticator to each other. In case of one-way authentication, one of the communicants is assumed to be trustworthy and the other authenticates itself to the trustee. The latter type of environment is most common e.g., in controlling users' access to protected resources in computer systems.

An ***Authenticator*** is a distinguishable attribute of the message which enables the receiver to detect any alterations made to the message prior to reception.

Authentication Server is the unit which can check the access rights of a user to a protected resource. It can also serve as supplier of session keys to the users for secure communication.

A ***Protocol*** is defined to be a prescribed sequence of rules and procedures for interaction between/among two or more entities in a communication system.

Digital Signature is a property, private to a user or a process that is used for signing messages. The receiver of the message can check the authenticity of the received message and its sender by checking the validity of the signature.

Why authenticating users is important?

Actually many enterprise or public networks are projected and built to work on-line with the WWW: that is because the spreading of e-commerce and web technologies among the people, due to technology itself improving and costs decreasing, has led to a subsequent stronger use of IT medias to purport many daily operations previously performed on-site.

Obviously the greater and greater mass of sensible data published on the web or stored on a corporate database has to be protected properly: once operations took place on site and the authentication (recognition we'd better say) of the user was easy; now the service-provider and the service-user (commonly addressed as server and client in web-language) interact never seeing or meeting each other and the problem of trustful, reciprocal recognition is quite huge: privacy and security concerns are strong both for incorporates (whose private data are related to their business activities) and PA (whose sensible data implies strong privacy concerns for the citizens they represent and serve).

Moreover for a big company with many employees the control of their rights over certain data is a strict necessity: easily guessable not all the research-lab database should be browsable through the web nor possibly accessible from all insiders, but it's going to be used and administrated only by few authorized users who must be able to prove doubtlessly their rights to the system daemon before they could interact with it.

Authentication vs. Identification

All of the previous security concerns are authentication problems and they make us aware why authentication is important: it is the process of uniquely associating users to their rights and capabilities in a trustful way developing all possible protocols and mechanisms to determine if the identity they claim is their own one. But through years and technology improving misunderstanding the process of identification with the authentication one is almost common sense: despite this in the subsequent pages we would like to focus on authentication and the next paragraph could help the reader determining the topic of the paper.

Namely the process of identification is a one-to-many match test while authentication is a one-to-one comparison. Even though the perception retrieved by the user could be likely the same (submitting some credentials) these processes involve huge differences referring implementation, protocols, and performances. Authentication could be addressed as an identification subroutine and in some cases this could be real but this is not always true. Obviously the identification processes involve database's queries in a larger measure but the main difference is due to the level of reliability the process must grant to the system administrator. Users to be identified do not claim their identity: the system searches, among a determined group of profiles, the one matching with these credentials and optionally performs authentication, submitting whatever else request to the user. When a user tries to authenticate on a system he will submit two factors: an identity (coded as a username or wired on a token for example) and then some credentials to prove he is the one he pretends to be. The security focus is greater in the second scheme because of the two factors process and due to the fact that a check on determined data could take place in a more accurate way then a long search (resulting not time-consuming) and because this process involves a larger user-training interacting with the system. Of course identification is generally only one part of an embedded security system operating on a workstation or a system: in the following paragraph a description of this subsystem is briefly undertaken.

Authentication, Authorization and Accounting

Authentication is only one part of the security functions usually set up on systems. The problem of trustful recognition during a e-communication is the first step of any act a user, on behalf of some credentials, can perform within a system. As well as in social interaction the principals shall recognize each other (Authentication), subsequently they would act according their confidence degree or their business relationship (Authorization) and then they should constantly check out how their meeting is evolving and remember what they have been speaking about (Accounting).

So AAA_is a term for a system to smartly controlling access to computer resources, enforcing policies, auditing usages and providing any relevant information concerning users' actions.

- Authentication

An authentication process provides the techniques to certainly identify a user, granting him access only once he has been able to exhibit his unique set of criteria which he previously gained, according a secure protocol, from a principal of the communication.

- Authorization

After users get logged into a system we have to state what type of actions, activities and services the user trustfully authenticated has right to perform. This is the process of enforcing policies within the system providing user the rights they possess according his authenticated identity. This technique allows smarter administration using multiple group-profiles for example.

- Accounting.

A set of statistic controls focuses on the amount of system time, data and resources a user is sharing. Through processing these information the system is able to plan and optimize its use, as well as to protect itself from atypical profiles: auditing authenticated users' profiles allows to verify abnormal usage.

The best way to integrate AAA process depends on how authentication is performed, and on the specific implementation, that is the object of next paragraphs: easy guessable no security could be achieved when system's user is not safely and certainly identified.

Techniques, factors and protocols to achieve authentication

In this part we try to evaluate what kind of peculiarities an authentication scheme should have and how we could achieve these performances to be pursued efficiently. Starting from a list of desirable topics we're then going to determine how we can achieve authentication and what kind of authentication.

The goals for an authentication scheme

We would like to focus our attention on what kind of requirements, utopistic or not, an authentication process should satisfy in order to accomplish as smartly as possible its tasks:

- Users should be registered only once, no matter how many system they may need to use in order to perform their function. Supposing A-company owns five servers and the employee user has to work with all of them to pursue his task: he will be registered only once and a mechanism of trust-chain should be developed through the servers to allow him to be recognized equally by and authenticated on any of those.

- The process should be conducted promptly and efficiently: there's no advantage in getting money from an ATM if the authentication protocols take half an hour to let you interact with your bank's server.

- Once they are registered users should have to log-on the system, submitting their credentials, interacting with a user-friendly daemon. Both for ease of use and user-guidance the authentication subsystem should be designed with a framework dedicated to user-acceptance and assistance. For example users should authenticate only once per session and have their identity confirmed as long as that session lasts. A system that asks you your credentials every time you turn to the next page on the same web site would be useless and time-consuming, as well as a system pretending users to remember 64 digits alphanumeric passwords.

- The authentication process has to be effectively able to determine whether prospective users are registered and they really are whom they say to be and not, for example, the wife of an employee trying to get access to the enterprise database to check her husband's dater.

- The process shouldn't be time or cost consuming above the company enforcing policies: none would realize an authentication system that guarantees absolute perfection in user-authentication but that needs 15 minutes for each user to be granted access and many additional expensive hardware infrastructures to accomplish that.

- Soundness: the system should grant, according to the site which is built, consequent soundness and provide adequate protection to the authentication devices. Supposing a strong authentication system is implemented to gain access from the street, through the front door of the directive building, in a company establishment, this system has to be rain-proof, tamper-resistant, temperature independent and tuned to perform authentication even until a certain dirt-degree of the devices.

- Security: an authentication system is part of a security system and should grant some guarantees. Supposing the authentication system protects a high confidence archive that is often accessed by many users the availability of the system should be nearly 24/24 unless we accept to deny all access to the data (or grant anyone reading rights) during the periods when the authentication system is down. Supposing the authentication system eventually crashes down more than a reserve method should be implemented to protect the private data as well as not to lock the productive process of the company. Even the integrity of the data managed by the authentication system should never be violated by the system itself unless specific policies of suspension or enrollment are implemented. An authentication system is to be able to provide secure recognition under many circumstances providing recovery protocols even under particularly hard circumstances.

Every resource and every environment requests different authentication protocols: there's not one authentication methodology that fits all kind of data and any kind of system user so that special care should be taken in projecting a determined identification system customizing protocols according to their future use and requested strength.

Classifications of user-authentication

These topics should be differently stressed whenever we distinguish between local and remote authentication whether we talk about static, rigid or continuous authentication. We can easily pick it up: the strength of the protocols used depends on what we are defending and from whom.

Local vs. remote

In a dedicated and protected environment authentication will take place on trusted networks by authorized users to give them access to those resources they can interact with. In this situation you would probably perform physical access to the structure where hosts stand and moreover every authentication session takes place on a machine that you consider trustworthy because it's held under your administrative control. This situation seems (we should always remember that the common idea that evil is out of our enterprise is quite wrong: the 90% of attacks to our IT systems is taken by insiders!) to be much more friendly than the classic remote authentication scheme where you can't trust the link between your host and the other one: that could be a tampered, totally unsecured channel or a network monitored by an eavesdropper trying to steal your credentials to fraudulently use them for his purposes. Outside your organization's area you can't perform any kind of physical access control so that definitely you don't know who's sitting in front of the host contacting you : this is possible within your company where this could be achieved but providing a large amount of money (might be even discarding user-friendliness from your employees' perception, forcing them to many controls and procedures to let them reach their workstation): so the authentication session is exposed to many kind of hacking attacks able to fool your system. This scenario presents many more risks than an intranet or local authentication scheme and still represents a great challenge for security experts to succeed in uniquely and securely granting access only to authorized users. Software devices and protocols should be designed to discard the flaws of the following dangerous situations:

- Communication privacy: according the fact that the communication takes place on third party's channels and lines that we cannot control, the parts of an authentication session could be eavesdropped and recorded for further fraudulent uses and we would have no perception of this.

- Authenticator's authentication: if we authenticate outside of our intranet towards a server we will be obliged to trust that server, submitting it our credentials, and we will authenticate on that server but who's going to authenticate that server towards us and make us aware that is not a fake application trying to steal our profile? The problem is quite huge.

- Authentication channels and services availability: supposing we refer to a third party to authenticate towards some services we use for our business, once the authentication server is down will the service-company allow us to interact with its services or will those services be denied? This could be a great challenge if we consider health-care problems and hospitals' computer networks. Inside our company we could be able to provide many recovery protocols but outside our control everything could happen that would stop the service-availability or that could interrupt the authentication channel.

Static, rigid or continuous authentication

These adjectives don't define three different techniques but three different ways to authenticate users.

In a Static authentication scheme once the user has been certainly recognized the subsequent communication between client and server takes place in clear, without any further control being performed until the session comes to an end.

A rigid authentication scheme provides authentication along a secure channel that allows performing the communication session between the two principals: this technique guarantees a secure channel to exchange sensible data throw it. This channel is kept alive until the session ends.

A Continuous authentication scheme establishes a memory-less secure channel: for any single part of the communication the principals have to prove each other their identity again: this scheme may result slow but affords even stronger security than the previous ones. Practically both client and server never trust each other without a previous evidence of identification had been shown by the other principal: this happens every single time they just exchange one message.

Of course this different typology of authentication implies different protocols to be developed and needs different media to have these ones implemented.

On-line and Off-line authentication

Due to many different protocols implemented through year's authentication could be achieved on-line or off-line. This could seem a useless definition but providing the differences between using web procedures and relying on third parties static factors implies huge differences. In fact, according to the different status of the protocol we could face different kind of dangers: the interaction with live principals allows us to better discard any kind of attack that use old credentials (such as replay attacks) but expose our workstation to all the kind of attacks that are performed live such as man in the middle or interleaving ones. On the contrary off-line authentication does avoid all the problems related to networks' sniffing, eavesdropping and tampering over unsecured channels else making the system more prone to any kind of attacks that relies on no third party live interaction.

User or workstation authentication

Careful readers may have noticed that no emphasis has been posed on user whether workstation authentication but this is a huge problem. Every single protocol that will follow is to be observed through this lens in order to understand whether we are authenticating a user or his workstation. We are used to think about a user and his workstation as one thing but in large environments even a quick coffee-break could result in a dramatically serious threaten. Once a user has successfully authenticated himself for the valuable data stored on a workstation or collected on a server opened working session we should find a way to prove that the user actually accessing them is the same that initiated the transaction; moreover this is to be done without discarding user-friendliness or privacy perception of the user itself. Despite this there's no larger chance, out of continuous security system (a memory-less one) to safely achieve that: the use of screen locks or the proximity token technology (using devices that unlock workstations only when the authorized users is in the nearby of the machine) might enforce some good defense but most of this attacks success hopes relies on user laziness: once a user felt safe and has left his workstation without locking it the threaten grows strongly.

Different credentials to achieve authentication

After a first review of authentication topics has been developed we should now explain how we could practically grant these features that have been described previously. Particularly we should stress the topic: "With what kind of credentials a user can authenticate himself on a system?"

That's because many commercial authentication protocols are different according the factor they choose to recognize the users, usually these categories are addressed as:

- what you know:

Something that you know and that is supposed to be kept secret or not to be shared with others such as a Personal Identification Number (PIN) or a password;

- what you have:

Something like a key that we solely possess: a token or a smart card for example;

- what you are:

A biometric feature that belongs to us: usually divided in behavioral or physical biometrics such as sign verification and keystroke rhythm or thumbprints, our iris shape and colours, our ear shape and whatever else can be measured from our body (even DNA code!).

Each of these factors implies different hardware support and different protocols to achieve the same goal: doubtlessly authenticate users on an IT system. We will now develop a quite wide review of possible solutions according these three different kinds of authentication reporting, when it's interesting, examples of working protocols. For each of these protocols we'll focus on topics, caveats and advantages introduced.

Using a single factor in flat authentication schemes

For each kind of single factor in the next chapter we will report a review of used protocols evaluating advantages and disadvantages, examples, security features and ease of usage. Moreover we're going to briefly talk about costs, set up problems and user acceptance because these topics assume relevant importance on the market-place. Before reporting this we would like to stress much on the three categories we have just exposed, trying to understand the nature of each and the inner capabilities it refers. Then we will see how these skills are combined in actual products.

- Something you know

Protocols using this genre of factor had been the first to be implemented on many systems and are still the most used ones over the WWW. Actually is a really mature field of work and, due to the exclusive use in the past years, is practically integrated in most of the software available on the market. Even though they take advantage of user-acceptance over the average (the IT market was born and bred with them) these systems no longer provide an appropriate degree of protection and their use is going to be dedicated to low-impact protection environment (such as unlocking the pin of a mobile phone) or as a step during a layered or multi-factor authentication schemes.

- Something you have

The second factor that we can use to authenticate ourselves on a system is something we have and others have not. Tokens as well as keys represent the best example of objects of this class. Devices of this kind have always been used through the years for many different purposes and for protection of private properties: actually they've been used to secure IT systems since the end of the 70ies.

- Something you are

Before we could talk about biometric authentication we'd better stress on biometric itself: this topic is not yet so common and many misunderstandings could confuse the reader of this report. So we will briefly focus on this resource, its many actual uses and its possible future many other ones. In fact biometrics it's not a new but it's still today a quite obscure field for the majority of the people and divulgating its strengths as well as its weaknesses it's a determined scientific goal.

What biometrics are and how they are used

Biometric is a measure of a person's unique biological feature such as a fingerprint or voice: this measure can be stored in somewhat a template that a third party, that you have previously authorized, is going to use for verifying your identity granting you access to any sensible resource it wants to share with you. Biometrics represents the smartest way to prove your identity (we would say the common way in everyday life.) and basically the most natural; you have nothing to remember or to forget: you have only to be yourself.

For example having your face scanned by an opportune device could allow you to enter a restricted-area in your bank or check-in in an airport: no queues, no stops, and no questions.

Therefore there are many biometrics and not all of them might be well accepted because of their intrusiveness and the high-collaboration degree required from the user.

That's why we have to mind about biometrics' real potentialities: bringing a huge improvement in ease of use and tolerance between users is as important as trading none of the security tips actually available on the market as well as keeping operations' average times close to the actual ones.

Biometrics is actually used mostly for identify or verify users: the logical application field is physical access to restricted areas. Biometrics had been used for this purpose since 80ies to build unmanned access controls. Probably they represent the ideal solution for 24 hours high-volume access control. In the future they could help authorities in implementing covert surveillance systems such as automatic check-in for airports and many other types of scenery.

Granting stronger security through multi factor and layered authentication schemes

As declared previously, the authentication process may be required either between two individuals through processes or between a process and a set of users. Therefore, different methods exist for the authentication depending on the environment in which it is going to take place. The constraints to be considered for selecting an authentication scheme are those which indicate that the scheme is not tedious, time consuming, too complex and frustrating to users.

The two major classes of authentication schemes are,

- Protocol based authentication: This type of scheme is applicable especially for on-line or live authentication i.e., the user authenticates himself to the system for the purpose of logging in or one user is authenticating himself to another user for the purpose of exchange or disclosure of secret information. This class includes handshaking (i.e., request-response) protocols and challenge-response protocols. The request-response method consists of just answering the random but relevant questions, in order to supply time invariant identifiers and time variant/invariant authenticators to the authentication server, of the system and the challenge-response mechanism needs the answer to be changed in a pre-determined manner before sending.

- Encryption based authentication: This type of scheme is applicable for both association bounded and unbounded (i.e., datagram services) authentication schemes. The only criterion is secure transmission of the secret key to the proper recipient. The main application of encryption based authentication is for establishing authentic interactive communication between two users in computer network environment.

We usually authenticate ourselves many times a day into our workstations giving proof of who we claim to be. Whatever method we use for each purpose usually only one method is used: in a layered authentication scheme this is no longer true. The aim is to provide more dimensions to an otherwise flat security model introducing many different authentication steps to be undertaken in different situations and for different system resources. For example every generic employee of X company has the same software installed on his/her workstation and to have access to it he ought to be able to digit his own private password. To access the research lab folders' tier and related applications, virtually available from all workstations, he has to insert his own USB token into a port of the workstation. If now he wants to remove, modify or edit one of this files he will have to be strongly authenticated through a retinal scan process, proving the system his writing rights over these folders. This plausible scenario addresses a classic example of a layered authentication scheme. Each different layer of the system software is protected, according to data sensibility, with a greater effort by the system. This is what is called a layered authentication system and such a scheme could be adopted for two main reasons:

- User friendliness: if a super user, who has full rights even on the research files, is working on his workstation in order to browse some white papers on the Web he doesn't need to insert his token and face a retina-scan.

- Stronger security: if a third malicious person wants to harm the company's valuable know how he should know a password, possess the related token, and manage to fool the retinal-scan system: moreover these factors should be stolen all from the same employee and this make it harder to succeed. There is no need to stop at three distinct levels and administrators could choose to implement some tiers they want according to the data protection requirements. Another important point to make is that you may grant authentication from one level to the next by an entirely different form (smart card to password to biometric) or you may choose to use different formats of similar devices between the levels (such as fingerprint and then voice recognition after an iris scanning).

Of course any layered authentication system is a multifactor authentication system (at least from the second tier on) but this does not imply that every multifactor system is a layered one at all. In fact it is possible that your bank provides you, in order to access its on-line financial consultant offices, a pin and a token to be unlocked with. Anytime you enter your bank web-site if you want to talk with your consultant you have to type in your user ID, insert your token and digit your pin to gain access to the reserved section of the site. This is a multifactor authentication scheme but not a layered authentication scheme and it addresses all situations when a stronger authentication is required according to the second point in layered systems: it's harder to steal more than a credential from a user. In fact multifactor authentication is common practice (as well as layered one) in high security environments where every effort is produced to keep sensible data secured.

