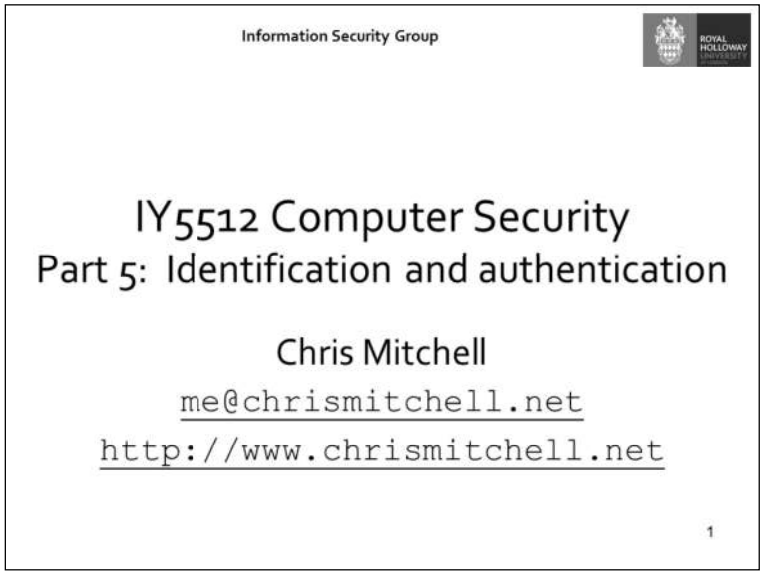
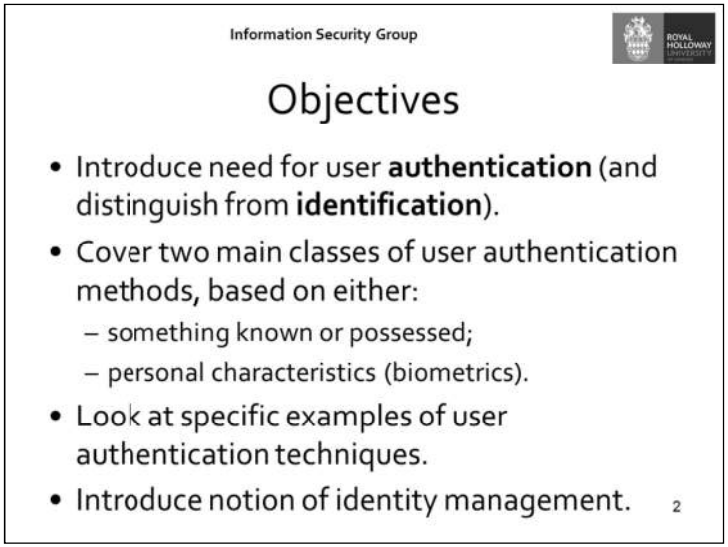




This part of the course will address the following topics.

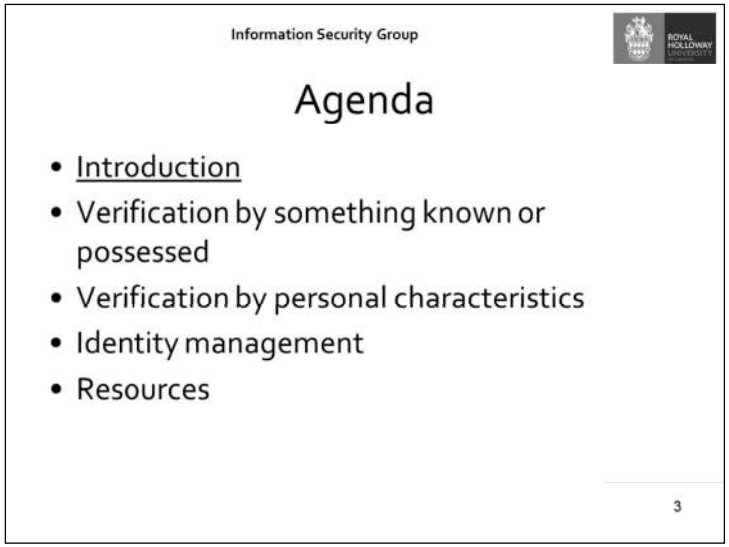
We start by introducing the need for user authentication. We also distinguish it from the notion of user identification.

We then describe the two main classes of user authentication techniques, namely those based on:

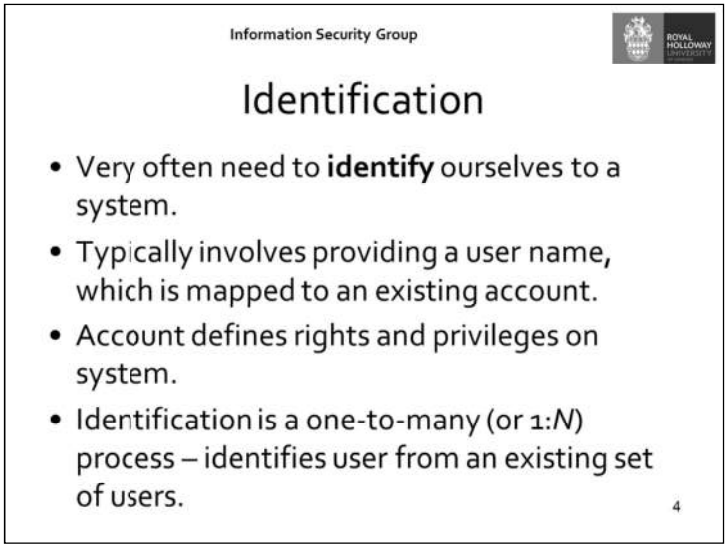
- something known or possessed;
- personal characteristics (biometrics).