Authentication technologies

In general, there are three factors that might be used in an authentication system [2]:

- something a user knows (a password or Personal Identification Number (PIN));
- something a user has (a device such as a smart card or token);
- something a user is (biometrics).

Traditionally, passwords and PINs have been used as the most common authentication factor. The benefits and drawbacks of a number of technologies addressing each of the above factors are discussed in this section:

- Passwords;
- Smart Cards and Tokens;
- USB Authentication Tokens;
- Number Generation Tokens;
- Biometrics.

An authentication system will often use more than one of these factors. A combination of factors can be used either to strengthen the authentication mechanism as a whole or to retain the strength whilst reducing the reliance on a single factor. Thus, for example, the use of shorter passwords could be accepted whilst retaining the security of the system through the use of a second factor, such as a smart card.

The use of a password or PIN may be considered suitable as the sole method of authentication. Otherwise, the use of an authentication token r a biometric system may be considered as a second factor of authentication. For highly sensitive systems, it may be necessary to employ three-factor authentication.

Passwords

Using ‘something that is known’ to authenticate an individual usually means requiring them to demonstrate knowledge of a password. PINs may be viewed as a specific subset of passwords; they are passwords that are comprised of numeric characters only. The idea of authentication-bypassword is a simple one. The individual lays claim to a particular identity, often represented by a username, and then supports this claim by demonstrating knowledge of some ‘secret’ information known only to that individual and the authenticator.

Technical Maturity and Effectiveness

Passwords have been built into systems and used effectively as the sole authentication mechanism for many years. It is a technically mature technology, although passwords suffer from two conflicting requirements: the passwords must be sufficiently ‘random’ to prevent them being guessed by an attacker, yet simultaneously not too difficult for the user to remember. The security of a password-based authentication system relies on achieving the right balance between the two.

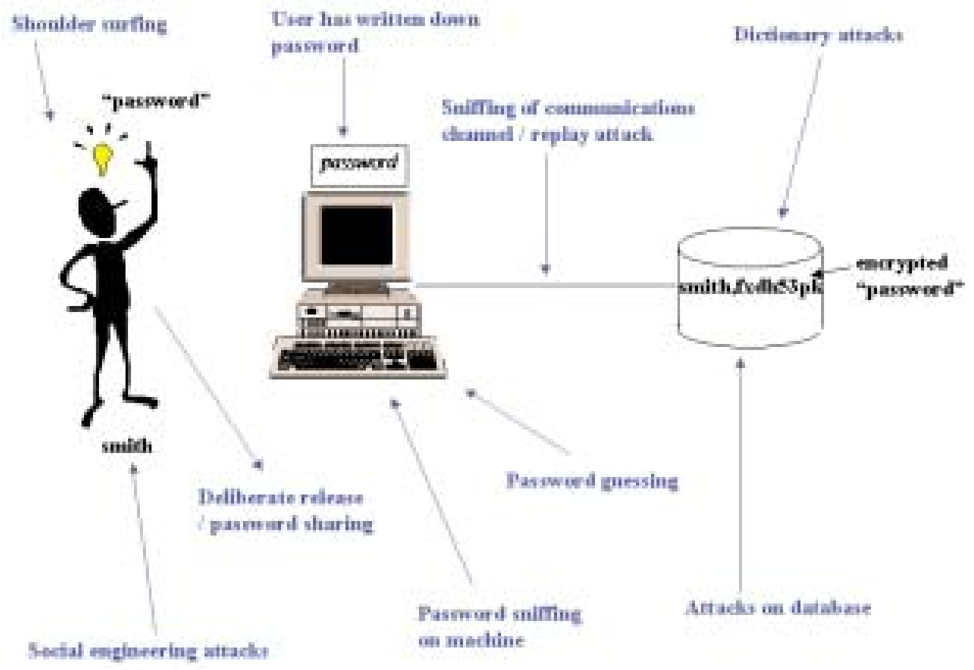

Shoulder surfing
User has written down password
Dictionary attacks
"password"
password
Sniffing of communications channel / replay attack
encrypted "password"
smith,fo8h53pk
smith
Deliberate release / password sharing
Password guessing
Password sniffing on machine
Attacks on database
Social engineering attacks

Figure 2.Vulnerabilities of a password system

User Impact

As an authentication mechanism, passwords have several benefits over other authentication systems. Passwords are commonly used for logging-on to computers and most popular operating systems have password authentication built in. This can make passwords the easiest option when choosing an authentication mechanism. Also, the majority of users are likely to understand how to use passwords and to find them acceptable.

Threats

As Figure 2 shows, there are many methods that can be used to attack a password-based authentication system. Complex passwords will often be written down, making them vulnerable to discovery, whereas passwords that are easier to remember will be easier to crack through dictionary and social engineering attacks.

In an attempt to improve security, authentication systems may impose controls on the passwords that can be chosen and the number of authentication attempts that can be made.

Attackers may attempt to eavesdrop on (i.e. capture) passwords as they traverse a system, or observe them as they are typed in. Passwords are also extremely vulnerable to social engineering attacks, whereby individuals may be fooled into revealing their password to an attacker. With passwords, it is the users that are the weakest link in the security chain: it is human nature to choose simple passwords, share them with others, or write them down.

Evaluation

There are several password handling systems that have been approved by the UK CESG Assisted Products Scheme (CAPS).

Cost

Password systems are often cheaper to deploy than other authentication mechanisms because they tend to require less investment in hardware. However, this lower initial expenditure does not necessarily mean that a password system is a cheaper option, as there are hidden costs involved in managing and maintaining passwords.

Users will always forget their passwords (whether or not they are complicated), so there needs to be a mechanism in place to deal with forgotten (or compromised) passwords. For large networks this may require the provision of a dedicated password helpdesk. To make matters even more complicated, it is good security practice to force users to change their passwords on a regular basis.

We therefore categorize this technology as representing medium value for money when compared with the other technologies discussed in this section.

Smart Cards and Tokens

The terms smart card and smart token are used in this paper to define integrated circuit microprocessor cards or tokens requiring a reader device. The reader itself is designed with connections in one of two forms: it can be plugged either into a computer's serial or USB port. The cards are generally credit card-sized, following the ISO 7816 standard. Smart tokens are produced in a variety of form factors.

The card or token may be either contact or contactless. Contactless smart cards and tokens do not require physical contact to communicate with the reader device. They rely on an antenna wound into the body of the card or token to send and receive data through electromagnetic fields. One advantage of contactless smart devices is a longer hardware lifetime, as there is no wear through contact with a reader for most applications.

Contactless cards are available as dual interface or hybrid smart cards, and therefore can be used in contact or contactless mode. A dual interface card incorporates two interfaces to a single chip whereas a hybrid card is essentially a contactless smart card with an extra contact chip.

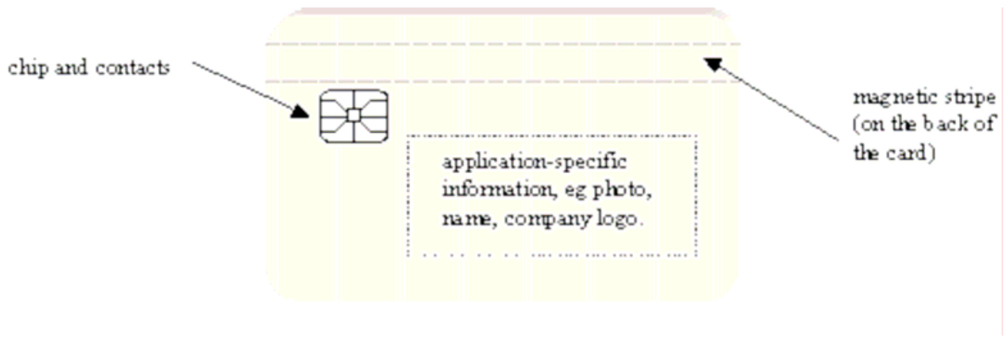

chip and contacts
application-specific information, eg photo, name, company logo.
magnetic stripe (on the back of the card)

Figure 3. Physical appearance of a smart card

Technical Maturity and Effectiveness

Smart cards are technically mature and widely used. They can perform multiple functions, for example combining IT network and building access control with storage of information such as medical records, financial information and biometric templates, all on one card or token. In their card form, smart devices can also be used as identity passes. They can easily incorporate identity information (e.g. photographs) and can also include a magnetic stripe for use with legacy systems.

Contactless cards are generally used for fast ticketing, such as in public transport, and building access applications. There is a possibility that such cards could be adapted for a secure log on application. However none are available at the present time and this would require extra system software to be written.

There are large-scale deployments of smart cards today, such as payment EMV (Europay, Mastercard and Visa) cards and the US Department of Defense (DOD) Common Access Cards (CACs).

User Impact

One reason that smart cards are widely used is that they are generally found to be acceptable to users. They are lightweight, portable, and easy to use and most people are used to carrying cards with them. However, as the smart card must be carried, there is the possibility that it will be forgotten or lost and, unless checked to confirm identity, users may share smart cards for convenience in the same way that they may share passwords. Smart cards are also not very robust and can be easily broken.

As smart cards and tokens can store, retrieve and process information, they can be used to generate and store private keys. Therefore they are ideal for use with Public Key Infrastructures (PKIs) as the private key is not held on the system and should never leave the card, giving good assurance that the private key has not been compromised. Currently available contactless cards use their contact interface for complex PKI functions.

Information exchange between the card and the system takes place using one of the following methods:

- complex (unguessable) passwords;
- one-way challenge response (using cryptography) whereby the system issues a challenge and the card responds;
- two-way challenge response (using cryptography) whereby the card also issues a challenge and the system responds.

Contactless smart cards use symmetric cryptography for challenge response.

Threats

Smart cards can be subject to two types of attack. Invasive attacks concentrate on a physical penetration of the card's integrated circuit to gain the information stored within. However, smart cards are tamper evident and an invasive attack will inevitably cause some damage to the card's surface, thus alerting the owner to the attack. Non-invasive attacks rely on observing and manipulating the card's operating environment and/or the signals that it transmits or receives in an attempt to gather useful information.

If the authentication secret held on a smart card becomes compromised, it can be easily revoked. A new secret can be generated or a new card can be issued to the authorized user.

Evaluation

Various smart card products have been certified to EAL4 under the Common Criteria scheme. The Infosec Assurance and Certification Services (IACS) at

CESG in the UK have evaluated two of these products. However, neither of these products are PKI-enabled smart cards. In cases where evaluation against Common Criteria has been achieved in the UK, it is also important to compare the evaluation against the protection profile used, to ensure that correct emphasis has been given to the required security aspects.

Cost

As smart cards require a reader device, this makes them expensive. The rough cost of a PKIenabled contact smart card and reader will be in the range of \$62-\$120 each based on a user group of 1000. The rough cost of a contactless smart card and reader is in the region of \$164.

We therefore categorise this technology as representing medium value for money when compared with the other technologies discussed in this paper. It should be noted that there will be many other costs associated with the deployment of this technology, for example there may be significant PKI (if appropriate) and underlying system costs as well as annual support costs.

USB Authentication Tokens

USB authentication tokens are small hardware devices that plug directly into the computer's USB port. Typically such tokens consist of a hard plastic outer casing that houses the electronic circuit and a metal USB port connector at one end. Although their appearance is completely different, USB authentication tokens are functionally very similar to the smart cards discussed in the previous section. However, there are some important differences in the technologies, which are discussed below.

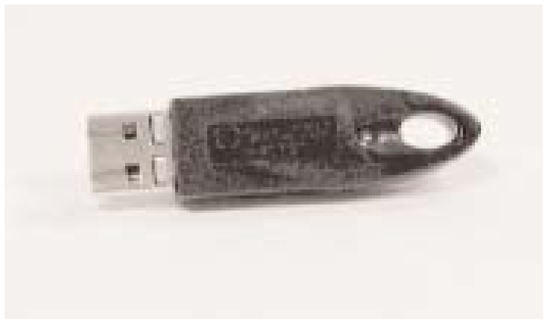


Figure 4. USB authentication token

Technical Maturity and Effectiveness and User Impact

One drawback of USB authentication tokens over smart cards is the amount of information or personalisation that can be displayed on their surface. These tokens are generally not designed to incorporate a legible digitised photograph of the user on their surface, which could be used to confirm the identity of the legitimate user.

Although technically mature, USB authentication tokens take up a very small share in today's market, and are mainly used for network access purposes. USB tokens are bulkier than smart cards, but may be carried easily on a key ring.

Threats

Tokens can afford better protection to the circuitry that they house. They are more robust than smart cards as the chip is held within a hard outer casing. They could also be filled with hard epoxy resin, and therefore there is more scope for tamper resistance measures.

However, these tokens can be less tamper evident than smart cards. If an attacker is able to remove and subsequently replace the hard plastic casing, attacks on the chip may go unnoticed. To avoid this, tamper evident seals can be incorporated onto the token surface.

Evaluation

Various USB authentication tokens use chips evaluated to E4 (equivalent to EAL5), although no evaluations have been carried out in the UK for this technology. FIPS 140–1 Level 2 certification (which requires that the product includes tamper evident seals or coatings) is also achieved by the main vendors. A current solution is undergoing evaluation for EAL2 certification in Australia.

Cost

As they do not require a separate reader, the cost of USB authentication tokens is lower than that of smart cards. Token cost per user is likely to be in the region of $52.

We therefore categorize this technology as representing high value for money when compared with the other technologies discussed in this proposal. As with smart cards, there will be many other costs associated with the deployment of this technology, such as PKI and underlying system costs as well as annual support costs.

Number Generation Tokens

Number generation tokens are hardware devices that generate 'pseudo-random' one-time numbers, which change periodically at regular time intervals. The token is synchronized with the authentication system, which contains the same code as the token. Thus this technology is based upon the fact that the system generates the same sequence of numbers at the same time. The user enters the number provided by their token onto the system and is authenticated if this number matches that produced by the system itself.

A keypad can be incorporated on the token, to allow PIN authentication to the token itself.

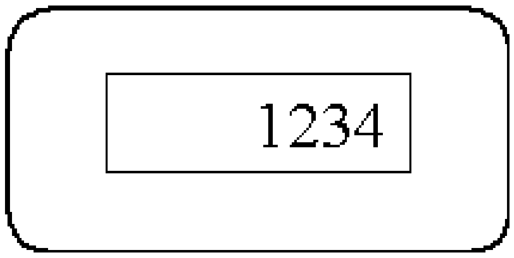


Figure 5. Appearance of a number generation token

Technical Maturity and Effectiveness

This technology is technically mature. The tokens are widely used, mainly for remote authentication and home working. Like the USB authentication tokens, they are robust and there is scope for tamper modification resistance as the chip is encased in a hard shell.

User Impact

This technology does not require any sort of reader; all that is required is a numeric keypad with which to enter the number onto the system. However, as the number must be physically entered, the process is more user-intensive and prone to human error. Mistakes are particularly likely to be made if the number generated.

The requirement for the user to type in a long string of digits, plus the bulkiness of the devices means that user acceptance of this technology may be lower than that of smart cards and USB authentication tokens. Some devices can be quite unwieldy, particularly those with a numeric keypad. By contrast, smart cards may be easily carried in wallets or on key rings. As with all tokens and cards, number generation tokens can be lost, stolen or shared with others.

As these tokens are continually performing calculations, they use a relatively large amount of power. In the absence of a reader, the power source is internal, and thus these tokens have a limited lifespan.

This authentication technology relies on the fact that the user will enter the same sequence of numbers as is produced by the system at that time. The major drawback to such a system is in cases where the number generation in the token and the system are not correctly synchronized.

There is no capability for storage of information on a number generation token. Therefore, as well as having a reduced functionality, number generation tokens are not suitable for use with a PKI as there is no scope for storing private keys on them. There is also no electronic interface to the system, which would be essential for PKI functions.

Threats

As there is no electronic interface with the computer system, the risks of electronic attack are reduced. The attacker must obtain the token before performing such attacks. As for smart cards and USB authentication tokens, there are possible physical or logical attacks that can be carried out on number generation tokens.

This technology uses symmetric keys in the process of generating the ‘pseudo-random’ number. Therefore a copy of each symmetric key relating to each number generation token will be stored on the system. This is a potential weakness of the authentication process, but as the symmetric key is combined with a timestamp, this makes the task of the attacker much more difficult because they must also correctly deduce the time and combination method used.

Number generation tokens can be revoked if they become compromised, and new tokens can be issued easily. The tokens may not be tamper evident, as it is possible for an attacker to penetrate the hard outer casing and reassemble it without causing damage that might alert the owner.

Evaluation

At present, there are no assured number generation tokens.

Cost

The cost of a number generation token is roughly \$53, based on 1000 users. Extra costs will be required, such as server software for this technology and annual support costs. We therefore categorize this technology as representing medium value for money when compared with the other technologies discussed in this paper.

Biometrics

Biometrics is essentially an automated process of the recognition of individuals through a physiological or behavioral characteristic. There are many different types of biometric, although the most commonly used are physiological ones: fingerprint and hand geometry, facial recognition and iris scanning.

For the authentication of a user, a capture device (biometric reader) takes a measurement, known as a template, of the particular characteristic (e.g. an electronic representation of the fingerprint) and this measurement is compared with a previously stored template. The authentication decision is then made based on the result of this comparison.

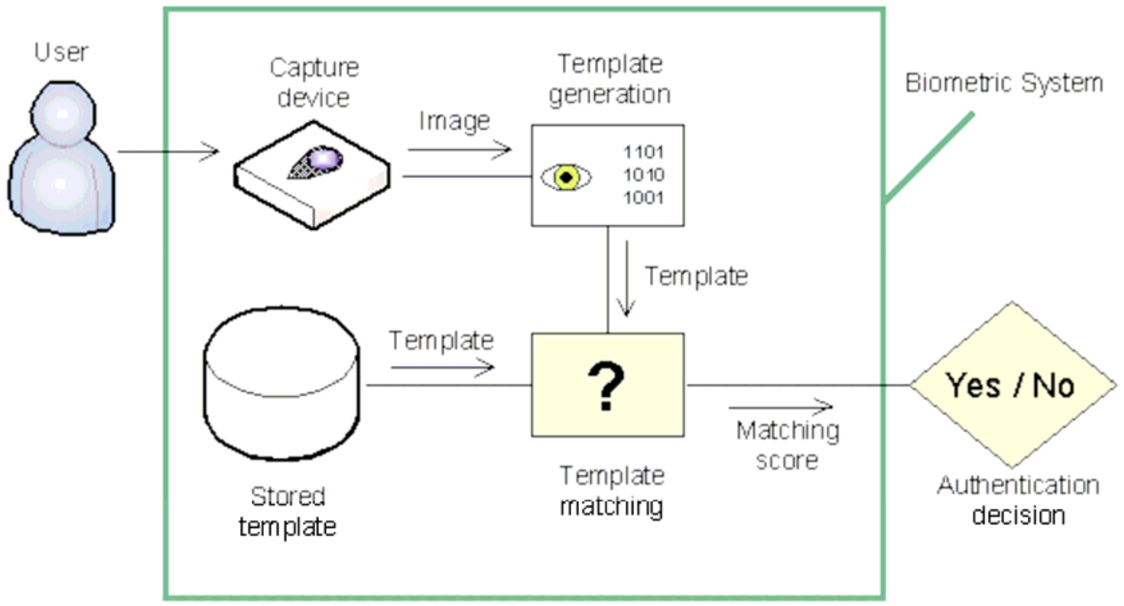


Figure 6. Biometric authentication

Technical Maturity and Effectiveness

This technology is not as technically mature as the cards and tokens discussed previously.

Biometrics companies are still fairly small and generally only poorly designed and implemented biometrics systems are available at the time of writing. Up to now there has been little emphasis on the security aspects of these systems, and therefore they are relatively easy to attack.

Since the conception of this technology, biometrics has been slow to take off. However, the rate of development and adoption has increased recently in response to the threat of terrorism, the US Enhanced Border Security Act and identity theft crackdown. The major uses of biometrics today are in prisons, at airports, passport/visa integration and for immigration purposes.

Unlike passwords, cards and tokens, the use of biometrics as the authentication mechanism is an inexact science. For any one individual, it is highly unlikely that two measurements taken of the same characteristic will be exactly the same. This may be due to fluctuations in the environment or the manner in which the individual interacts with the capture device. During the comparison process, a matching score is defined, which is essentially a measure of the degree of similarity between the two templates. If the score exceeds a predetermined threshold the templates are regarded as 'similar enough'; if not, the authentication attempt is rejected.

The threshold is usually determined through balancing two parameters, named the False Acceptance Rate (FAR) and the False Rejection Rate (FRR). The FAR gives a measure of probability that an individual who is not enrolled on the system is accepted; whereas the FRR indicates how likely it is that an authorized user will be rejected. Tuning the system to make the FAR lower (and hence a more secure system) means that the FRR will be higher, which could result in legitimate users being prevented from accessing critical information. Conversely, reducing the FRR to make the system more usable will result in an increased FAR, thus making it more likely that a non-authorized individual will be able to gain access. Obviously there will be a fine balance of FAR and FRR, which will entirely depend on the environment in which the system is to be used; for example, if strict physical measures are in place to protect the system it may be acceptable to increase the FAR.

User Impact

Unfortunately, there is a low user acceptance of biometrics. There are several reasons for this, including health and safety issues, association with criminal databases and users' fears that a third party may be able to deduce sensitive personal information such as certain medical conditions from the stored information. Some users may be unable to use the system at all. The characteristic that the system measures may have been lost damaged or not present in the first place.

As the biometric characteristic is part of the user, it cannot be forgotten, lost or stolen. Therefore this technology is more convenient than devices that a user must carry around and certain types of attack relevant to cards or tokens are eliminated. It is also unlikely that this form of authentication technology would be 'shared' with a colleague for convenience.

Storing users' biometric templates means that the Data Protection Act will apply with the use of this technology.

There is no multi-functionality with the technology. Private keys cannot be stored, so biometrics as a technology is not suitable for use with a PKI.

Threats

Biometric systems can be used in verification or identification mode. Verification mode is the traditional authentication mode whereby a one- to-one comparison of the individual against a previously stored template is undertaken. Identification mode consists of a one-to-many comparison of the individual against a database of stored templates from many different users. Identification mode is a far less secure method, as it gives an unauthorized user a much greater chance of matching a stored value in the database. Again, the mode employed will depend on the system requirements.

Many biometric systems can be spoofed: replay attacks can be deployed and false biometrics (such as the Matsumoto 'gummy' finger demonstration) or even dismembered limbs could be used. The original enrolment process will be critical to the security of the system.

Revocation of the biometric characteristic is impossible as a user cannot be given a new one! It is possible to use a second biometric characteristic, although there are only a finite number of characteristics that could be used. Biometrics has limited use within multiple systems, since the same (set of) biometrics must be used.

Evaluation

A fingerprint solution has been evaluated to EAL2 in Canada. An iris scanning solution has also been evaluated to EAL2, this time in Australia. In general, biometric suppliers are reluctant to publicize details of the algorithms used in their solutions.

Cost

The cost of a biometric reader will be in the region of $138 each based on a user-base of 1000. There are also associated annual support costs to be taken into consideration.

We therefore categorize this technology as representing low value for money, when compared with the other technologies.

Further Considerations for the Technology

Having analyzed each technology in turn, there are further issues that must be addressed and questions that must be answered when considering which technology or technologies to deploy for a given system. One important consideration is to determine to what extent the technology should be compatible with other programmes; for example, taking into account whether the existing infrastructure could be reused. It is also important to consider whether PKI enablement is a requirement or future requirement for the technology. Finally, current policy should be analyzed and a gap analysis undertaken and resolved, where appropriate, with respect to using the technology in the planned environment.

We have observed that PKI is becoming an increasingly important business driver in many sectors. It is therefore concluded that it would be short-sighted to not include a PKI capability with the technology. This does not imply that a full PKI is needed at this stage; the technology could simply be 'PKI-ready'. There are many products that can support both the password and PKI challenge-response information exchange functionality.

As, of the options considered, only smart cards and USB authentication tokens can store private keys, only they are compatible with PKI-based services. If PKI enablement is a requirement then, biometrics and number generation tokens can be excluded at this stage.

Boot authentication

Authentication is the process of determining that someone is the person he or she claims to be. When used with a personal computer, the end result of the authentication process is to grant or deny access to a computing resource. Authentication takes place in two steps:

1. Identification, where an identity is established.

2. Verification, double-checking the identity.

The Boot Authentication feature makes it possible to configure a system so that only authorized users are allowed to boot the machine into Single-user Mode.

Pre-Boot vs. Post-Boot Authentication

Notebook PCs by definition are roaming devices where the data contained on the computer is often far more valuable than the cost of the notebook. The number of notebook PC thefts continues to be a problem worldwide. To protect the physical assets and the data on those assets two types of authentication are may be implemented:

- Pre-OS or Pre-Boot Authentication (PBA)

- Post Boot or OS level protection

PBA protection is most relevant for biometric devices that are integrated into the notebook PC where BIOS level software uses the device to authenticate the user during the power-on sequence. Additional data protection is provided by PBA with some increased complexity for integrating such support into the platform.

Short term focus of these guidelines will cover only the Post Boot or OS level protection using fingerprint sensors for user authentication. Additional architecture work for PBA may be a natural follow on to these guidelines as driven by customer demand for improved security implementations.

BAPI (Biometric Application Programming Interface): A software interface standard for biometric devices used to generalize communications between an application program and a biometric hardware device. BAPI is a standard published by IO Software, and is currently licensed by Microsoft.

BioAPI (Biometric Application Programming Interface): A software interface standard for biometric devices used to generalize communications between an application program and a biometric hardware device. This is a software interface standard defined by an industry group.

Biometric generally refers to a device that is associated with a unique human characteristic such as a fingerprint, an iris, hand or other characteristic that can be used to uniquely identify an individual.

Pre-boot biometric authentication

Background

For the majority of businesses and organizations, information is considered to be an asset, and so worthy of protection. Information security can support a wide variety of objectives, including:

- Compliance with laws and regulations;
- Reducing the risk of fraud or other falsification of data to an acceptable level;
- Reducing the risk of unauthorized access or disclosure to an acceptable level.

The protection afforded to information is usually expressed in terms of the following categories:

- Confidentiality – concerned with protecting information from unauthorized disclosure;
- Integrity – protecting information from unauthorized modification in order to preserve its accuracy and completeness;
- Availability – ensuring that authorized people are able to access information when they need to without undue delay;
- Non-repudiation – ensuring that a user who performs an action that could have an impact on the security of information cannot later refute that action.

No technique, device or procedure is going to provide all of these services. Authentication, however, has a significant contribution to make to all of them. Acceptable authentication processes allow an organization to have a reasonable degree of assurance that the people who read, originate, send, or alter material on an information system are:

- Who they claim to be;
- Have the authority to do whatever it is they are doing;
- Cannot avoid accountability for their actions.

Biometrics (the ‘measurement of people’) is one approach to the authentication of an individual’s claimed identity, and therefore to the authentication of that person’s claimed rights to gain access to particular processes, systems or pieces of information. It is an attractive solution because of the perception that aspects of one’s physiology can not be reliably emulated. There is a range of products on the market that provide biometrics-based authentication. Biometrics is currently the subject of sustained and rigorous academic study, and is now incorporated into several large-scale authentication processes (such as US immigration control).

Biometric technologies [3] have been seized upon by writers of fiction, and have found a place in the popular imagination because of this culturally assisted perception of reliability, enhanced by their public use in authentication pilots and projects. But biometrics does not provide certainty. Information security as a whole – dealing as it does to a large extent with human behaviors and motivation – is not infallible. As a technique, biometric authentication has both strengths and weaknesses, but cannot provide a cast-iron guarantee of identity. It will sometimes produce false responses, for reasons that are set out in this paper; but when used wisely and when its limitations are recognized, biometrics can provide a valuable degree of assurance in verification of identity and enhance the authentication processes that can be critical to the protection of an organization’s assets.

This proposal examines the contributions that biometrics may make to the processes of authentication within a service, and the security issues that surround the use of biometric technologies.

Related Techniques

In general, there are three main factors that might be taken into account when authenticating a user to a service: ‘something that is known’, ‘something that is possessed’ and ‘something that a person is’.

The first factor, ‘something that is known’, usually takes the form of a password or PIN (Personal Identification Number). Traditionally, this has been the most common authentication factor. However, recent trends reflect a growing realization that the users of the service often adversely affect the strength of this approach. Users are notoriously bad at choosing and retaining passwords (particularly if they require access to multiple systems), and can be deceived into revealing them through so-called 'social engineering' attacks. The combination of increasingly powerful and relatively inexpensive computers means that attackers are better placed to carry out 'bruteforce' attacks against password mechanisms, while users are ill equipped to deal with the increased password complexity required to protect against such attacks.

The second factor, 'something that is possessed', involves issuing users with a smartcard or token, which is used to identify that user to the service. Often, the token is simply a convenient and secure place to store a password, which is passed to the system when requested. The alternative is a more intelligent device that is capable of responding to a challenge issued by the service. The response is crafted on the token by combining the challenge with some secret information stored on the token. As an authentication mechanism, smartcards and tokens are growing in popularity. A great advantage is their multi-functionality – the smartcard used to access a service can store public key certificates, access buildings, carry an 'electronic purse', collect loyalty points, and a host of other functions.

However, the disadvantage is that they can be lost or stolen, not only preventing legitimate users from gaining access, but also increasing the risk that an unauthorized individual could authenticate to the service using the card.

This proposal is concerned primarily with the third factor: 'something that a person is'. Recognizing individuals through observation of particular physiological characteristics is known as biometrics. A biometric authentication is a two stage process: Firstly, some sort of capture device is used to take a measurement of a particular physiological or behavioral characteristic, and secondly, that measurement is compared to a stored value. Based on the result of this comparison, the system makes an authentication decision.

Often, multiple authentication factors are used to increase the strength of the overall authentication mechanism. For instance, tokens are often used in conjunction with a PIN to mitigate the risk posed by their loss or theft, the idea being that both the token and the PIN are required to gain access to the service. However, the security gained through the use of multiple factors must be weighed against the extra management and purchase costs, and the additional burden that is placed on the user.

Once logged-on, there are other processes within a service that might require users to authenticate themselves. An obvious example is access to a private key for decrypting or digitally signing information within a Public Key Infrastructure (PKI). The security of these processes rely on the fact that only the legitimate owner of the key is able to access it, so it is important that the system properly authenticates the user before permitting them to use the key.

Examples of physiological characteristics that are used in biometric devices include fingerprints; the geometry of the face or hand; and patterns within the iris or retina, or in the layout of veins. Behavioral characteristics include voice pattern; gait (the manner in which an individual walks); and the dynamics of handwriting (signatures in particular) or keystrokes (i.e. keyboard typing patterns). Of course, for the authentication mechanism to be secure, the chosen characteristic must be unique to each individual. Also, it must be possible to measure the characteristic with a reasonable degree of accuracy.

Once a measurement has been taken, the data is converted into a biometric template. A template is a representation of the measurement that retains all the relevant information but takes up far less space than the original. It is this template that is compared to a template generated in the same manner during the initial enrolment procedure, and based on the similarity of the two a decision is taken on whether the user should be granted access.

For any one particular individual, it is highly unlikely that two measurements of the same characteristic will be identical. There are simply too many factors that can affect the result. The measurement may be influenced by fluctuations in the environment (such as variations in heat, light, humidity, and so on). The manner in which the user interacts with the capture device also has an impact on the measurement. On different occasions, an individual is likely to behave slightly differently, for example they might adopt a different stance, expression, exert a different amount of pressure on the reader, move while the measurement is taken, etc. Even in the absence of all these factors, physiological and behavioral characteristics vary over time. Fingerprints become worn with hard labor and age, voices are affected by illness, and signatures vary depending on the emotional state of the subject. Even faces change over time – gradual changes in shape and complexion, as well as sudden changes such as the growth of a beard, or the addition of glasses, make-up, or a physical injury.

Such variations will often result in the generation of different templates. This makes the authentication decision far more complex than it is for passwords. The system compares the newly generated template against the one that was generated during enrolment, and calculates a matching score – a measure of the degree of similarity between the two. If this score exceeds a predetermined threshold set by the system designers (i.e., if the templates are 'similar enough'), the identity of the individual is confirmed; if not, it is rejected. Other related work can be found in [12-23].

Modules of a biometric system

Any biometric system is basically made of the following components [4]:

1. Portal. Its purpose is to protect some assets. An example of a portal is the gate at an entrance of a building. If the user has been successfully authenticated and is authorized to access an object then access is granted.

2. Central controlling unit receives the authentication request, controls the biometric authentication process and returns the result of user authentication.

3. Input device. The aim of the input device is biometric data acquisition. During the acquisition process user's liveness and quality of the sample may be verified.

4. Feature extraction module processes the biometric data. The output of the module is a set of extracted features suitable for the matching algorithm. During the feature extraction process the module may also evaluate quality of the input biometric data.

5. Storage of biometric templates. This will typically be some kind of a database. Biometric templates can also be stored on a user-held medium (e.g., smartcard). In that case a link between the user and her biometric template must exist (e.g., in the form of an attribute certificate).

6. The biometric matching algorithm compares the current biometric features with the stored template. The desired security threshold level may be a parameter of the matching process. In this case the result of the matching will be a yes/no answer. Otherwise a score representing the similarity between the template and the current biometric sample is returned. The central unit then makes the yes/no decision.

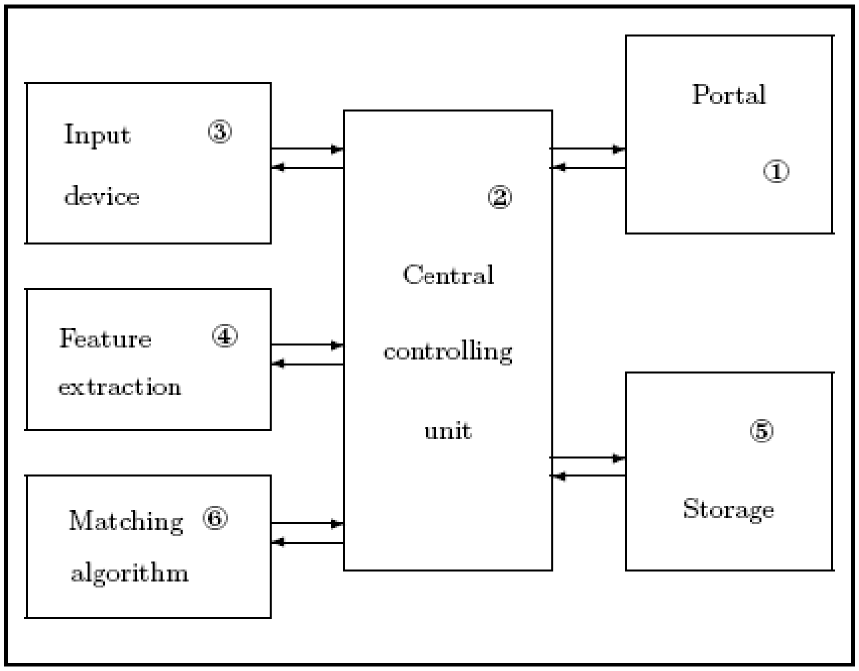


Figure 7. Model of a biometric system

Parameters of biometric systems

What does it take for one biometric system to be more secure than another one? What are the differences among various systems?

Liveness testing: Incorporation of a liveness test makes an attack against the biometric system more difficult. There are various liveness tests offering various levels of protection. Most of the tests, however, can be easily cheated. A combination of multiple liveness tests can make the system more secure.

Tamper resistance: If the biometric system is not under constant human supervision it has to rely on tamper resistance. Without tamper resistance or supervision the system can be tampered with and forged/replied biometric data can be injected into the system.

Secure communication: Biometric system components can be either standalone and communicate with each other over an external insecure medium or can be coupled in a tamper-resistant box. The communication among modules within a tamper-resistant cover need not be secured, but the communication over an insecure line should be authenticated and encrypted.

Security threshold level: Lower false acceptance rate means higher level of security (and unfortunately, in most cases, also higher false rejection rate causing user frustration). A proper value must be set in accordance with goals of the biometric system.

Fall-back mode: In some systems the biometric authentication may be sufficient for the user authentication. In some systems an additional authentication method must be used and the biometric authentication is only a necessary part of user authentication. Successful authentication using this additional method may but need not be sufficient for user authentication.