We also look at specific examples of user authentication techniques.

Finally, we introduce the notion of identity management and give some examples of identity management systems.

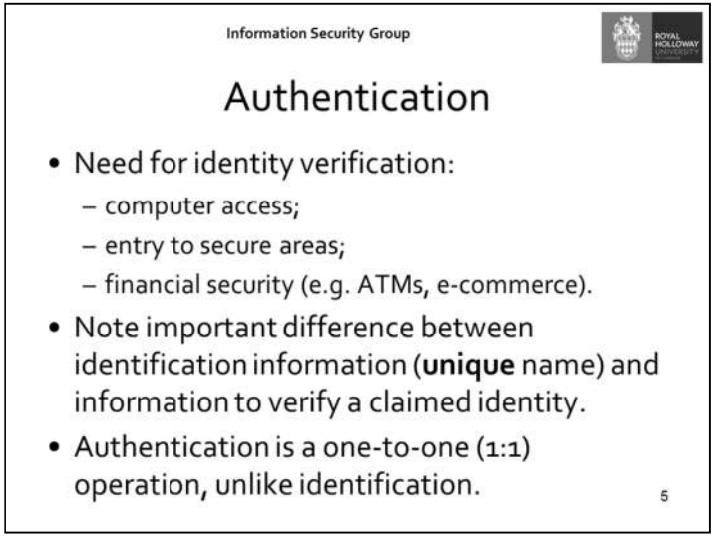


IY5512: Part 5



In many settings we need to **identify** ourselves to a system of some kind. In particular, when using a computer we typically need to provide a user name, which maps to an account held by the system. This account defines the rights and privileges a user has on the system.

Identification is a one-to-many (sometimes written 1:*N*) operation, where we identify which of an existing set of users we are.



The need for **reliably** identifying human users of systems is clear, and covers many different application domains. For example, identity verification is required for:

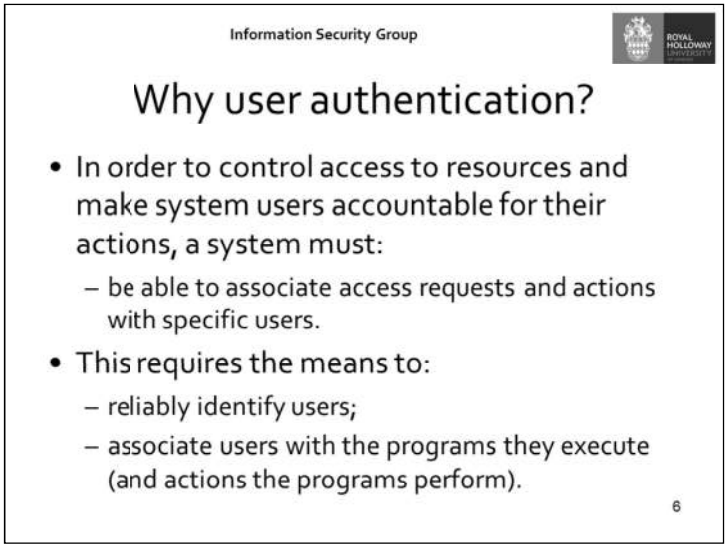
- controlling access to computers;
- limiting access to buildings and secure areas within buildings, etc.;
- controlling use of banking and e-commerce activities.

It is important to understand the difference between the information needed for unique identification of an individual, and that needed to verify that someone has the identity they claim to possess. Thus:

- account numbers and email addresses (i.e. identifying information) often need to be relatively long, in that they need to uniquely identify one of a large class of users, and
- PINs (Personal Identification Numbers), and other types of password (i.e. authenticating information), are often only 4 digits long, since the length of the password is determined by the level of risk, and not by the size of the population.

Ultimately, authentication is a one-to-one operation, where a decision is made whether a claimed identity is correct.

IY5512: Part 5

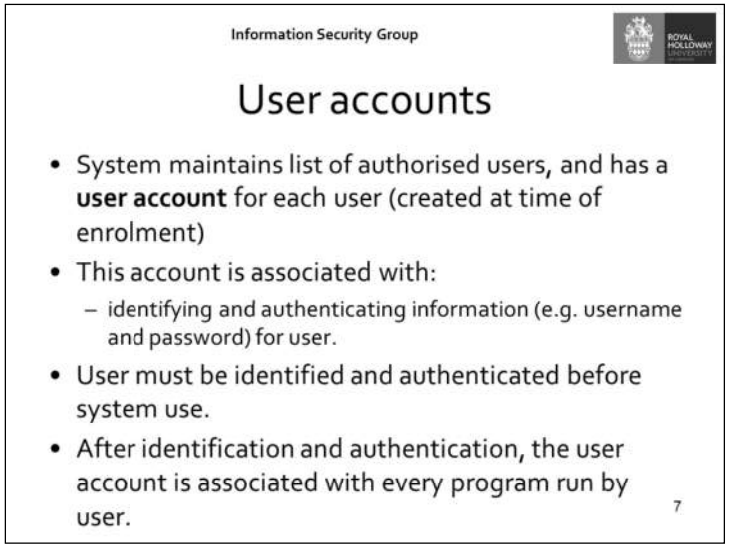


A system must control access to resources and to make users of a computer system accountable for their actions. In order to meet these requirements, a system must be able to associate access requests and actions with specific users.