FAR and FRR: Usually, the permissible discrepancy between templates is determined by adjusting the False Acceptance Rate (FAR) and the False Rejection Rate (FRR). The FAR gives a measure of the probability that an individual will be ‘falsely accepted’ by the system (i.e. that the system accepts someone who is not enrolled on the system, or makes an incorrect identification). The FRR indicates how likely it is that a legitimate user will be rejected. The balance between the FAR and the FRR greatly influences both the security and usability of the system. Tuning the system to make the FAR lower will improve its security, as fewer impostors will be accepted. However, this will result in a higher FRR, meaning that legitimate users are rejected, thus making the system less usable. This may be acceptable in high security environments where it is wise to err on the side of caution, but if the assets that the system is protecting are not of great value, users will soon become annoyed. Conversely, reducing the FRR to make the system more user-friendly means that the FAR is increased, resulting in a less secure system. Again, whether this is acceptable depends entirely on the environment in which the system is deployed.

Since the FAR/FRR balance may be adjusted, neither rate provides a suitable metric for expressing the overall accuracy of the biometric system. The Crossover Error Rate (CER) is defined as the error rate of the system when the FAR and FRR are equal. Usually expressed as a percentage, the CER can be used to compare the performance of different biometric systems.

Performance of Biometric Systems

Identification vs. Verification

In some cases, authentication is achieved by a comparison with a claimed identity. Authentication that incorporates this type of ‘one-to- one’ matching is known as verification. Alternatively, authentication might be achieved by comparing a candidate against a group of possible identities. In this case, it is the authentication mechanism that decides on the identity. Authentication achieved through a ‘one-to-many’ matching such as this is known as identification.

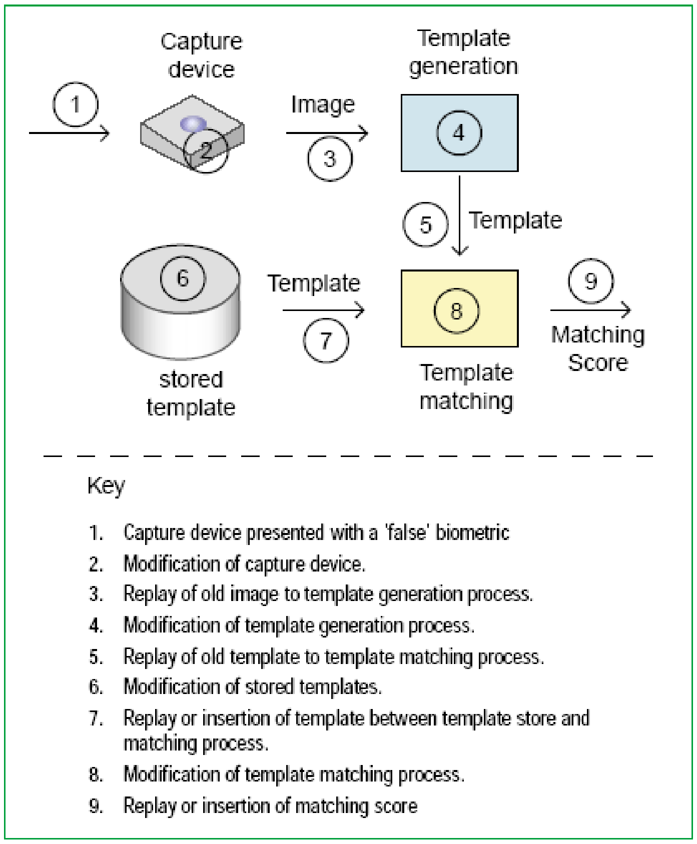
Although biometric technologies can be designed to support either type of authentication, careful consideration is needed before using biometrics in identification mode. Far greater security may be achieved by demanding that service users follow a verification procedure instead. To illustrate why this is the case, consider a biometric with a FAR of 0.01%. In verification mode, this might be considered quite accurate, giving a 1 in 10000 chance that an impostor will be accepted. However, if the same system is used in identification mode on a database of just 500 users, the probability of a false match is nearly 1 in 20. Thus, a biometric technology used in identification mode must be significantly more accurate than a verification mode technology if equivalent levels of security are to be achieved.

Threats to a Biometric Authentication

This section discusses the main threats to a biometric authentication system. It should be noted that the threats identified are applicable to a ‘general’ biometric authentication system comprising the components that were illustrated previously in Figure 6. Any particular system is likely to be subject to additional threats that are specific to the design of that system. Figure 8 below identifies the main threats to a general system:

The first general threat is the use of a ‘false’ biometric. The capture device might be fooled into accepting an imitation (usually referred to as a ‘false artefact’), or the real thing that has been separated from its owner (such as a recording of a voice or a severed finger). This is represented by threat 1 in the Figure.

The second major threat is concerned with modification of the components of the authentication system. The capture device could be modified to output a different image to the one captured. Alternatively, it could be modified to create a denial-of-service (DoS) that will prevent legitimate users accessing the system (by smashing a camera, for example). The processes of template generation or template matching could be subverted to produce erroneous results; for example, a piece of malicious code could interfere with the template generation software to produce the attacker’s template rather than that of the legitimate user, or the matching process could be modified to produce an artificially high or low matching score. These are threats 2, 4 and 8 in Figure 8.



Key

1. Capture device presented with a 'false' biometric
2. Modification of capture device.
3. Replay of old image to template generation process.
4. Modification of template generation process.
5. Replay of old template to template matching process.
6. Modification of stored templates.
7. Replay or insertion of template between template store and matching process.
8. Modification of template matching process.
9. Replay or insertion of matching score

Figure 8. Threats to a biometric authentication system

An ideal form of attack on a biometric system is for an attacker to insert their own template into the template store. If this can be achieved, the attacker will appear to the system to be a legitimate user. Another possibility is to modify the stored template of a particular individual so that they are no longer authenticated by the system (or alternatively, if the system works in identification mode, swap templates around so that one individual is identified as another). This is threat 6 in Figure 8.

Rather than modify the components of the system, an attacker might instead try to subvert the communications between those components. It might be possible to replay old images to the template generator, or old templates to the template matcher, to fool the system into treating an attacker as an individual who has previously been authenticated. It might also be feasible to inject the wrong template into the transmission from the template store to the matching algorithm. Alternatively, the entire template generation and matching process could be effectively bypassed by the insertion or replay of an erroneous matching score. These possibilities are illustrated by threats 3, 5, 7, and 9 in Figure 8.

For each particular scenario, the level of risk posed by the threats listed above is likely to vary. To ensure that the relevant threats are mitigated, the design of the biometric system must take into account the environment within which the system will operate.

Advantages and disadvantages

The primary advantage of biometric authentication methods over other methods of user authentication is that they really do what they should, i.e., they authenticate the user. These methods use real human physiological or behavioral characteristics to authenticate users. These biometric characteristics are (more or less) permanent and not changeable. It is also not easy (although in some cases not principally impossible) to change one's fingerprint, iris or other biometric characteristics.

Users cannot pass their biometric characteristics to other users as easily as they do with their cards or passwords. Biometric objects cannot be stolen as tokens, keys, cards or other objects used for the traditional user authentication, yet biometric characteristics can be stolen from computer systems and networks. Biometric characteristics are not secret and therefore the availability of a user's fingerprint or iris pattern does not break security the same way as availability of the user's password. Even the use of dead or artificial biometric characteristics should not let the attacker in.

Most biometric techniques are based on something that cannot be lost or forgotten. This is an advantage for users as well as for system administrators because the problems and costs associated with lost, reissued or temporarily issued tokens/cards/passwords can be avoided, thus saving some costs of the system management.

Another advantage of biometric authentication systems may be their speed. The authentication of a habituated user using an iris-based identification system may take 2 (or 3) seconds while finding your key ring, locating the right key and using it may take some 5 (or 10) seconds.

So why do not we use biometrics everywhere instead of passwords or tokens? Nothing is perfect, and biometric authentication methods also have their own shortcomings. First of all the performance of biometric systems is not ideal (yet?). Biometric systems still need to be improved in the terms of accuracy and speed. Biometric systems with the false rejection rate under 1% (together with a reasonably low false acceptance rate) are still rare today. Although few biometric systems are fast and accurate (in terms of low false acceptance rate) enough to allow identification (automatically recognizing the user identity), most of current systems are suitable for the verification only, as the false acceptance rate is too high.

The fail to enroll rate brings up another important problem. Not all users can use any given biometric system. People without hands cannot use fingerprint or hand-based systems3. Visually impaired people have difficulties using iris or retina based techniques. As not all users are able to use a specific biometric system, the authentication system must be extended to handle users falling into the FTE category. This can make the resulting system more complicated, less secure or more expensive. Even enrolled users can have difficulties using a biometric system. The FTE rate says how many of the input samples are of insufficient quality. Data acquisition must be repeated if the quality of input sample is not sufficient for further processing and this would be annoying for users.

Biometric data are not considered to be secret and security of a biometric system cannot be based on the secrecy of user's biometric characteristics. The server cannot authenticate the user just after receiving her correct biometric characteristics. The user authentication can be successful only when user's characteristics are fresh and have been collected from the user being authenticated. This implies that the biometric input device must be trusted. Its authenticity should be verified (unless the device and the link are physically secure) and user's liveness would be checked. The input device also should be under human supervision or tamper-resistant. The fact that biometric characteristics are not secret brings some issues that traditional authentication systems need not deal with. Many of the current biometric systems are not aware of this fact and therefore the security level they offer is limited.

Some biometric sensors (particularly those having contact with users) also have a limited lifetime. While a magnetic card reader may be used for years (or even decades), the optical fingerprint reader (if heavily used) must be regularly cleaned and even then the lifetime need not exceed one year.

Biometric systems may violate user's privacy. Biometric characteristics are sensitive data that may contain a lot of personal information. The DNA (being the typical example) contains (among others) the user's preposition to diseases. This may be a very interesting piece of information for an insurance company. The body odour can provide information about user's recent activities. It is also told that people with asymmetric fingerprints are more likely to be homosexually oriented, etc.

Use of biometric systems may also imply loss of anonymity. While one can have multiple identities when authentication methods are based on something the user knows or has, biometric systems can sometimes link all user actions to a single identity.

Biometric systems can potentially be quite troublesome for some users. These users find some biometric systems intrusive or personally invasive. Even if no biometric system is really dangerous, users are occasionally afraid of something they do not know much about. In some countries people do not like to touch something that has already been touched many times (e.g., biometric sensor), while in some countries people do not like to be photographed or their faces are completely covered.

Lack of standards (or ignorance of standards) may also posses a serious problem. Two similar biometric systems from two different vendors are not likely to interoperate at present.

Defence in Depth Approaches

Multifactor Authentication

As mentioned earlier, one way to increase the strength of an authentication mechanism is to use multiple factors of authentication. In the case of biometrics, this could involve requiring the user to input a password or PIN, or to produce some sort of authentication token (such as a smartcard or USB token). The advantage of cards and tokens is that many are designed to operate with biometric systems and have sufficient space for storage of biometric templates, enabling users to carry their templates with them. However, assessing the extent to which an additional authentication factor can increase the overall strength of the authentication service is far from trivial.

Multimodal Biometrics

A multimodal biometric authentication system combines two or more biometric measurements to reach an authentication decision. There are various reasons for wanting to use multimodal biometrics. Firstly, with any particular biometric there are always some individuals who are unable to use the system. For example, based on testing they have carried out, NIST estimate that 2% of the population "may not be easily fingerprinted ". A multimodal system increases the chances that a user will be able to use at least one of the biometrics. Another reason for using a multimodal system is a desire for a higher degree of accuracy than may be achieved through a single system. If a user is required to pass two separate tests, the resulting authentication can be stronger than if a single test result had been taken in isolation. Alternatively, the individual measurements taken might be of a particularly poor quality (due to environmental conditions, uncooperative users etc.), so a multimodal system might serve to increase the system's accuracy to the same level as provided by a single biometric used under ideal conditions.

The key to achieving a multimodal system that satisfies the requirements is to choose the appropriate method for combining the individual matching scores. The main possibilities are discussed below, along with their likelihood of achieving the different needs listed above.

Methods for Combining Individual Decisions

Suppose there are N individual biometric authentication systems, B1, B2, …, BN, and suppose that the output of each system is an accept/reject decision. One way to form a multimodal authentication system is to require a user to pass at least k out of the N individual tests, for some k ≤ N. For each individual system, an 'accept' decision could be represented by a '1' and a 'reject' decision by a '0' – in this case the sum of all the individual 0s and 1s would need to total at least k for the user to be accepted. This system could be used to overcome the problem that some users are unable to use a particular biometric. For example, imagine a multimodal system comprising face recognition, voice recognition and an iris scan (N = 3). If k = 2, each user is required to pass only two of the three tests, which makes allowances for users that are unable to use one of the three systems. With k set to 1, users would only need to pass one of the three. Another option might be to vary the requirements according to the classification of the system or data in question (the higher the classification the more stringent the requirements). For instance, in a multimodal system with three biometrics (N = 3), the user might be required to pass one test to access Restricted material, two tests to access Secret, and all three to access Above Secret.

Clearly, the drawback with this method is that the overall strength of authentication is limited to the k weakest individual systems. This is a particular problem where k = 1, and the attacker is able to choose the least accurate biometric system to attempt to gain false authentication. With this in mind, an alternative option would be to require the user to pass all N individual tests (i.e., k = N). This would almost certainly result in a stronger authentication than a decision based on some subset of the individual systems, but might prove to be unusable in practice. The problem is that the larger k becomes, the higher the chances are that a legitimate user will be falsely rejected.

A compromise might be to give the individual systems a 'weighting', based on factors such as environmental conditions, accuracy (in terms of equal error rate) and so on. So, rather than always awarding an individual 'accept' decision a score of '1', it could, for example, be given a score out of ten, reflecting the confidence that can reasonably be placed in that decision. In this manner, more credence could be given to, say, a positive authentication decision from an accurate fingerprint biometric used in ideal conditions than to a 'covert' face recognition system operating at a distance in poor visibility.

The idea of applying a weighting to each individual authentication decision can be taken a step further by allowing the weightings to vary according to conditions. For example, a high weighting applied to a face recognition system used in daylight hours could be gradually reduced as night falls, reducing to a very low weighting at night.

Conclusions and Further Research

Current COTS (Commercial-off-the-Shelf) biometric technologies are focused on the authentication of individuals. Initial research suggests that biometrics systems could also be used to indicate the present physical or mental state of an individual. Factors such as ‘liveness’ (confirmation that the subject is alive), state of health, and level of stress or anxiety could all contribute to the level of trust that is apportioned to a particular individual. However, further research would be needed to ascertain precisely how this could be achieved and how accurate it might be.

When passwords or tokens are used for authentication, the decision made is relatively straightforward – if the correct password / token is supplied the result is a positive authentication, otherwise the individual is rejected. A biometric authentication is conceptually different, in that the decision is based on a probability. Any organization considering the use of biometrics needs to understand the impact of this when reaching a trust decision.

Biometrics is not secrets. The security in a biometric system depends on the ability to ensure that a reading is authentic (i.e. current, and belongs to the individual in question). For this reason, a ‘secure’ biometric system would be one that used a second authentication factor, such as a password or token.

A biometric used in identification mode is far less reliable than the same biometric used in verification mode, making the latter the preferred option. Where identification mode is used, the overall accuracy of the system will need to be much greater if the reliability is to be equivalent to verification mode.

The performance of a biometric system is very dependent of the architecture of the system. While the figures quoted in this report give a good indication, further tests need to be conducted on systems that meet the specific requirements of the scenario in question.

The success of a multimodal biometric system will depend on the method used to combine the individual decisions or matching scores. Further research is needed in this area, particularly into the prospect of using variable weightings based on environmental factors or system classification.

By far, what we are talking about is the user authentication, that is, how to verify that the user who wants to get into the computer is authorized. On the other hand, the computer and all the programs also need to be authenticated before the valid user run them. Authenticated boot in the next section blow is focused on the authentication during the running of a computer.

Authenticated Boot

The authenticated boot process [8] depicted in Figure. 9 below provides a good example of transitive trust. During authenticated boot the TCPA-compliant platform measures its boot process and stores the results of that measurement in the TPM. Integrity metrics can be derived from the BIOS, boot-loader, OS loader, and the OS security policy, using cryptographic hashing to extend transitive trust from the BIOS to other areas of the platform. There is no attempt to stop the boot process or to change the process flow. The stored measurements can be used by challengers to determine what boot process occurred. The measurement can be reported multiple times. The benefit is that challengers of the platform can trust that the measurement, storage, and reporting of integrity metrics is done in a secure manner.

The measurement process takes a cryptographic hash of the entity to measure. For authenticated boot, this starts with the hashing of the BIOS. The resulting hash is stored in the TPM. When challenged, the TPM digitally signs the stored integrity metric and returns it to the challenger. The challenger bases its trust in the remote platform on the credential associated with the key used to sign the integrity metrics.

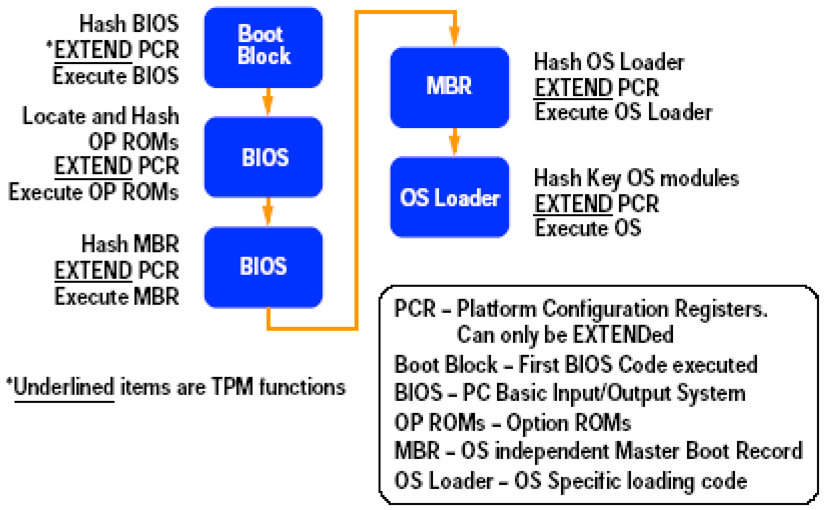

Hash BIOS
*EXTEND PCR
Execute BIOS
Boot Block
Locate and Hash OP ROMs
EXTEND PCR
Execute OP ROMs
BIOS
Hash MBR
EXTEND PCR
Execute MBR
BIOS
MBR
Hash OS Loader
EXTEND PCR
Execute OS Loader
OS Loader
Hash Key OS modules
EXTEND PCR
Execute OS
PCR - Platform Configuration Registers. Can only be EXTENDed
Boot Block - First BIOS Code executed
BIOS - PC Basic Input/Output System
OP ROMs - Option ROMs
MBR - OS independent Master Boot Record
OS Loader - OS Specific loading code
*Underlined items are TPM functions

Figure 9. Authenticated boot process

The trust that a challenger places on a remote platform is based on transitive trust. The transitive process starts from the root of measurement trust, which measures the next application, stores the measurement, and then passes control on to the next application (e.g., from the BIOS to the option ROMs to the operating system loader, and so on). If the measurement is done correctly, the challenger knows what the chain of applications is. If a "bad" application exists, the bad application cannot perform the right measurement. In addition, the previous application would have measured the bad application and reported it. The bad application can still run; it just cannot hide its presence. This preserves the openness of the system while enabling a chain of trust to be built from the trust root. The challenger can make a decision regarding the trustworthiness of the remote platform for its intended purpose.

Authentication for wireless networks

Authentication is critical in authoring operations on mobile station registration, accessing and/or sending private information. In a mobile wireless network, the authentication scheme cannot be too complicated or computation intensive at the mobile station side because part of the authentication process is executed on a mobile station which has hardware limitation. However, another fact is the computation units for mobile stations are getting faster and consume less power from batteries, which allows more complicated cryptographic algorithms, such as the public-key algorithms, running on mobile stations lightly. We will find that there are some limitations in the current schemes to work with Internet applications sunning on mobile stations.

The traditional authentication mechanisms in current Internet applications are not quite suitable for wireless environment either because they are originally designed for traditional Internet, or fixed network environment. And now they considered being unsafe even in fixed network environment. For example, some real information was sent by the email client while my email was being retrieved from the email server in the lab. Considering the session key generated in the registration process mentioned above, a foreign network can then decrypt the information that the mobile station send if it wants to get the private information right away. So far, it can be seen that there is no specific or suitable authentication scheme designed for the authentication in wireless Internet applications [11].

Introduction

Because the broadcast data on the open airways, wireless networks present unique challenges for authentication mechanisms not encountered on wired networks. In this section, we explores how wireless networks are different from wired networks with regard to authentication and presents the requirements that an authentication method must meet in order to be appropriate for wireless networks. We then consider several families of authentication methods that have been designed specifically around the needs of wireless networks -- the public key certificate-based methods, the password methods, and the strong password methods. One particular strong password method, known as SPEKE for Simple Password-authenticated Exponential Key Exchange, is examined in some detail. The comparison of different authentication methods and conclusion are given at the end of this part.

Authentication is the process of verifying a claimed identity. In perhaps the earliest form of authentication, the person being authenticated--called the user in this proposal --would present a password to the authority requiring authentication--called the authenticator. If the user were able to present the correct password, he or she would be authorized to gain access to something or to receive services. For some purposes, simple password authentication can provide relatively strong security, but in order to do so, certain assumptions must hold true:

- The user must have some assurance that the authenticator is in fact the authority in question.

- The communication channel between the user and the authenticator must itself be secure (user and authenticator can be sure that no one is listening).

- It must be highly unlikely that an attacker would be able to guess the password. Usually this is accomplished by limiting the number of wrong guesses.

- If the user is a human being (as opposed, say, to a software process running on a computer), the password must be easy to remember--but not so easy that it can be easily guessed.

Today's wireless networks are not the timesharing system. Consider a user with a laptop computer accessing an 802.11 wireless network. The first problem is that the user has no way of knowing whether the access point is, in fact, operated by the administrator of that network. It might be a rogue access point operated by another user (an imposter) who may have a connection to the target network. If so, the user we are concerned with may not even know that the data is being routed through an imposter's computer.

The second problem is that the communication channel in this case is a radio network that can be monitored by anyone with a radio receiver. It is easy for an attacker to monitor legitimate user's access attempts and collect their passwords without being detected. This problem can be mitigated somewhat through using a challenge/response authentication system in which the password is not itself transmitted over the air, but the user is presented with a challenge that is joined with the password and hashes with a secure hash function.

But now we have a new problem. The attacker can make password guesses on a separate computer by observing a single challenge and response and then attempting to join the challenge to his guesses, computing the resulting response, and comparing it to the observed response. Guesses can then be made at a very fast rate with neither the user nor the network administrator knowing about it. This form of attack is known as a dictionary attack because the attacker selects his guesses from a cracker's "dictionary" of possible passwords.

Offline dictionary attacks can be mitigated by using a large random number in place of an easily remembered password. This makes it unlikely that the password would be in the attacker's dictionary. But this violates the fourth assumption that the password be easy to remember. To get around this problem, the password can be stored on the user's computer, but now the user has to prevent the attacker from gaining access to it by walking up to the computer without the user's knowledge or stealing the computer or, more alarmingly, by gaining unauthorized access to the user's computer over the very network the user is trying to use.

As you can see, the requirements for wireless network authentication are much more stringent than those placed by a dialup timesharing system.

Requirements for wireless authentication

What then are the requirements for an authentication method that will be used to gain access to a wireless network? The following sections list requirements that an authentication method must meet (must have), additional characteristics that are highly desirable (should have), and features that may be quite useful in certain environments (may have).

Requirements (must have)

Mutual --- It must provide mutual authentication, that is, the authenticator must authenticate the user, but the user must be able to authenticate the authenticator as well. Mutual authentication is particularly important over wireless networks because of the ease with which an attacker can set up a rogue access point. There are two possible attacks here. In one, the rogue is not connected to the target network and merely wishes to trick the user into divulging authentication credentials. In the other, the rogue is connected to the target network. The attacker may then ignore the credentials presented by the user and "authorize" network access. The user's session may then be recorded or even altered because the attacker has been inserted in the data path.

Self-protecting --- It must protect itself from eavesdropping since the physical medium is not secure. The authentication must proceed in such a way that eavesdroppers cannot learn anything useful that would allow them to impersonate the user later.

Immune to Dictionary Attacks --- It must not be susceptible to online or offline dictionary attacks. An online attack is one where the imposter must make repeated tries against the authenticator "on line" These can be thwarted by limiting the number of failed authentication attempts a user can have. An offline attack is one where attackers can make repeated tries on their own computers, very rapidly, and without the knowledge of the authenticator. Simple challenge/response methods are susceptible to offline attacks because if attackers capture a single challenge/response pair, they can try all the passwords in the dictionary to see if one produces the desired response.

Produces Session Keys --- It must produce session keys that can be used to provide message authentication, confidentiality, and integrity protection for the session the user is seeking to establish. These keys will be passed to the user's device drivers to be used as WEP or TKIP keys during the ensuing session.

Additional characteristics (should have)

Authenticates User --- It should authenticate the user rather than the user device. In that way it will be hardened against attacks against the user device. One useful way to meet this requirement would be for the method to depend on a simple secret that can easily be remembered by the user. Another way is to encase the secret in a smart card that is carried by the user and is separate from the device.

Forward Secrecy --- It should provide forward secrecy. Forward secrecy means that the user's secret, whether password or secret key, cannot be compromised at some point in the future. An attacker who recorded a user's session encrypted by a key produced during authentication cannot, given knowledge of the user's secret, decrypt the recorded session. Once secure, the session data stays secure forever.

Access Points --- It should work with all access points that support 802.1x with EAP authentication.

Quick and Efficient --- The authentication should complete in a minimal number of protocol round trips, and computations necessary to complete the authentication should require a minimal amount of computing resources.

Low Maintenance Cost --- It should be easy to administer. A method that requires the installation of a certificate on each user device, for example, is not easy to administer. Maintenance of certificate revocation lists can be a costly administrative burden.

Convenient for Users --- It should be convenient enough to use that users will not balk. For example, using a certificate stored on a device, though, burdensome to administrators, is convenient for users. Smart cards, though inconvenient for users, are easier for administrators. Users don't mind typing a small, easy to remember password, but most would object to typing a long string of hex digits.

Other useful features (may have)

Augments Legacy Methods --- It may protect a less secure, legacy method in such a way that the combination of the wireless authentication method and legacy method meet the above requirements. This feature is useful in environments with legacy authentication systems that cannot quickly be replaced.

Fast Reauthentication --- It may provide a reauthentication mechanism that is less time and/or compute intensive than the legacy authentication. Of particular concern is enabling fast handoffs for mobile users. Since the time constraints on a handoff may be very tight, a reauthentication mechanism that takes few round trips or can be accomplished by a server in the service provider's domain rather than the user's home domain would be helpful. However, care should be taken that such reauthentication mechanisms provide strong security.

Authentication methods for wireless networks

Certificate based authentication methods

Today's 802.11 networks authenticate users according to the IEEE 802.1x standard. 802.1x specifies how to run the Extensible Authentication Protocol (EAP) directly over a link layer protocol. EAP is essentially a transport protocol that can be used by a variety of different authentication types known as EAP methods. EAP was standardized by the IETF in March 1998 for use over point-to-point network connections.

Among the EAP methods developed specifically for wireless networks are a family of methods based on public key certificates and the Transport Layer Security (TLS) protocol. These are EAP-TLS, EAP-TTLS, and PEAP. We will consider each of these in this section, and then consider another family of EAP methods, the strong password methods (sometimes known as Zero Knowledge Password Proof--ZKPP).

EAP-TLS

EAP-TLS uses the TLS public key certificate authentication mechanism within EAP to provide mutual authentication of client to server and server to client. With EAP-TLS, both the client and the server must be assigned a digital certificate signed by a Certificate Authority (CA) that they both trust.

Features of EAP-TLS include:

- Mutual authentication (server to client as well as client to server)
- Key exchange (to establish dynamic WEP or TKIP keys)
- Fragmentation and reassembly (of very long EAP messages necessitated by the size of the certificates, if needed)
- Fast reconnect (via TLS session resumption)

EAP-TTLS

The Tunneled TLS EAP method (EAP-TTLS) provides a sequence of attributes that are included in the message. By including a RADIUS EAP-Message attribute in the payload, EAP-TTLS can be made to provide the same functionality as PEAP (discussed below). If, however, a RADIUS Password or CHAP-Password attribute is encapsulated, TTLS can protect the legacy authentication mechanisms of RADIUS. When the TTLS server forwards RADIUS messages to the home server, it decapsulates the attributes protected by EAP-TTLS and inserts them directly into the forwarded message. Because this method is so similar to PEAP, it is being used less frequently.

PEAP

Like the competing standard TTLS, PEAP makes it possible to authenticate wireless LAN clients without requiring them to have certificates, simplifying the architecture of secure wireless LANs. Protected EAP (PEAP) adds a TLS layer on top of EAP in the same way as EAP-TTLS, but it then uses the resulting TLS session as a carrier to protect other legacy EAP methods. PEAP uses TLS to authenticate the server to the client but not the client to the server. This way, only the server is required to have a public key certificate; the client need not have one. The client and server exchange a sequence of

EAP messages encapsulated within TLS messages, and the TLS messages are authenticated and encrypted using TLS session keys negotiated by the client and the server.

PEAP provides the following services to the EAP methods it protects:

- Message authentication (Imposters may neither falsify nor insert EAP messages.)
- Message encryption (Imposters may neither read nor decipher the protected EAP messages.)
- Authentication of server to client (so that the protected method only needs to authenticate client to server)
- Key exchange (to establish dynamic WEP or TKIP keys)
- Fragmentation and reassembly (of very long EAP messages, if needed)
- Fast reconnect (via TLS session resumption)
- PEAP is especially useful as a mechanism to augment the security of legacy EAP methods that lack one or more of the above features.

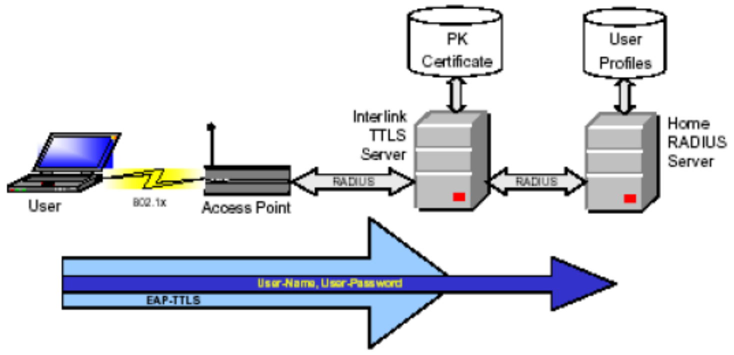


Figure 10. How a TTLS server interacts with a legacy RADIUS server

Microsoft PEAP

Microsoft PEAP supports client authentication by only MS-CHAP Version 2, which limits user databases to those that support MS-CHAP Version 2, such as Windows NT Domains and Active Directory.

To use Microsoft's PEAP, users must purchase individual certificates from a third-party certification authority (CA) to install on their IAS, and a certificate must be installed in the user's local computer certificate store. For wireless clients to validate the IAS certificate chain properly, the root CA certificate must be installed on each wireless client.

Windows XP, however, includes the root certificates of many third-parts CAs. If the IAS server certificates correspond to an included root CA certificate, no additional wireless client configuration is required. If users purchase IAS server certificates for which Windows XP does not include a corresponding root CA certificate, they must install the root CA certificate on each wireless client.

Cisco PEAP

Cisco PEAP supports client authentication by One-Time Password support (OTP) and logon passwords. This allows support for OTP databases from vendors such as RSA Security and Secure Computing Corporation, and also supports logon password databases like LDAP, Novell NDS, and Microsoft databases. In addition, the Cisco PEAP client can protect user name identities until the TLS encrypted tunnel is established. This provides additional assurance that user names are not being broadcast during the authentication phase.

Problems with certificate based methods

Despite the many advantages of certificate-based EAP types, there are some disadvantages as well.

Cost of Administration

The biggest down side to certificates is the cost of administration. All of the methods in this family require the authenticator to have a public key certificate signed by an authority that is recognized by the clients (the users' devices). This requires network administrators either to purchase server certificates from a commercial certificate authority (CA) or to acquire the software and expertise to create their own. Next, each device that will access the network must be configured to recognize the certificates of the authenticator and the CA. The EAP-TLS method requires all the user devices to have certificates as well. This significantly increases the cost of administration. Not only do certificates have to be created or purchased for each user device, but distribution can be a problem as well--there must be a method of securely installing the certificates on the user devices. Also, it can be difficult to maintain a Certificate Revocation List (CRL) so that the authenticator will know which certificates are good and which are not.

Lengthy Protocol Exchange

A second disadvantage of using a certificate-based EAP method is the number of sequential protocol exchanges (round trips) that are required between the user client and the authenticator in order to complete the authentication. For example, to authenticate a single user via EAP-MD5 protected by PEAP requires six round trips between the user station and the authenticator. Requiring a large number of protocol exchanges both lengthens the authentication delay for the user and uses more computing resources on the authenticator. Because the authentication delay is a particular problem for mobile users who must be reauthenticated when moving from one access point to another and who require a seamless handoff so as not to disrupt ongoing sessions, these methods all permit use of the TLS session resumption feature. This mitigates the handoff problem, but does not help the initial authentication.

Authenticates the Device Instead of the User or Requires a Smart Card

A third disadvantage is that the certificate must either be stored on the user device or on a smart card that the user carries. When certificates are stored on the user's device, it is the device that is authenticated rather than the individual user. In environments where the device cannot be sufficiently secured or where many individuals use the device, it is important to authenticate each individual user. A smart card is a way users can carry their certificates with them, but they are a source of inconvenience and require all the devices to have a card interface.

Password Authentication Methods

Although password authentication methods are more convenient than certificate-based methods, they still have vulnerabilities. They are specifically vulnerable to offline dictionary attacks, where an attacker can select guesses from a cracker's "dictionary" of possible passwords.

LEAP and Cisco CCX

LEAP is Cisco's Lightweight Extensible Authentication Protocol, and is based on mutual authentication, which means that both the user and the access point must be authenticated before access onto the corporate network is allowed. Mutual authentication protects against unauthorized (or "rogue") access points attempting to gain entry into the network. Cisco LEAP is based on a username/password scheme and is proprietary to Cisco access points. Cisco CCX (Cisco Compatible Extensions Program) provides assurance of compatibility between Cisco Aironet wireless infrastructure products and wireless client devices from third-party companies. This helps to maintain compatibility with Cisco features and protocols, including LEAP.

LEAP

With Cisco's LEAP, security keys change dynamically with every communications session, preventing an attacker from collecting the packets required to decode data. The new keys generated through LEAP use a shared secret key method between the user and the access point. Because LEAP is proprietary to Cisco, it can be used only with a Cisco access point. LEAP also adds another level of security to the network by authenticating all connections to the network before allowing traffic to pass to a wireless device. Using constantly changing secret keys coupled with user authentication provides additional security for wireless data.

Strong Password Authentication Methods

In response to the cost and inconvenience of using certificate-based authentication methods, security researchers have developed a whole new family of authentication methods based on the use of passwords, but addressing all the deficiencies of traditional password methods. We will use the term strong password to refer to this family.

The main benefit of the strong password methods is that two parties can prove to each other that they both know a secret without revealing that secret to a third party who may be listening in on the conversation. In fact, they neither reveal the secret nor make it easier for the attacker to discover the secret. Strong password methods achieve strong authentication by using a small, easily remembered password.

At the core of these methods is a Diffie-Hellman exchange. A Diffie-Hellman exchange permits two parties to create encryption keys in such a way that an observer watching the entire session will not be able to learn the keys. Diffie-Hellman exchanges take place between web browsers and online merchants, for example, in order to encrypt personal information such as credit card numbers. If the customer and merchant have never done business before, how are they to agree on an encryption key without third parties who may be eavesdropping on the session finding out what it is? Diffie-Hellman supplies the solution.

The Power of SPEKE

The SPEKE method uses a series of random-looking messages exchanged between devices. SPEKE modules perform computations with these messages, and then determine whether the password used at the other device was correct. When the passwords match, SPEKE puts out a shared key for each device.

To a third-party observer, SPEKE messages look like random numbers and cannot be used to verify any guesses as to what the password might be. SPEKE's additional power comes from the public key computations that are central to this method. There is no need for any long-lived public keys, private keys, or any sensitive data other than the password. SPEKE uses the Zero Knowledge Password Proof (ZKPP) authentication method to securely transmit passwords, which prevents revealing information to any participant unless they use the exact password in the protocol.

Because of this, SPEKE makes password-based authentication stronger and safer. With SPEKE, even a small or poorly chosen password receives greater protection from attack. Other security characteristics of SPEKE include:

- Strong, unlimited length of key can be negotiated
- Protection from off-line attacks that crack hash-based challenge/response methods
- Client and server are authenticated simultaneously
- No other security infrastructure requirements
- No client or server certificates are required
- Complete benefits of modern cryptography using an ordinary small password

Ease of Use

To implement SPEKE, users perform a one-time setup when installing the device driver or contacting an access point for the first time. There is no need for additional infrastructure (unlike TLS and other 802.1x authentication alternatives) to get the same level of authentication, and can be built into simple wireless access point devices.