This, in turn, means that the system must have a way of:

- identifying users in a reliable way (i.e. user identification and authentication);
- associating users with the programs they execute (and hence the actions that these programs perform on behalf of the user).

Arguably, since authorisation depends on its correct operation, user authentication is at the root of all of computer security. If user authentication fails, then all other security measures may become pointless.



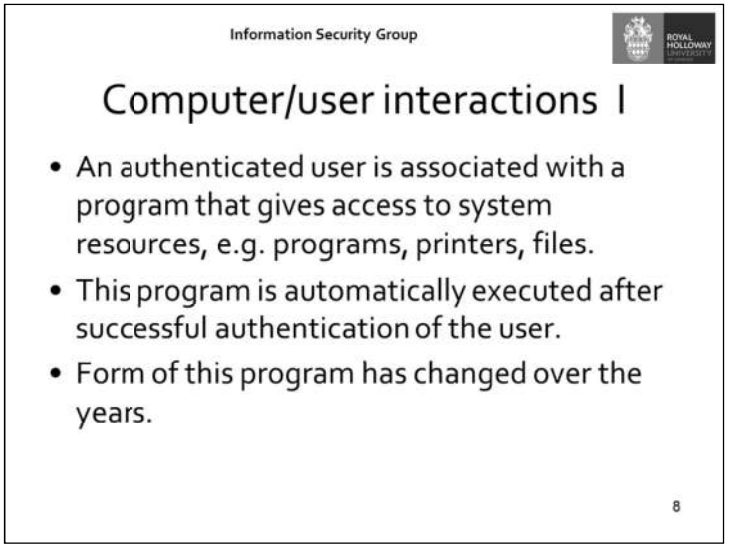
In practice, a system will typically maintain a list of authorised users and associate a **user account** with each such user. This used account is created as a result of a (one off) **enrolment** process. This user account is associated with:

• identifying information (typically a username or user identifier) that is used to identify the corresponding user; and

• authenticating information (e.g. a password) that is used to verify that the identification is correct.

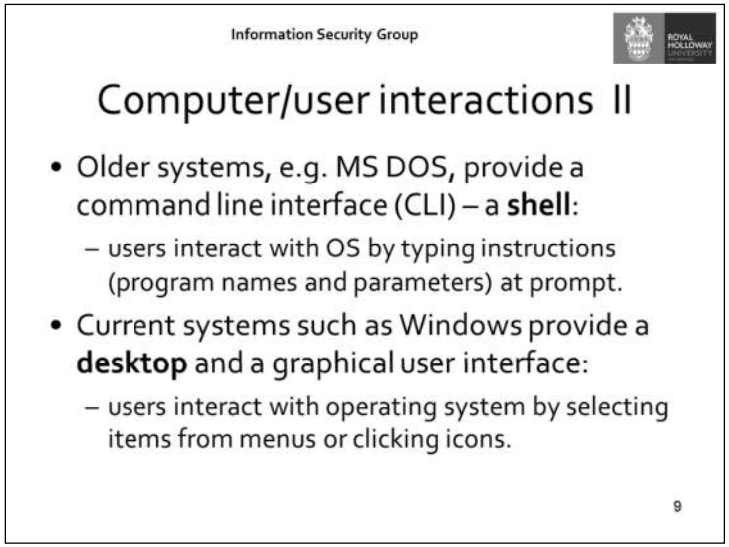
A user must be identified before using a computer system – that is, he or she must provide the system with the identifying information associated with the account. The user must also be authenticated, using the authentication information associated with the account.

Once a user has been identified and authenticated, the user account must then be associated with every program that the user chooses to execute.



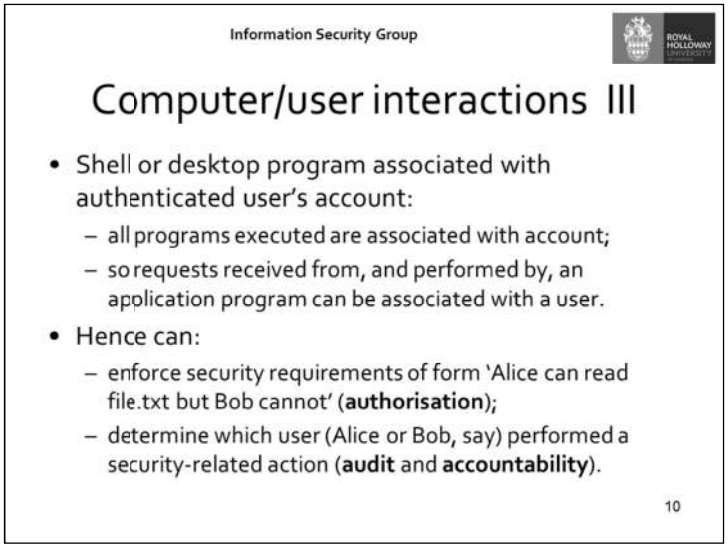
After authentication, a user is associated with a program that provides an interface to system resources such as programs, printers and files. This program is automatically executed following successful authentication of the user.