SPEKE vs. LEAP

Cisco LEAP (Lightweight Extensible Authentication Protocol) is a proprietary protocol that may be used with Cisco access points only. It is a derivative of EAP, providing mutual authentication between client and server, but is proprietary at the access point level of the network. SPEKE is access point independent and will work with any 802.1x compliant access point. This provides maximum flexibility for mixed networks or networks that do not exclusively use Cisco WLAN infrastructure.

SPEKE vs. PEAP

Protected EAP (PEAP) provides support for one-time token authentication, password change and expire support, and database extensibility to support LDAP/NDS directories. PEAP encrypts the conversation between the EAP client and the server, and security is maintained by using a TLS channel.

Mutual authentication is required between the EAP client and the server. SPEKE, however, does not require using tokens or certificates, and provides simultaneous authentication. Passwords are exchanged securely, without revealing information to third parties, and there is no need for a TLS channel.

Conclusion

Securing your wireless network provides tremendous cost savings, productivity benefits, and a competitive market advantage. It is not a question of whether enterprises will require wireless network security, but when. Choosing the highest level of security available is a good investment, because security breaches can be a significant expense. Most attacks go unnoticed, and enterprises can be vulnerable to damages. Security breaches such as stolen information, corrupt data, and network downtime can be expensive. They can also result in consequential damages, such as those resulting from increasing a competitor's position or market share at the expense of your future revenues and profitability. The cost can be both significant and recurring.

Authentication in Ad-hoc networks

An ad-hoc (or "spontaneous") network [10] is a local area network or other small network, especially one with wireless or temporary plug-in connections, in which some of the network devices are part of the network only for the duration of a communications session or, in the case of mobile or portable devices, while in some close proximity to the rest of the network. In Latin, ad hoc literally means "for this," further meaning "for this purpose only," and thus usually temporary. The term has been applied to future office or home networks in which new devices can be quickly added, using, for example, the proposed Bluetooth technology in which devices communicate with the computer and perhaps other devices using wireless transmission. Other related work can be found in [24-32].

One vendor offers an ad-hoc network technology that allows people to come to a conference room and, using infrared transmission or radio frequency (RF) wireless signals, join their notebook computers with other conferees to a local network with shared data and printing resources. Each user has a unique network address that is immediately recognized as part of the network. The technology would also include remote users and hybrid wireless/wire connections.

Jini is an approach to instant recognition of new devices in a network that would seem to make it easier to have an ad-hoc network.

Mobile ad-hoc network

A mobile ad-hoc network (MANET) is a self-configuring network of mobile routers (and associated hosts) connected by wireless links—the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet. Minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural or human-induced disasters, military conflicts, emergency medical situations etc.

Ad hoc networking enables wireless devices to network with on another, as needed, even when access to the Internet is unavailable. It enables a wide range of powerful applications, from instant conferencing between notebook PC users to emergency and military services that must perform in the harshest conditions, wireless communications without routers, base stations, or Internet Service Providers. An ad-hoc network might consist of several home-computing devices, plus a notebook computer that must exist on home and office networks without extra administrative work. (Key applications - conferencing, home networking, emergency services, Personal Area Networks, *Bluetooth*, and more.) Addressing the key challenges of ad hoc networking - resource management, scalability, and especially security is of the essence.

The earliest MANETs were called "packet radio" networks, and were sponsored by DARPA in the early 1970s. BBN Technologies and SRI International designed, built, and experimented with these earliest systems. Experimenters included Jerry Burchfiel, Robert Kahn, and Ray Tomlinson of later TENEX, Internet and email fame. It is interesting to note that these early packet radio systems predated the Internet, and indeed were part of the motivation of the original Internet Protocol suite. Later DARPA experiments included the Survivable Radio Network (SURAN) project, which took place in the 1980s. Another third wave of academic activity started in the mid 1990s with the advent of inexpensive 802.11 radio cards for personal computers. Current MANETs are designed primarily for military utility; examples include JTRS and NTDR.

The popular IEEE 802.11 ("Wi-Fi") wireless protocol incorporates an ad-hoc networking system when no wireless access points are present, although it would be considered a very low-grade ad-hoc protocol by specialists in the field. The IEEE 802.11 system only handles traffic within a local "cloud" of wireless devices. Each node transmits and receives data, but does not route anything between the network's systems. However, higher-level protocols can be used to aggregate various IEEE 802.11 ad-hoc networks into MANETs.

Authentication issue

In most applications where security matters, authenticity is an essential prerequisite. Effective authentication is a necessary measure to guarantee the authenticity of communication networks. It determines that the user who wants to enter the systems and share resources with other users is the valid one who he/she claims to be. In ad hoc wireless communication, freshness, availability, integrity and confidentiality are inherently included in the authentication issue.

Wireless communication in ad hoc networking is highly vulnerable to security threats: an advanced security problem is the absence of an online server. When a node comes within range, we cannot connect to an authentication server or to check the validity of an exhibited certificate. The information over the air can be intercepted by anyone with a radio receiver. Many methods have been proposed, trying to solve this challenging problem. However, up to now, there is no perfect solution yet and the problem is still unsolved.

Today, authentication in peer-to-peer communication over ad-hoc networks is done by the owner of wireless device granting access to specific request. In the common case, where two or more devices are to be used in the most basic ad hoc networks, there is usually the presence human, which intervenes like a base station. This kind of authentication over ad-hoc networks needs the presence of human, at least at initial stage and then is not done automatically.

"Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary" --- Ross Anderson

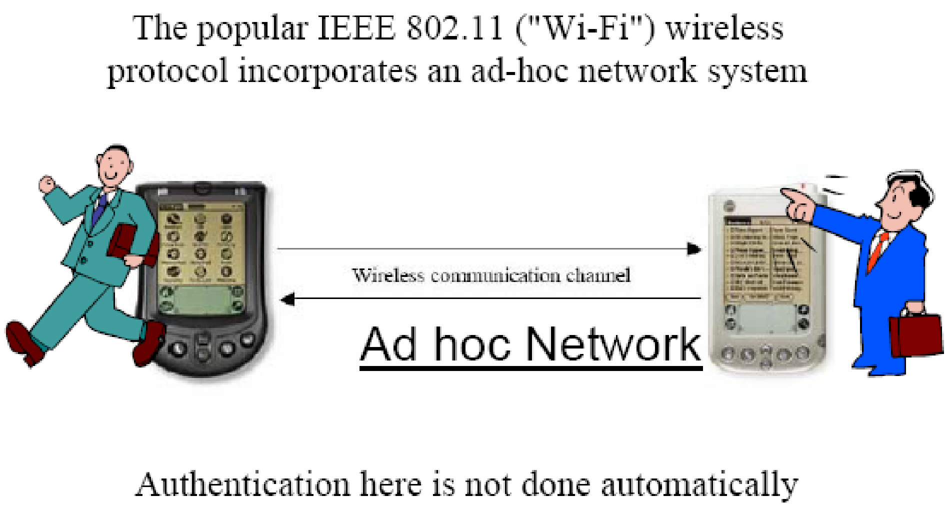

The popular IEEE 802.11 ("Wi-Fi") wireless protocol incorporates an ad-hoc network system
Wireless communication channel
Ad hoc Network
Authentication here is not done automatically

Figure 11. Ad hoc network

Possible solutions: Pre-boot biometric authentication

In mobile ad-hoc wireless network, network is becoming the most important resource that need to be protected very safely (which is different from what we think traditionally). This is because of the low cost of wireless devices and their limited storage, which render wireless devices obtain and exchange information most from the network. To adapt this situation, we propose a possible solution to the authentication over mobile ad-hoc networks: pre-boot biometric authentication in this section.

The system we are exploring, as shown in Fig. 12, would work along these lines: There is a central biometric based authentication server in the network which can authenticate all the users from the beginning (by checking users' primary biometric factor, for example fingerprints). This server issues an identical session key to each user once his/her identity is verified. In the meanwhile, the second biometric factor of each valid user (such as the facial image of the user) can be stored in his/her wireless device for further use. Operating systems are loaded upon authentication. (In this stage, authentication gives devices access to the network before the boot process starts, which gives devices the option to load the operating system either from their local storage or by downloading an OS image from the network.) Users with the same session key can communicate with each other during the valid period of the session key. This key will be invalidated at an arbitrary but configurable timeout. Renew of a session based on the checking of users' second biometric factors: wireless device takes the picture of current user and then compare it with the one that has already been stored there to make sure that current user is still the same as the one who was verified. In fact, the authentication in this stage is not device-based, but user-based, which can prevent any person other than the owner of the wireless device from using and getting sensitive information illegally (authentication in this stage is important since it is easy to steal or take over wireless device from its owner temporarily in many cases).

Authentication over mobile ad-hoc networks

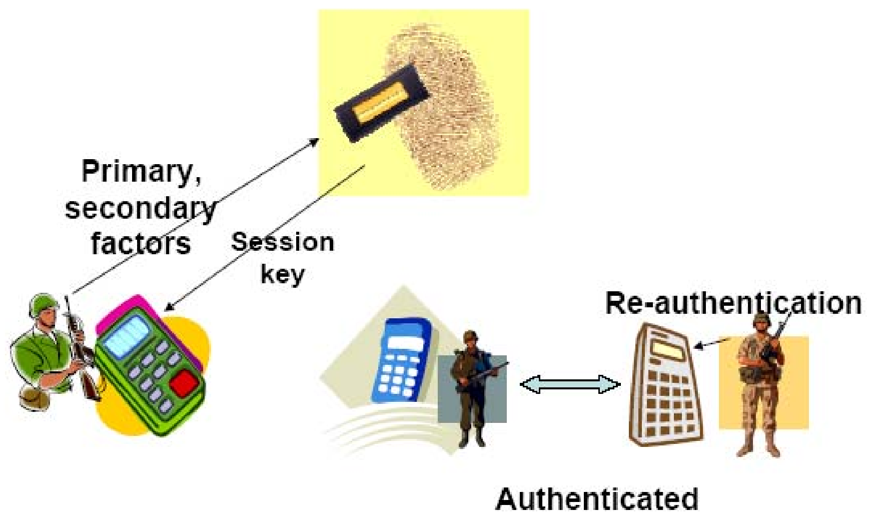


Figure 12. Possible solution: Pre-boot Biometric Authentication

In this scenario, the network (peer devices and static servers) broadcasts encrypted information, which every device with a session key will be able to decrypt. For the network to estimate the number of users at any given point, it can resort to a "show of hands" strategy by broadcasting a request for acknowledgement.

The reasons and advantages of proposing this method are as follows: Pre-boot authentication is suitable here because authentication systems of this kind are OS-independent, which is very necessary for heterogeneous wireless networks. Also in our possible solution, the actual wireless communication devices are used in single-user mode. This requirement is not difficult to be satisfied because the devices in ad-hoc networks are normally small (such as cell phone and palmtop) and they don't run multi-user operating system. Moreover, biometric-based authentication is not hard to implement technically nowadays and can be enhanced to meet the requirements for wireless authentication.

There are some aspects we are going to explore in the future research. First, in the proposal, the biometric sensor is connected to the authentication sever, but not to the wireless device. This is because of power-consuming issue of the wireless device, not to mention the impossibility to guarantee physical security of the sensor in an ad-hoc wireless network setting. However, in the wired version, the sensor is connected to the device (computers), not to sever. This difference makes it necessary to propose a new solution in authentication over ad-hoc networks. Second, the issue of protecting the communication channels, the possible threats on our proposed system and newly appeared authentication requirements in mobile ad-hoc networks are not discussed with more details in our proposed solution and need further exploration in the future work. Last, the proposed solution we introduced in this section only addresses some specific situation (a group that can gather together first and then scatter around) and is not a standard solution for more general cases. More common solutions are expected for general use in near future.

User Authentication

Many organizations struggle with the problem of authentication; how users prove their identity in order to get access to applications or other IT resources. How strong or how secure does the authentication process need to be? Is the combination of a username and password sufficient? If not, should a token device – like SecurID or digital certificate be required? None of these solutions is without its problems. In this section, we will discuss some possible solutions especially for the user authentication of device in ad hoc network.

Password protection in BIOS

Passwords have been the predominant method of authentication in client/server environment and still today they are the most widely used form of authentication: they represent the first and best example of a static security approach. To prove their claimed identity users provide an identifier, a typed in word or phrase (usually referred as an ID) along with a password and once they are logged they could interact with the system freely (according to their system's rights). Password authentication doesn't require additional hardware device since authentication of this type is in general simple and achieved through software operations. Of course password-systems do not require much processing power and they aren't time-consuming.

One type of password authentication is done prior to the booting of the OS. The so called Pre-Boot-Authentication can protect against Trojan horses that patch parts of the OS to spy out passwords or other data. To avoid the user having to use multiple passwords Pre-Boot-Authentication has to be combined with a single logon solution.

For a modern IBM-compatible PC [5], it is generally possible to password-protect the BIOS setup and to prevent booting from a floppy disk; therefore it is possible to ensure pretty strictly that the machine can only be booted over the network and, therefore, only after an authenticated logon. This is so because the network-card boot-ROM code is executed during the BIOS initialization, i.e. before control is passed to code which is loaded from a vulnerable source (e.g. the hard or floppy disk).

Currently, most of the operating systems, such as Windows platforms, use post-boot authentication strategy. It is unable to provide boot protection, as their user-based security works only after starting the operating system. In such a way, since the authentication process has to go through a running workstation, we will have to take the risk of hidden Trojan horse programs that can launch the following attacks:

- Sniffing attack: using a sniffer program that copies information out of the authentication device input buffer after the user typed in his/her password.
- Replay attack: an attempt by an unauthorized third party to record an over the air message; this may be used later in a process to fool the receiver.

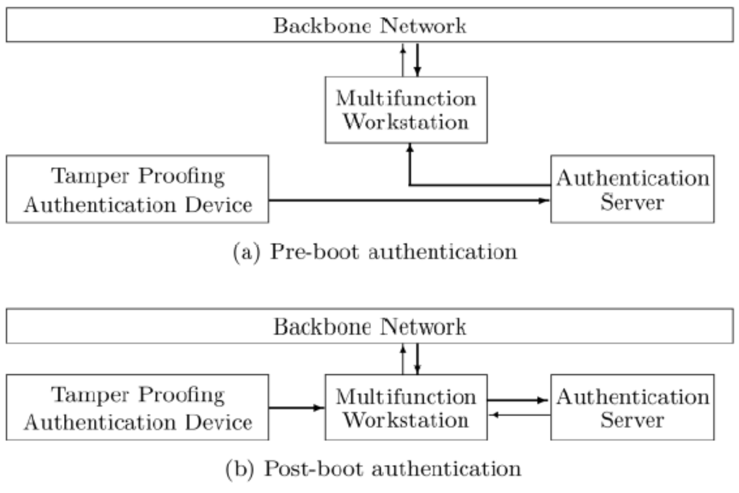

Backbone Network
Multifunction Workstation
Tamper Proofing Authentication Device
Authentication Server
(a) Pre-boot authentication
Backbone Network
Tamper Proofing Authentication Device
Multifunction Workstation
Authentication Server
(b) Post-boot authentication

Figure 13. Pre-boot and post-boot authentications

However, it should be noted that BIOS passwords are vulnerable as there are several programs readily available which can read the BIOS password from the CMOS RAM (and possibly decrypt it, if it is encrypted) - of course, prior authentication is required to get to a state where such a program can be run. It is also very easy to open up a PC's case (there are generally only 4 screws to be undone) and to discharge or disconnect the battery which is keeping the CMOS memory, in which the BIOS password and the basic configuration are stored, alive - physical precautions, such as loop alarms, can be used to discourage such attacks.

Pre-boot Biometric Authentication

In information technology, biometric authentication refers to technologies that measure and analyzes human physical and behavioral characteristics for authentication purposes. Biometrics is essentially an automated process of the recognition of individuals through a physiological or behavioral characteristic. There are many different types of biometric, although the most commonly used are physiological ones: fingerprint and hand geometry, facial recognition and iris scanning [5].

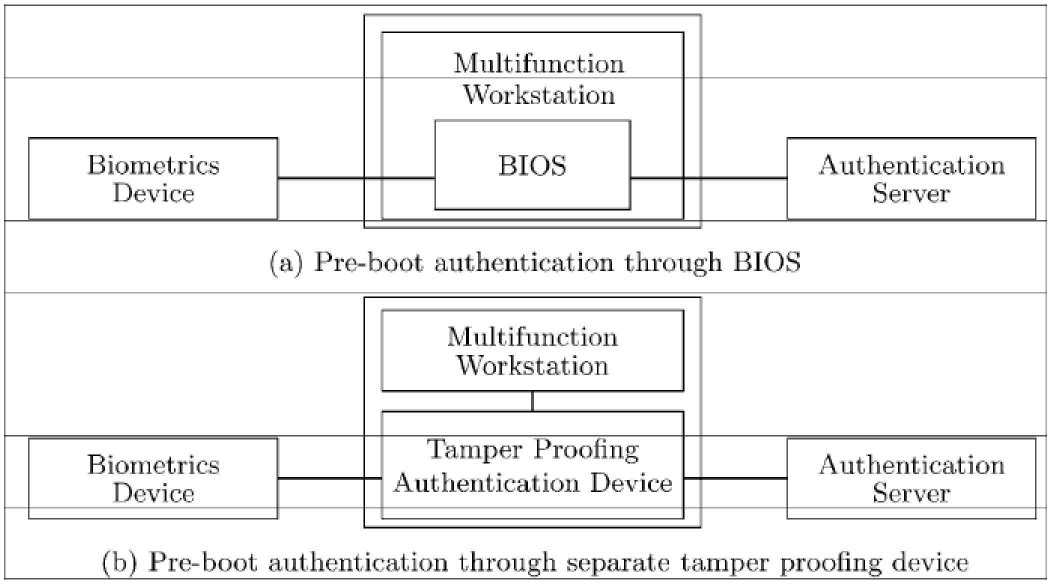

Multifunction Workstation
BIOS
Biometrics Device
Authentication Server
(a) Pre-boot authentication through BIOS
Multifunction Workstation
Tamper Proofing Authentication Device
Biometrics Device
Authentication Server
(b) Pre-boot authentication through separate tamper proofing device

Figure 14. Different pre-boot authentication schemes

The pre-boot authentication scheme requires a user to be authenticated before booting the operating system. Without correct authentication, the information system cannot be started. Therefore, it will prohibit unauthorized assess to the system and prevent hackers from gaining control. There are several ways to implement pre-boot authentication functionality with different considerations on degrees of difficulty, security level and cost. The major difference is either using the existing workstation functionalities or building a separate authentication device, as shown in above Figure.

There are basically 4 possibilities where to store the template: in a card, in the central database on a server, on a workstation or directly in an authentication terminal. The storage in an authentication terminal cannot be used for large-scale systems, in such a case only the first two possibilities are applicable. If privacy issues need to be considered then the storage on a card (magnetic stripe, smart or 2D bar) has an advantage, because in this case no biometric data must be stored (and potentially misused) in a central database.

Because typical biometric authentication products need the computational power of the multifunction workstation to authenticate the user, the pre-boot authentication can be handled at the basic input/output system (BIOS) level before the operating system is read from the hard disk (Fig.13 (a)). The advantage of this design lies in that it uses as much the workstation's resources as possible to reduce the cost. The disadvantage is that a Trojan horse on the workstation may affect the security of the communication. If there is no cryptographic module on the biometrics capture device, there is no way to send an encrypted biometric trait to the authentication server. This will be a critical problem.

A higher security pre-boot authentication scheme can be achieved by adding a separate tamper proofing authentication device that will generate all of the information required to boot the system after the user's identity is successfully verified by the authentication server (Fig.13 (b)). The advantage of adding a separate authentication device is that the network interface card of the workstation cannot be activated before the user is authenticated, which will reduce the system vulnerability. In such a device, the minimum requirements include operating system, hard drive controller, network interface card and cryptographic module if there is none on the biometrics capture device. Therefore, the major disadvantage is a higher cost.

Our concern is now to look at the use of biometrics technology to determine how secure it might be authenticating users with these techniques and how the users' job functions or roles would impact an authentication process or protocol.

Using biometrics on unsecured networks

Main issues

The use of biometric information introduces new privacy concerns over the huge challenge of granting secure remote authentication over unsecured networks. Today Web growth implies new schemes that can't be easily addressed with old practices but neither with new ones. The growing complexity of the WWW implies on-line 24hrs threatening of valuable data stored on your system and possibly a greater quantity of potential hackers interested in penetrating your network. The use of biometric does not represent in itself the solution to these concerns but introduces new possibilities in implementing a safe authentication protocol. We should wisely consider that new methodologies bring along with them new undiscovered threatens and, as we reported in the previous paragraphs, biometrics is not rid of them.

The main problem of remote authentication is the use of a third party channel that is totally untrustworthy and usually unsecured. No matter what kind of information our IP packet is conveying over the WEB this could be discarded, intercepted and replaced whether strong defenses are undertaken on both side of the communication. Many products promise the "construction" of a solid channel via the web to safely interchange information between two points of the web but as the complexity of the security design grows even the possible break points number increase. In a generic biometric remote authentication scheme those are:

- The biometric reader: someway we must prove that the sample he submits out of its box is genuine and is the result of a live-scan.
- The link between the biometric reader and the local host: even this link could be eavesdropped and we must take care of this eventuality.
- The local host workstation is another possible break point.
- The channel between the local host and the server presenting all the generic risks of an open channel: interleaving and man in the middle attacks.
- The device storing the templates, no matter what is, represents a valuable data vault for hackers and it must be properly secured.
- If the biometric match test takes place on the local host we must implement a protocol according which the server could trust the local host first and the communication channel as well then.
- On client's side is probable the request for server authentication.

Each of these points addresses a wide range of problems and concerns quite huge to solve and thus they represent a harder step to overcome if mixed all together. So step by step we should try to explain how a solution could be reached.

The Trouble with Biometrics

Biometric authentication is an effective method identifying who a user is. But, the same with the password method, biometric authentication also meets many problems such as replay attack, identity stolen and information sniffing on communication channels. More details will be discussed in this section.

An easy way to implement biometric authentication is to collect biometric data from the user and then transfer it to the authentication server, which will compare the biometric data against its user database, just like the traditional password authentication. Authentication models of this kind are widely used nowadays, though lots of problems will occur during the process.

First, what should a user do if his/her biometric information has been stolen? When passwords are stolen, you can change your passwords. When the card numbers are stolen, you can cancel this card and apply for a new one. But, what can we do when our biometrics is stolen? It is almost impossible to get a new one, except that you're willing to have an organ transplant.

Second, biometric data are not perfect constants. Due to imprecision in measures, temperature variations, biological factors… the same data measured from the same person at different times will not give perfectly equal results: the difference will be small, but probably not zero. Biometrics based authentication takes this into account using something we could define as loose equality: two samples are considered equal if their difference is so small that they cannot possibly come from different persons. However, this loose equality induces some problems with biometrics. A biometrics system does not give a basic authentication mechanism with a yes or no answer, but rather something like: with a confidence level of 96%, I have authenticated the user. More formally, the performance of a biometrics system is commonly given with his False Rejection Rate (FRR) and his False Acceptation Rate (FAR). Those rates indicate the probability of rejecting a correct user (FRR) and the probability of accepting a false user (FAR). False acceptation rate is more important because even without pirates, the system will sometimes wrongly identify somebody.

In the process of password authentication, hash functions are often used to encrypt the user's password, which can avoid most replay attacks in many cases. But in biometric authentication, the introduction of a hash function completely destroys loose equality because a hash function will put the stress on small differences between similar messages, producing completely different digests. Thus, the question "do these two digests correspond to similar original messages?" is impossible to answer. This specificity of biometrics rules out hash functions from the candidate building blocks: to be able to perform biometrics-based authentication, it must be possible to recover, at some time in the protocol, the input and reference values "in clear".

Moreover, there are several vulnerable points that can be attacked during the process of remote biometric authentication, such as transmission and replay attacks on the biometric device-to-computer link and the client/server network link (this can be weakened by data encryption); spoofing attack on the biometric sensors and template attack on the database of authentication server.

As we have shown here, the target of applying biometric authentication is to avoid the disadvantages of password-based authentication. But in the meanwhile, some new threats arise when using this kind of model, which tell us that biometric systems can be very useful to improve the security but they must be used carefully because of their own specific properties. Till now, some other solutions have been proposed to solve those problems: a challenge/response protocol keeps the biometric data from being stolen at intermediate points, and authenticating only to a crypto card addresses all the issues with biometric being stolen or replayed, more details about those methods can be found in [7]. Currently, more and more interest is concentrated on multi-modal biometric systems since they can either increase the security level or improve the user convenience of biometric systems, see [14]. However, all those solutions have theirs own shortcomings and more effective methods are still expected to make up those problems.

Lack or ignorance of standards can also present problems. At present, two similar biometric systems from two different vendors are not likely to interoperate. Such issue must be solved before secure and reliable biometric systems are widely deployed.

Information Privacy

Biometric systems can violate user privacy. Biometric characteristics are sensitive data contain personal information. Users may find some biometric systems intrusive or personal invasive. Use of biometric systems also implies loss of anonymity. Whereas you can have multiple identities when authentication methods are based on something you know or have, biometric systems link all user actions to a single identity.

Classification of biometric authentication on security levels

In this part, we classify the biometric authentication into four different levels with the increment of security, based on their liveness testing methods, tamper resistance and communication schemes, etc. Proper level can be chosen according to the purpose of authentication systems, the threats they meet and available fund. More details about this can be found in [4].

Biometric authentication systems which only provide very limited protections fall into level 1, very simple systems. This kind of systems doesn't offer liveness test, tamper resistance or protection of communication channels. Thus they are easily to be cheated and subject to simple attacks.

Biometric authentication systems that offer mutual authentication of particular components and encrypted communication belong to the level 2, simple systems. The systems in this level don't provide liveness testing or tamper resistance and remain relatively cheap. They are still vulnerable to many attacks such as faked biometric characteristics and intent substitution of biometric input devices, etc.

Biometric authentication systems of level 3 provide some liveness tests and moderate tamper resistance against moderate attacks. The communication of this level must be encrypted and authenticated. Biometric authentication systems on this level offer a certain level of security. But they will still be cheated by some advanced faked biometric characteristics.

Biometric authentication systems of the highest level, level 4, provide more than one liveness test method and advanced tamper resistance to against tampering attacks of higher levels. Multiple biometric techniques are involved in biometric authentication of this level. Communication among particular components is required to be encrypted and mutually authenticated.

We conclude these four categories of biometric authentication system in Table 1. Biometric systems we are talking about here are supposed to resist professional and well-funded attacks. However, all we all know, nothing is bullet proof, any technology can be broken by someone, in some way, with some effort.

Level	Liveness testing	Tamper-resistance	Secure Comm.
1	No	No	No
2	No	No	Yes
3	Yes	Moderate	Yes
4	Multiple	Advanced	Yes

Table 1. Classification proposal on biometric authentication

Conclusions

Biometric authentication is a great way of authenticating people to computer applications. Most current implementations, however, leave a lot to be desired. What's scary is that some people assume that because biometrics is nearly infallible for identification, applications that use biometrics for authentication are automatically safe and secure. Many current biometric systems offer only limited security. User authentication can succeed only when the biometric characteristics are fresh and collected from the user being authenticated, implying a trusted biometric input device. The system should verify the device’s authenticity (unless the device and link are physically secure) and check the user’s liveness. Input devices should be either tamper-resistant or under human supervision. There's also the question of whether authenticating a person is appropriate for a particular application (think Swiss banking privacy laws).

Two alternatives to the current password-like biometric authentication model: challenge/response and authenticating to a crypto card have been proposed in [7]. The challenge/response scheme has issues with the final authentication database/process being subverted, while the crypto-card scheme has issues with the client software being subverted.

Which one is safer will depend on the nature of the application. Both are superior to any password-like scheme. They're light-years ahead of current password-like implementations, which feature a distinct lack of attention to proper encryption of the data stream and crypto-authentication of the devices.

Very promising are solutions where the cryptographic functions as well as the biometric matching, the feature extraction and the biometric sensor are all integrated in one (ideally also tamper-resistant) device. Such devices provide a very high protection of the secret/private key as the biometric data as well as the secret/private key will never have to leave the secure device.

We conclude our opinions on biometric authentication in this part: Different biometric samples of the same person will never be same; biometric systems make errors and biometric data are not secret; the role of the input device is crucial, and this device must be trusted or well secured; the biometric system should check user's liveness, biometrics is good for user authentication, but cannot be used to authenticate data or computers.

Biometrics can be used for dozens of applications outside the scope of computer security. Facial recognition systems are often deployed at frequently visited places to search for criminals. Fingerprint systems (AFIS) are used to find an offender according to trails left on the crime spot. Infrared thermographs can point out people under influence of various drugs (different drugs react in different ways). Biometric systems successfully used in non-authenticating applications may but also need not be successfully used in authenticating applications.

References

1. “Understanding the Linux boot process”, *http://www.kevinboone.com/boot.html*

2. Bryony Pomeroy, Kevin Shorter, “Authentication technologies”, *www.QinetiQ.com\perspectives*

3. Kevin Shorter, Lan Nice, “Biometrics and Security—an introduction”, *www.QinetiQ.com\perspectives*

4. Václav Matyás Jr., Zdenek Riha, “Biometric authentication –security and usability”, *Advanced Comm. And Multimedia Security*, Kliuwer Academic, 2002, pp. 227-239.

5. Qinghan Xiao, “Biometric user authentication for heightened information security”, Lecture Notes in Computer Science, 2004 – Springer, pp. 708-715.

6. Gaël Hachez, François Koeune and Jean-Jacques Quisquater, “Biometrics, access control, smart-cards: a not so simple combination”, *Proc. 4th IFIP WG8.8 Working Conference on Smart Card Research and Advanced Applications (CARDIS) 2000.*

7. Christopher Calabbrese, “The trouble with biometrics”, *login, Vol. 24, no. 4, 1999*, pp. 56-61.

8. David Grawrock, “Building trust and privacy into open PC systems”, *Intel Developer Update Magazine, Nov. 2000*, pp. 1-6.

9. C. Viti, S. Bistarelli, “Study and development of a remote biometric authentication protocol”, *ACM D.4.6 Security and Protection, Internal Note.*

10. A. O. Salako, “Authentication in ad hoc networking”, *Proceedings of London Communications Symposium 2002.*

11. *“EAP methods for 802.11 wireless* LAN security”, http://www.oec.org.

12. Santa Monica, “Can biometrics help the Army solv**e** an identity crisis?” *Arroyo Center. Army Research Division, 2001.*

13. Smith, R. E. *Authentication: From Passwords to Public Key,.* Addison-Wesley, Upper Saddle River, NJ 2001.

14. Qinghan Xiao, “Security issue in Biometric authentication”, *Proceedings of the 2005 IEEE, Workshop on information assurance and security,* United States Military Academy, West Point, NY, pp. 8-13.

15. K. Palmgren, “Biometric authentication, an introduction,” *SecurityDocs,* 9 February *2005,* http://www.securitydocs.c0milibrary/3003

16. P. Reid, Biometrics for network security. Upper Saddle River, New Jersey, Prentice Hall PTR, 2004.

17. J. Ortega-Garcia, J. Bigun, D. Reynolds and J. Gonzalez-Rodriguez, "Authentication gets personal with biometrics," *IEEE Signal Processing Magazine,* vol. 21, Issue 2, pp. 50-62, March 2004.

18. E41 J. Leyden, "Gummi bears defeat fingerprint sensors," The Register, 16 May 2002, http://www. theregister.com/2002/05/16/gummi~bears~defeat-fingerprint-sensors

19. L. Thalheim, J. Krissler and P. Ziegler, "Body check: Biometric access protection devices and their programs put to the test," C'T Magazine, http://www.heise.de/ct/englisWO2/11 /I 14/, May 2002.

20. D**.** Kingstey, "Fingerprint security ea**sy** to fool," ***News in Science, 20*** June 2002, http:/lwww.abc.net.au/science/news/stories/s585792.htm

21. International Biometric Group, "Liveness detection in biometric systems," **http://www.biometricgroup.comireports/public/reports/li**veness.html

22. R. K. Rowe, "A multispectral sensor for fingerprint spoof detection," *Sensors Magazine,* January 2005, http://www.sensorsmag.comiarticles/0105/25/

23. A. K. Jain and A. Ross, "Multibiometric systems," *Communications of the ACM,* Special h u e on Multimodal Interfaces, Vol. 47, No. I, pp. 34- *40,* January 2004.

24. Mark Norris, *Mobile IP Technology for M-business*, 2001, Artech House; ISBN: 1580533019.

25. James D. Solomon, *Mobile IP the Internet Unplugged*, 1998, Prentice Hall; ISBN: 0138562466.

26. Charles E. Perkins, *Ad Hoc Networking*, 2001, Addison-Wesley, London; ISBN: 0-201-30976-9.

27. Frank Stajano, *Security for Ubiquitous Computing*, 2002, John Wiley and Sons, Ltd, West Sussex, England, ISBN: 0-470-84493-0.

28. Frank Stajano, *The resurrecting duckling: What Next?* , 2000. http://www-lce.eng.cam.ac.uk/~fms27/papers/duckling-what-next.pdf

29. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar. *SPINS: Security Protocols for Sensor Networks*. Mobicom 2001. http://www.millennium.berkeley.edu/tinyos

30. David J. Wheeler and Roger M. Needham, *TEA, a Tiny Encryption Algorithm,* 1995 and *TEA Extension*, 1996, Computer Laboratory Cambridge University

http://www.ftp.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html
http://www.cl.cam.ac.uk/ftp/users/djw3/xtea.ps

31. Bruce Schnier, Applied Cryptography: protocols, algorithms and source code in C, 2ed, 1996, Wiley, ISBN: 0-471-11709-9.

32. Ross Anderson, *Security Engineering - a Guide to Building Dependable Distributed Systems*, 2001, John Willey & Sons, Inc., New York; ISBN: 0-471-38922-6.

Annexes

The PC Boot Process - Windows XP

The flow chart below shows us how Windows XP to boot up the computer step by step.

Power supply switched on.	➡	The power supply performs a self-test. When all voltages and current levels are acceptable, the supply indicates that the power is stable and sends the Power Good signal to the processor. The time from switch-on to Power Good is usually between .1 and .5 seconds.
⬇		
The microprocessor timer chip receives the Power Good signal.	➡	With the arrival of the Power Good signal the timer chip stops sending reset signals to the processor allowing the CPU to begin operations.
⬇		
The CPU starts executing the ROM BIOS code.	➡	The CPU loads the ROM BIOS starting at ROM memory address FFFF:0000 which is only 16 bytes from the top of ROM memory. As such it contains only a JMP (jump) instruction that points to the actual address of the ROM BIOS code.
⬇		
The ROM BIOS performs a basic test of central hardware to verify basic functionality.	➡	Any errors that occur at this point in the boot process will be reported by means of 'beep-codes' because the video subsystem has not yet been initialized.
⬇		
The BIOS searches for adapters that may need to load their own ROM BIOS routines.	➡	Video adapters provide the most common source of adapter ROM BIOS. The start-up BIOS routines scan memory addresses C000:0000 through C780:0000 to find video ROM. An error loading any adapter ROM generates an error such as: XXXX ROM Error

where XXXX represents the segment address of the failed module.

↓

The ROM BIOS checks to see if this is a 'cold-start' or a 'warm-start' →

To determine whether this is a warm-start or a cold start the ROM BIOS startup routines check the value of two bytes located at memory location 0000:0472. Any value other than 1234h indicates that this is a cold-start.

↓

If this is a cold-start the ROM BIOS executes a full POST (Power On Self Test). If this is a warm-start the memory test portion of the POST is switched off. →

The POST can be broken down into three components:
The Video Test initializes the video adapter, tests the video card and video memory, and displays configuration information or any errors.
The BIOS Identification displays the BIOS version, manufacturer, and date.
The Memory Test tests the memory chips and displays a running sum of installed memory.

↓

Errors the occur during the POST can be classified as either 'fatal' or 'non-fatal'. A non-fatal error will typically display an error message on screen and allow the system to continue the boot process. A fatal error, on the other hand, stops the process of booting the computer and is generally signaled by a series of beep-codes.

The BIOS locates and reads the configuration information stored in CMOS. →

CMOS (which stands for Complementary Metal-Oxide Semiconductor) is a small area of memory (64 bytes) which is maintained by the current of a small battery attached to the motherboard. Most importantly for the ROM BIOS startup routines CMOS indicates the order in which drives should be examined for an operating systems - floppy first, CD-Rom first, or fixed disk first.