The precise form of this program has changed over the years.



Older systems such as MS DOS or the original version of Unix provide a command line interface (CLI) called a **shell**. Users interact with the operating system (and hence the hardware) by typing instructions (program names and parameters) at a prompt.

More modern systems such as Windows provide a **desktop** and a graphical user interface. Users interact with the operating system by selecting items from menus or clicking on icons. [Typically the command line interface still exists in some form].



The shell (or the desktop) program is associated with the authenticated user's account:

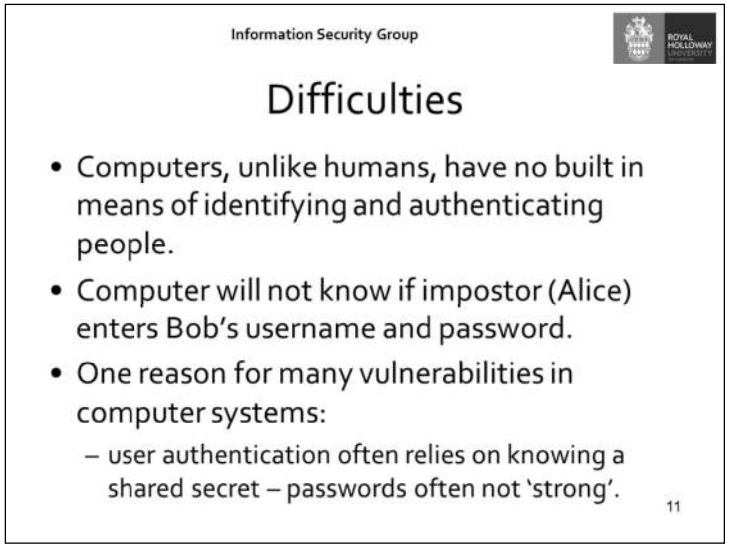
• all programs the user subsequently executes will also be associated with this account;

• as a result, requests received from and performed by an application program can be associated with a user.

Hence we can:

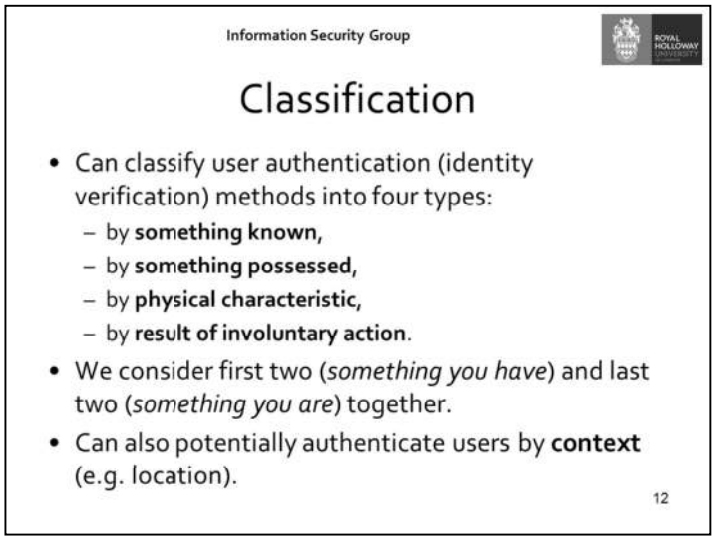
• enforce security requirements of the form 'Alice can read file.txt but Bob cannot' (authorisation);

• determine which user (Alice or Bob, say) performed a security-related action (audit and accountability).



Computers have no in-built means of identifying and authenticating people. Humans are pre-programmed to perform this task; however, a computer will not know if an impostor (Alice) enters Bob's username and password.

This is one reason for the many vulnerabilities in computer systems. User authentication (identity verification) often relies on knowledge of a shared secret, and passwords are often (usually) not strong secrets.

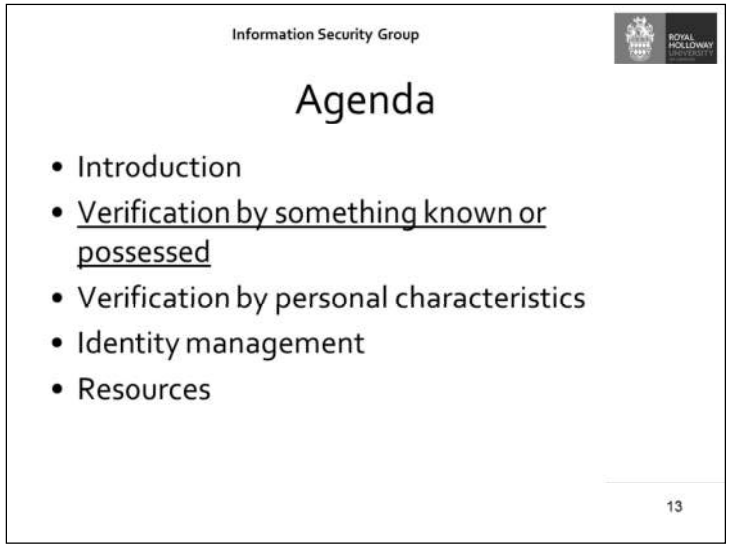


Davies and Price suggest that identity verification (user authentication) techniques can be divided into four classes:

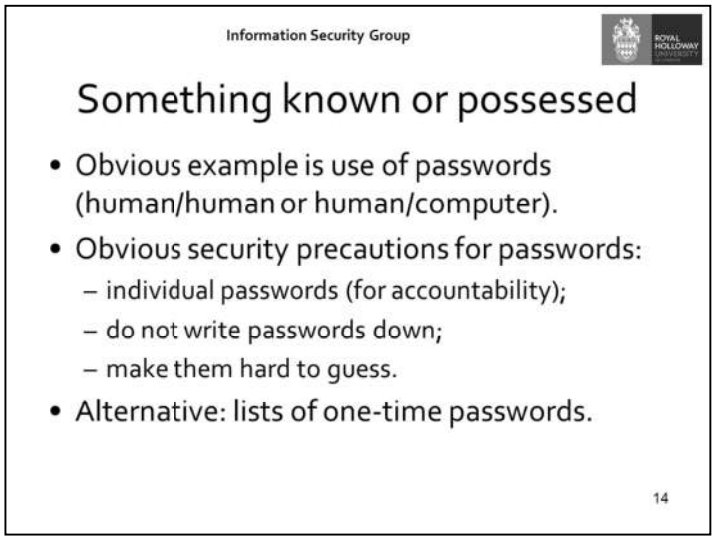
- verification by **something known** (e.g. a password or PIN),
- verification by **something possessed** (e.g. a passport or smart card),
- verification by a **physical characteristic** (e.g. a fingerprint or facial characteristics),
- verification by the **result of an involuntary action** (e.g. a signature).

The first two are both verification by something you *have* (information or a physical object), and the second two are both verification by something you *are*. We use this division into two classes to structure our description of identity verification techniques.

Note that it may also be possible to at least partially authenticate a user by the context of the interaction., e.g. user location (as measured by GPS or IP address) or use of a particular platform or browser type. (Indeed, *browser fingerprinting* appears to have become widely used).



We next consider in greater detail user authentication techniques based on passwords and user tokens.



Examples of verification schemes of this first type include:

• passwords used by pairs of humans (as in spy stories);

• passwords for computer accounts; and

• PINs for banking (note that there exist standards for PIN management, such as ANSI X9.8-1982).

There are some obvious security precautions one should take when allocating and using passwords. For example:

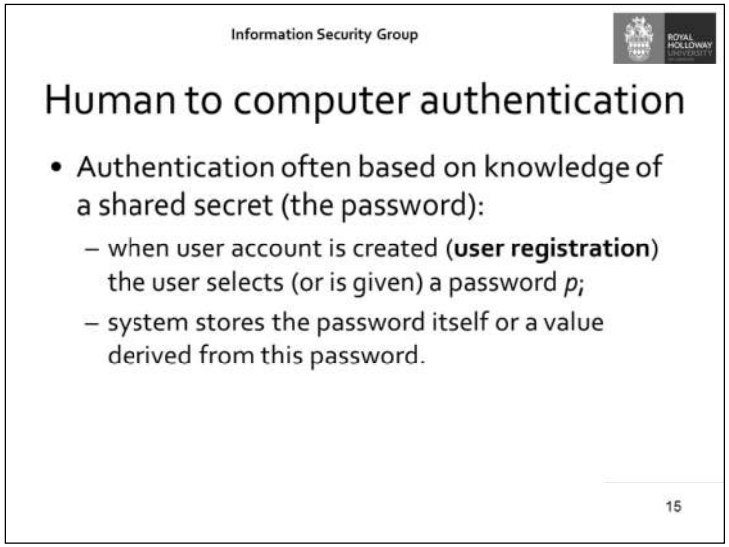
• where possible passwords should be issued to individuals rather than groups of individuals, and then the use of the password can be made accountable to one user,

• passwords should not be written down (at least in an 'obvious' form, or outside of a physically secure area),

• passwords should be chosen so that the risk of correct guessing is minimised;

• avoid re-use of passwords.

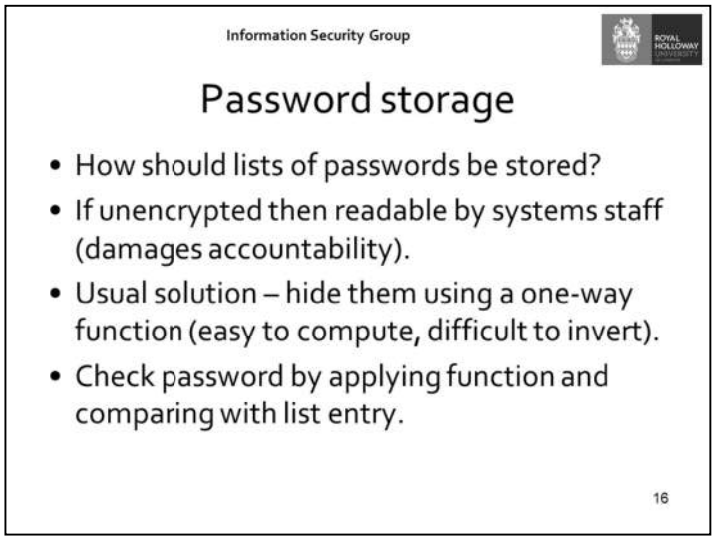
An alternative to conventional passwords is to use 'one-time passwords', i.e. passwords which are only ever used once. This is, however, rather inconvenient, since it is necessary to store lists of valid passwords (and delete the entries in the list once they have been used). However, such a system was used for a number of years within the SWIFT banking network (SWIFT = Society for World-wide Interbank Financial Transfers).



In practice, human to computer authentication is often based on knowledge of a shared secret (the password):

• when a user account is created (**user registration** or enrolment) the user selects (or is given) a password *p*;

• the system stores either the password itself or a value derived from this password.



Clearly, if passwords are to be used to control access to a computer, then password information will need to be held on the computer to enable users' entered passwords to be checked. However, storing a list of passwords presents obvious security problems. If they are stored on a computer in an unprotected form (albeit that regular users might be prevented from gaining casual access) then the 'super-users' (system management staff) will be able to read them. They could then subsequently use a password to masquerade as another user, and the computer's audit trail would wrongly account for actions taken.

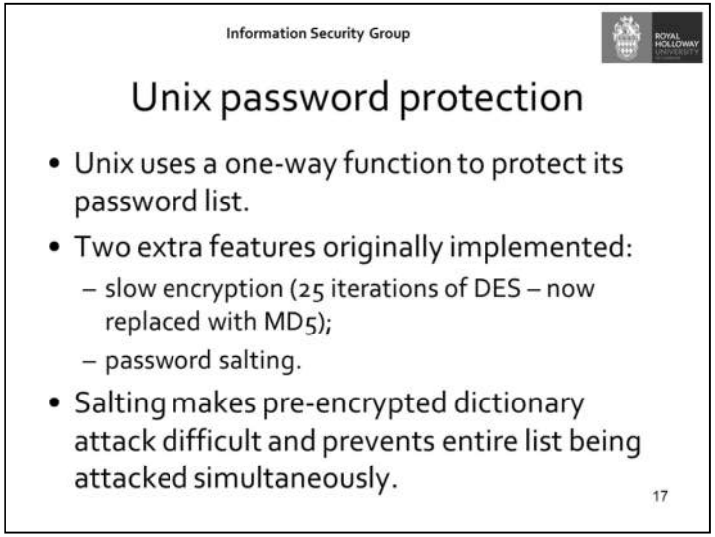
The usual solution to this problem (the idea is usually attributed to Roger Needham back in the 1960s) is to use a *one-way function*. Such a function needs to be:

• easy to compute, and

• difficult to invert.

Then, instead of storing a list of passwords, the results of applying the one-way function to each of the passwords is stored.

To verify a password, it is now necessary to apply the one-way function to the provided password and compare the resulting value with the value stored in the appropriate place in the list.



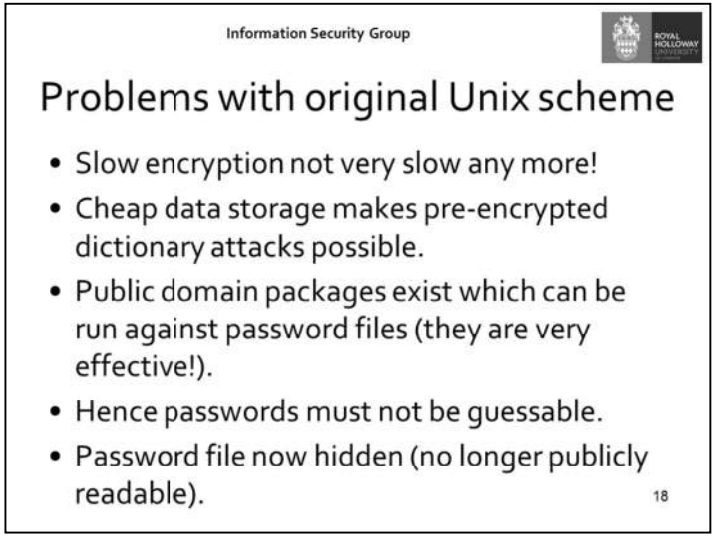
The Unix operating system's password scheme uses this idea. Practical experiments performed over the last 30 years show that many users will choose 'bad' passwords, even if they are warned about the consequences of choosing obvious passwords.

There are two extra features in the original Unix password protection scheme. They have both been designed to make the derivation of user passwords from possession of the *password file* (the list of one-way encrypted passwords) as difficult as possible.

• *Slow encryption*. The one-way function was deliberately made slow to compute. The function used was a version of the DES block cipher algorithm. The first eight characters of the user password were used to construct a 56-bit DES key, which was then used to DES-encrypt the string of all zeros. The output ciphertext is then re-encrypted (using the same key), with this process being repeated a total of 25 times. This means that verifying any guess at a password will take a significant amount of time. This lost effectiveness over time, because software DES implementations could be made to run very fast – use of DES has now been replaced by the hash function MD5.