Fixed Disk ↓

If the first bootable disk is a fixed disk the BIOS examines the very first sector of the disk for a Master Boot Record (MBR). For a floppy the BIOS looks for a Boot Record in the very first sector.

On a fixed disk the Master Boot Record occupies the very first sector at cylinder 0, head 0, sector 1. It is 512 bytes in size. If this sector is found it is loaded into memory at address 0000:7C00 and tested for a valid signature. A valid signature would be the value 55AAh in the last two bytes. Lacking an MBR or a valid signature the boot process halts with an error message which might read:

NO ROM BASIC - SYSTEM HALTED

A Master Boot Record is made up of two parts - the partition table which describes the layout of the fixed disk and the partition loader code which includes instructions for continuing the boot process.

MBR

With a valid MBR loaded into memory the BIOS transfers control of the boot process to the partition loader code that takes up most of the 512 bytes of the MBR.

The process of installing multiple operating systems on a single PC usually involves replacing the original partition loader code with a Boot Loader program that allows the user to select the specific fixed disk to load in the next step of the process

Partition Table

The partition loader (or Boot Loader) examines the partition table for a partition marked as active. The partition loader then searches the very first sector of that partition for a Boot Record.

The Boot Record is also 512 bytes and contains a table that describes the characteristics of the partition (number of bytes per sectors, number of sectors per cluster, etc.) and also the jump code that locates the first of the operating system files (IO.SYS in DOS).

Operating System

Boot Record

The active partition's boot record is checked for a valid boot signature and if found the boot sector code is executed as a program.

The loading of Windows XP is controlled by the file NTLDR which is a hidden, system file that resides in the root directory of the system partition. NTLDR will load XP in four stages:

1) Initial Boot Loader Phase
2) Operating System selection
3) Hardware Detection
4) Configuration Selection

	↓		
NTLDR Initial Phase	During the initial phase NTLDR switches the processor from real-mode to protected mode which places the processor in 32-bit memory mode and turns memory paging on. It then loads the appropriate mini-file system drivers to allow NTLDR to load files from a partition formatted with any of the files systems supported by XP.	→	Windows XP supports partitions formatted with either the FAT-16, FAT-32, or NTFS file system.
	↓		
NTLDR OS Selection BOOT.INI	If the file BOOT.INI is located in the root directory NTLDR will read it's contents into memory. If BOOT.INI contains entries for more than one operating system NTLDR will stop the boot sequence at this point, display a menu of choices, and wait for a specified period of time for the user to make a selection.	→	If the file BOOT.INI is not found in the root directory NTLDR will continue the boot sequence and attempt to load XP from the first partition of the first disk, typically C:\.
	↓		
F8	Assuming that the operating system being loaded is Windows NT, 2000, or XP pressing F8 at this stage of the boot sequence to display various boot options including "Safe Mode" and "Last Known Good Configuration"	→	After each successful boot sequence XP makes a copy of the current combination of driver and system settings and stores it as the Last Known Good Configuration. This collection of settings can be used to boot the system subsequently if the installation of some new device has caused a boot failure.
	↓		
NTLDR Hardware Detection	If the selected operating system is XP, NTLDR will continue the boot process by locating and loading the DOS based NTDETECT.COM program to perform hardware detection.	→	NTDETECT.COM collects a list of currently installed hardware components and returns this list for later inclusion in the registry under the HKEY_LOCAL_MACHINE\ HARDWARE key.
	↓		

NTLDR Configuration Selection	If this computer has more than one defined Hardware Profile the NTLDR program will stop at this point and display the Hardware Profiles/Configuration Recovery menu.	→	Lacking more than one Hardware Profile NTLDR will skip this step and not display this menu.
	↓		
Kernel Load	After selecting a hardware configuration (if necessary) NTLDR begins loading the XP kernel (NTOSKRNL.EXE).	→	During the loading of the kernel (but before it is initialized) NTLDR remains in control of the computer. The screen is cleared and a series of white rectangles progress across the bottom of the screen. NTLDR also loads the Hardware Abstraction Layer (HAL.DLL) at this time which will insulate the kernel from hardware. Both files are located in the \system32 directory.
	↓		
NTLDR Boot Device Drivers	NTLDR now loads device drivers that are marked as boot devices. With the loading of these drivers NTLDR relinquishes control of the computer.	→	Every driver has a registry subkey entry under HKEY_LOCAL_MACHINE \SYSTEM\Services. Any driver that has a Start value of SERVICE_BOOT_START is considered a device to start at boot up. A period is printed to the screen for each loaded file (unless the /SOS switch is used in which case file names are printed.
	↓		
Kernel Initialization	NTOSKRNL goes through two phases in its boot process - phase 0 and phase 1. Phase 0 initializes just enough of the microkernel and Executive subsystems so that basic services required for the completion of initialization become available.. At this point, the system display a graphical screen with a status bar indicating load status.	→	XP disables interrupts during phase 0 and enables them before phase 1. The HAL is called to prepare the interrupt controller; the Memory Manager, Object Manager, Security Reference Monitor, and Process Manager are initialized. Phase 1 begins when the HAL is called to prepare the system to accept interrupts from devices. If more than one processor is present the additional processors are initialized at this point. All Executive subsystems are reinitialized in the following order: 1) Object Manager 2) Executive
	↓		

		3) Microkernel 4) Security Reference Monitor 5) Memory Manager 6) Cache Manager 7) LPCS 8) I/O Manager 9) Process Manager
I/O Manager	The initialization of I/O Manager begins the process of loading all the systems driver files. Picking up where NTLDR left off, it first finishes the loading of boot devices. Next it assembles a prioritized list of drivers and attempts to load each in turn.	The failure of a driver to load may prompt NT to reboot and try to start the system using the values stored in the Last Known Good Configuration.
SMSS	The last task for phase 1 initialization of the kernel is to launch the Session Manager Subsystem (SMSS). SMSS is responsible for creating the user-mode environment that provides the visible interface to NT.	SMSS runs in user-mode but unlike other user-mode applications SMSS is considered a trusted part of the operating system and is also a native application (it uses only core Executive functions). These two features allow SMSS to start the graphics subsystem and login processes.
win32k.sys	SMSS loads the win32k.sys device driver which implements the Win32 graphics subsystem.	Shortly after win32k.sys starts it switches the screen into graphics mode. The Services Subsystem now starts all services mark as Auto Start. Once all devices and services are started the boot is deemed successful and this configuration is saved as the Last Known Good Configuration.
Logon	The XP boot process is not considered complete until a user has successfully logged onto the system. The process is begun by the WINLOGON.EXE file which is loaded as a service by the kernel and continued by the Local Security Authority (LSASS.EXE) which displays the logon dialog	This dialog box appears at approximately the time that the Services Subsystem starts the network service.

box.

One alternative option for boot process: Computer on a stick

Now, besides carrying USB flash derives around, we can even carry a mini computer with us. The aptly named "Computer on a Stick," is a small lighter-sized USB drive that carries its own operating system (Linux), a complete Microsoft Office-compatible suite of office software, a web browser, calculator, AOL-compatible chat program, email program... basically, everything in your current computer, only a lot smaller.

It works beautifully. Simply tell your computer to boot to the USB drive. You can do this even if your hard drive is fried, since you'll be working in the BIOS, which loads before any hard drive-based information. Your machine will boot to the Computer on a Stick, and you'll soon be doing everything you want to do without using your hard drive at all. Plus, the device also acts as storage, so you can still take your documents with you and work just about anywhere.

An example of pre-boot authentication: SystemLock--Pre-Boot Authentication with SmartCards in the BIOS

A BIOS password protects modern PCs against unwelcome access by outsiders, as the PC only resumes the system boot process once the correct password has been entered. An additional protection measure is password-protected access to the Operating System. Users of protection mechanisms like these labor under the delusion that they are secure and that these mechanisms provide adequate control over access to their PC. However, both mechanisms can be rendered ineffective by suitable means. In the case of the boot password, tips on how to crack BIOS passwords can be found on the Internet, so the intended protection effect scarcely applies any longer (source: German computer magazine c't, issue 22/2000). Using the cracked BIOS password, a hacker can then easily deletes or replace the original boot password. Once this obstacle has been overcome, the password protection of the Operating System can be bypassed using a suitable bootable floppy disk, and the contents of the PC are then laid bare to the intruder. Security loopholes like this can damage the reputation of a company, cause heavy financial losses and ruin entire companies due to espionage. Strict access control and encryption are therefore the most effective security measures that can be taken to protect data against unauthorized access. Fujitsu Siemens Computers solves this problem with the SystemLock access control function in combination with SmartCards and Security Software.

How does SystemLock help?

According to a MetaGroup study, the risk of user rights being abused is generally rated at 20%, but the true figure is around 70%.

With SystemLock, Fujitsu Siemens Computers offers a powerful access control mechanism which effectively bars the way into the system during booting, prevents unauthorized access to system, setup and hard disks. This is protection against violation of user rights. Keyboard inputs or booting from floppy disk, CD-ROM or via the network are precluded. This method is much more secure than a simple password.

What is SystemLock?

SystemLock is the successor of PC Lock which was already introduced in SCENIC Professional PCs in 1995. SystemLock provides access protection already in the pre-boot phase, before general interfaces and drives are initialized by the system BIOS.

This additional function is included in the system BIOS of SCENIC PCs in combination with the Pro or Pro Manageability and Security Packages. The Pro Manageability and Security Package comprises the internal SmartCard reader, SICRYPT SMARTY 2 and a SICRYPT CryptoCard SmartCard. SystemLock works only in combination with a SmartCard reader and a CryptoCard SmartCard, both included in the Pro Manageability and Security Package. With SystemLock activated, the SCENIC PC can only be started up with a SmartCard initialized for the individual PC and a personal secret number, the personal Identification Number (PIN). In other words, possession of the SmartCard and knowledge of the PIN are necessary in order to continue the boot process, to start the system and / or to start the BIOS setup. SystemLock therefore delivers a higher level of security than a conventional BIOS password.

Additional new functions of SystemLock:

- Support of built in SmartCard reader / writer;
- Support of external serial SmartCard reader / writer and keyboards with integrated SmartCard reader;
- SmartCard support for the hard disk drive password;
- Support of SICRYPT CryptoCard SmartCards, Siemens CardOS SmartCards and Fujitsu
- SmartCards;
- Enable / Disable of SystemLock in BIOS setup with administrator rights;
- Enable Wake-on-LAN without SmartCard and PIN;
- Support of group concept (SmartCard based access to PC in defined groups);
- Secure Auto Logon in combination with SMARTY.

How does SystemLock work?

SystemLock is a pure BIOS function and offers protection against unauthorized access to the PC independent of the installed Operating System. SystemLock is only possible on SCENIC PCs with built-in SmartCard reader or with an external SmartCard reader. With the configuration of a Pro Manageability and Security package, all the requirements for using SystemLock are met. In order to use SystemLock, the function must be enabled in the BIOS (Available from November 2001). SystemLock supports both, the internal and the external SmartCard reader. In the PCs pre-boot phase, a CryptoCard initialized for this particular PC must be inserted in the SmartCard reader. Once the 4 to 8-digit PIN has been entered, the boot process continues as normal with the initialization of the interfaces and drives. These prompts cannot be bypassed. It is not possible to call the BIOS setup before entering the PIN. Cracking the BIOS by using a different password is not possible. The key is stored on the SmartCard, which simultaneously verifies the PIN. The BIOS merely establishes whether a release for booting is validated by the SmartCard and whether the user is authorized to use the PC.

An example of biometric authentication: Fingerprint-based authentication

As we have mentioned above, the technique behind fingerprint-based authentication can be described as follows:

a. Enrolment: capturing or scanning a fingerprint image to create an initial fingerprint match for a specific user. An image or template of the fingerprint in question is created, using a fingerprint sensor. The requirements on image size and precision depend on how much information is needed for a proper identification. This fingerprint template is saved for subsequent comparisons.

b. Extraction: removal of redundant information. The characteristic traits are extracted from the image – all unnecessary information is removed. The more unique this information becomes the more secure the identification will be.

c. Matching: comparison with stored template.

d. Verification: Comparison of the input image with the image said to be yours (the image can be stored in a smart card or in a PC).

e. Identification: Comparison of the input image with all templates in a database

The authentication problem is similar to a ‘pattern recognition’ problem from the point of view that recognizing an object needs a supply of a sufficient number of points (identifiers) of different intensity level and this is what occurs during an authentication process based on matching of finger prints or hand geometry.

Pre-Boot Authentication has been a BIOS based feature for sometime, however it has not been a very popular feature, due to its inability to offer a convenient method of user participation. Additionally, most current solutions require the user to authenticate more than once (once in the BIOS and once in the O/S).

Historically, fingerprints have been used as the most authoritative method of authentication, but recent court cases in the US and elsewhere have raised fundamental doubts about fingerprint reliability. Other biometric methods are promising (retinal and fingerprint scans are an example), but have shown themselves to be easily spoofable in practice.

In a computer data context, cryptographic methods have been developed (see digital signature and challenge-response authentication) which are currently not spoofable if (and only if) the originator's key has not been compromised. That the originator (or anyone other than an attacker) knows (or doesn't know) about a compromise is irrelevant. It is not known whether these cryptographically based authentication methods are provably secure since unanticipated mathematical developments may make them vulnerable to attack in future. If that were to occur, it may call into question much of the authentication in the past. In particular, a digitally signed contract may be questioned when a new attack on the cryptography underlying the signature is discovered.

The Solution

OmniPass ROM is a powerful tool that enables you to authenticate a user with a Biometric device such as a fingerprint sensor or SmartCard even prior to the O/S booting. This makes the OmniPass application an easy to use alternative to the more common method of authenticating with a password.

OmniPass ROM also provides Asset Protection by enabling the hard disk drive security features that are available in most newer drives.

OmniPass ROM can also work in conjunction with the Softex OmniPass Password Management Software to provide a single-touch log-on method to the O/S.

Simple Integration into OEM System BIOS

OmniPass ROM has been designed for an easy integration process for OEM customers. Softex has designed this product with well defined interfaces into the system BIOS. To minimize the space required in the system BIOS, Softex has placed most of the code in a special hard disk partition.

Biometric Capability

OmniPass ROM works with fingerprint recognition devices from AuthenTec, Inc. and can also be customized to work with other biometric devices.

An example of multi-factor authentication: Two-factor authentication

Two-factor authentication is a security process in which the user provides two means of identification, one of which is typically a physical token, such as a card, and the other of which is typically something memorized, such as a security code. In this context, the two factors involved are sometimes spoken of as something you have and something you know. A common example of two-factor authentication is a bank card: the card itself is the physical item and the personal identification number (PIN) is the data that goes with it.

According to proponents, two-factor authentication could drastically reduce the incidence of online identity theft, phishing expeditions, and other online fraud, because the victim's password would no longer be enough to give a thief access to their information. Opponents argue (among other things) that, should a thief have access to your computer, he can boot up in safe mode, bypass the physical authentication processes, scan your system for all passwords and enter the data manually, thus -- at least in this situation -- making two-factor authentication no more secure than the use of a password alone.

Some security procedures now require three-factor authentication, which involves possession of a physical token and a password, used in conjunction with biometric data, such as fingerscanning or a voiceprint.

Bibliography

1. The PC boot process—Windows. XP
   http://dotnetjunkies.com/WebLog/unknownreference/articles/12284.aspx

2. SystemLock: pre-boot authentication with smartcards in the BIOS. White Paper
   Issue November 2001

List of symbols/abbreviations/acronyms/initialisms

API	Application Program Interfaces
BAPI	Biometric Application Programming Interface
BIOS	Basic Input/Output System
BP	Bootstrap Protocol
CA	Certification Authority
CER	Crossover Error Rate
CHAP	Challenge Handshake Authentication Protocol
COTS	Commercial-off-the-Shelf
CRL	Certificate Revocation List
DHCP	Dynamic Host Configuration Protocol
DND	Department of National Defence
DoS	Denial-of-Service
FAR	False Acceptance Rate
FRR	False Rejection Rate
IPL	Initial Program Load
MANET	Mobile Ad-hoc Network
NBP	Network Bootstrap Program
PBA	Pre-Boot Authentication
PXE	Pre-boot Execution Environment
SSL	Secure Sockets Layer
TSL	Transport Layer Security

Glossary

Technical term	Explanation of term
Ad-hoc network	A self-configuring network of mobile routers (and associated hosts) connected by wireless links
Accounting	A set of statistic controls focuses on the amount of system time, data and resources a user is sharing.
Authentication	The process that provides the techniques to certainly identify a user.
Authorization	Types of actions, activities and services that the user trustfully authenticated has right to perform
Biometrics	A measure of a person's unique biological feature such as a fingerprint or voice.
Boot	The process that loads an operating system into the computer's main memory or random access memory.
Loose equality	Two samples are considered equal if their difference is so small that they cannot possibly come from different persons.
Protocol	A prescribed sequence of rules and procedures for interaction between/among two or more entities in a communication system.
Sniffing attack	Using a sniffer program that copies information out of the authentication device input buffer after the user typed in his/her password.
Replay attack	An attempt by an unauthorized third party to record an over the air message; this may be used later in a process to fool the receiver.

DOCUMENT CONTROL DATA

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) Department of Applied Mathematics University of Waterloo Waterloo, Ontario N2L 3G1	2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable) UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Investigation on Vulnerabilities of Pre-boot and Post-boot Authentication (U)

4. AUTHORS (Last name, first name, middle initial)

Liu, Xinzhi; Wang, Lijun

5. DATE OF PUBLICATION (month and year of publication of document) March 2006	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.) 103	6b. NO. OF REFS (total cited in document) 32

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contractor Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DEFENCE R&D CANADA - OTTAWA
3701 Carling Avenue, Ottawa, Ontario, K1A0Z4

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) 15BQ04	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written) W7714-5-0943
10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC Ottawa CR 2006-083	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This report provides an assessment and clear understanding of pre-boot and post-boot authentication for the future development of biometric authentication system in high security applications. We first give a technology review of current PC boot process, present a brief summary of mainly existing boot techniques, and analyze the advantages and disadvantages of pre-boot and post-boot strategies. Then, a brief summary of pre-boot and post-boot processes is presented with the comparisons among different technologies.

Next, we give an insight view of authentication technologies for pre-boot and post-boot stages respectively, with a particular focus on new biometric techniques. The security differences between pre-boot and post-boot authentication are studied and a comparison among different strategies is presented. Based on the disadvantages and limitations of current authentication techniques, several alternative authentication solutions are proposed, especially for the wireless devices in ad hoc networks.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Pre-boot Authentication, Post-boot Authentication, Biometrics, Wireless Authentication, Ad Hoc Networks