• *Password salting*. Before applying the one-way function, a 12-bit random value is generated. This value (called the *salt*) is used to modify the internal operation of the DES algorithm, and hence affects the output of the 25 iterations. The salt is stored in the password table with the output of the one-way function. Salting prevents simultaneous searching of the entire password table (perhaps using a table of likely passwords, for example the contents of a dictionary, a list of names, etc.). It also makes the pre-computation of the results of applying the one-way function to a list of likely passwords much more difficult and costly (since it is necessary to store 2^{12} = 4096 different 'encrypted' versions of each candidate password).

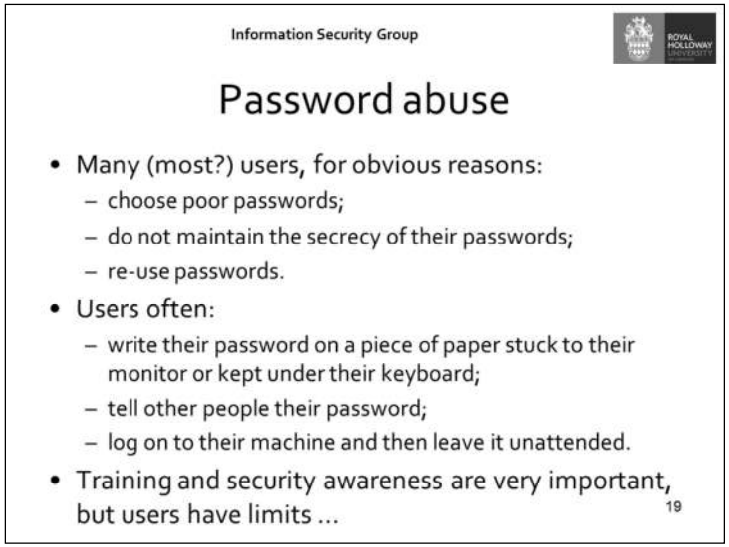
Use of a non-standard variant of DES (cf. the salting process) also helped by ruling out the use of high-speed DES hardware to speed up searches for passwords.



Unfortunately, advances in computing power (processing and storage) severely weakened the original Unix password protection system. The 'slow' encryption is not so slow any more, and large pre-computed tables of encrypted passwords are not so expensive any more.

Indeed there are various public domain software packages available which have been designed precisely for the job of finding badly chosen user passwords by looking only at the password file. These packages are typically equipped with large dictionaries of likely passwords, and personal experience has shown that in some cases over 50% of user passwords can be found in a day or so (the computer running its searches 'in the background').

Hence, if the password file is publicly available, as it was in Unix environments (although this is not the case any longer), then 'guessable' passwords should not be used. Indeed, system managers would be wise to use one of the password guessing packages at regular intervals to track down users making bad password choices. Otherwise an outsider might do the same thing maliciously – there are plenty of documented examples of such events.



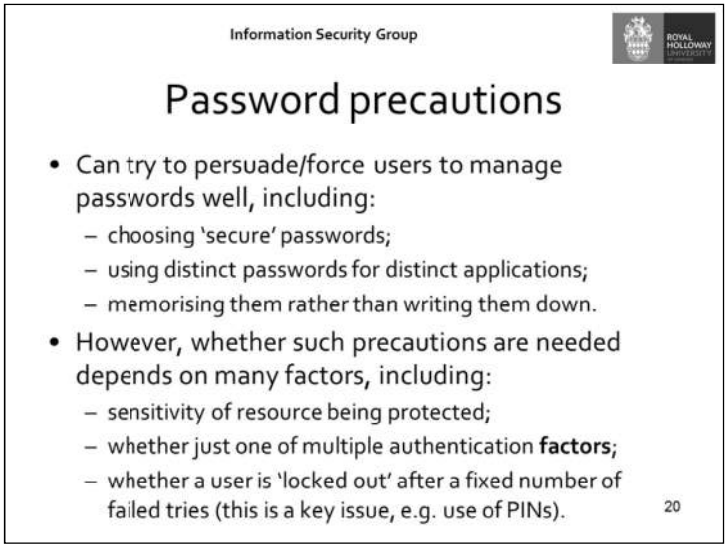
Many (perhaps most) users, for obvious reasons:

- choose poor passwords;
- do not maintain the secrecy of their passwords; and
- re-use the same password with multiple accounts.

Users often:

- write their password on a piece of paper stuck to their monitor or kept under their keyboard;
- tell other people their password;
- log on to their machine and then leave it unattended (although this is a slightly separate issue).

Training and security awareness are very important here! However, poor password choice is very likely given that users are expected to remember ever-increasing numbers of such shared secrets. Ultimately, we must not expect users to do the impossible.



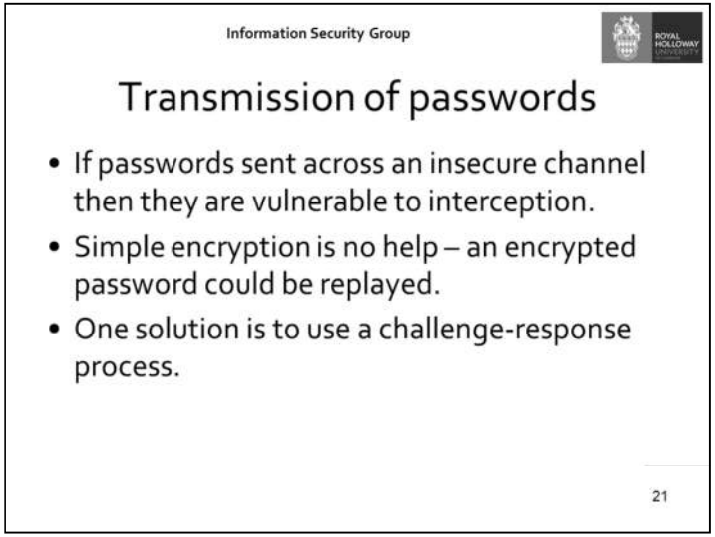
System administrators and security managers can try to persuade/force users to manage passwords well, including:

- choosing 'secure' (unguessable) passwords, e.g. not just a word;
- using distinct passwords for distinct applications;
- memorising them rather than writing them down.

However, whether or not such elaborate precautions are necessary depends on many factors, including:

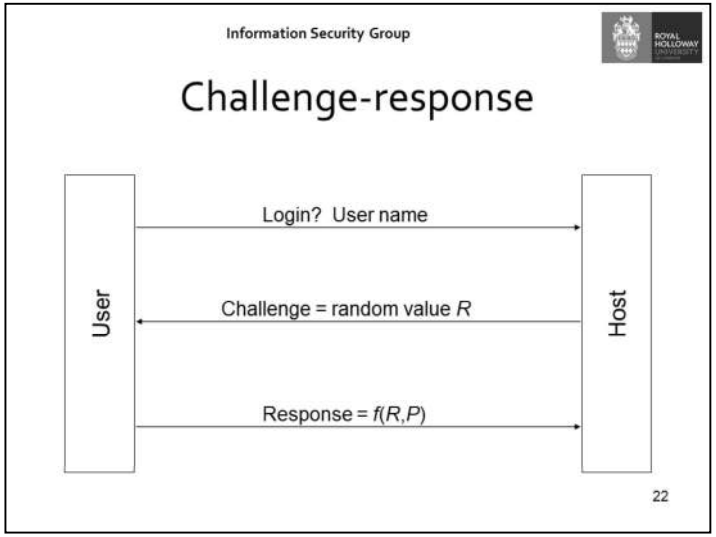
- the sensitivity of resource being protected;
- whether the password is just one of a number of authentication **factors**;
- whether a user is 'locked out' after a fixed number of failed tries (locking out users prevents an exhaustive search for passwords).

Indeed, if users can be locked out after a small number of failed log in attempts, then in some cases, particularly where **dual factor** schemes are used (e.g. PINs used with bank cards), relatively low entropy (variability) passwords, such as 4-digit PINs, can be highly effective.



If passwords are transmitted across an insecure network, e.g. a LAN or a public telephone line, then any passwords sent 'in clear' are extremely vulnerable to interception. Moreover, simply encrypting a password does not help, since an intercepted enciphered password can be replayed just like one sent 'in cleartext'!

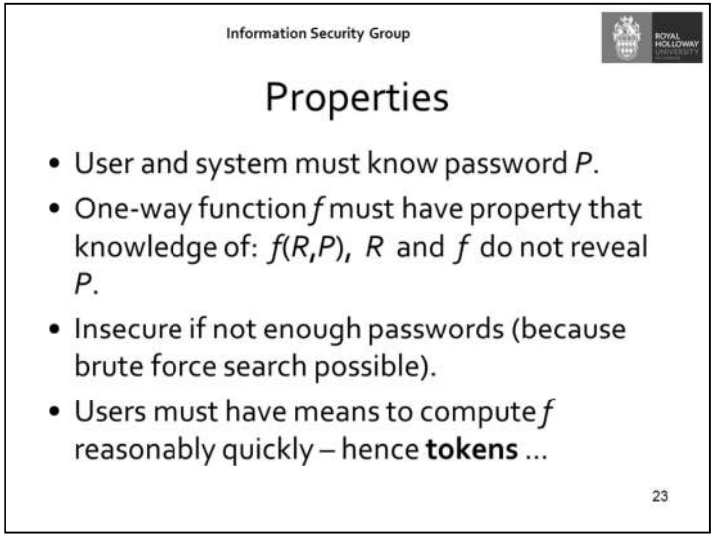
One well-established solution to this problem is to use a 'challenge-response' user authentication process.



A challenge-response system requires the user to possess a secret password *P* and the means to compute a one-way function *f*.

After requesting access to a system (and supplying a user name) the host system responds with a random 'challenge' *R*. The user then responds with the result of applying the function *f* to the combination of the values *R* and *P*.

The system (also equipped with the password *P*) performs the same calculation, thereby accepting or rejecting the user. Typically, the user passwords will be stored in a physically secure sub-system, to prevent unauthorised access to the password list.

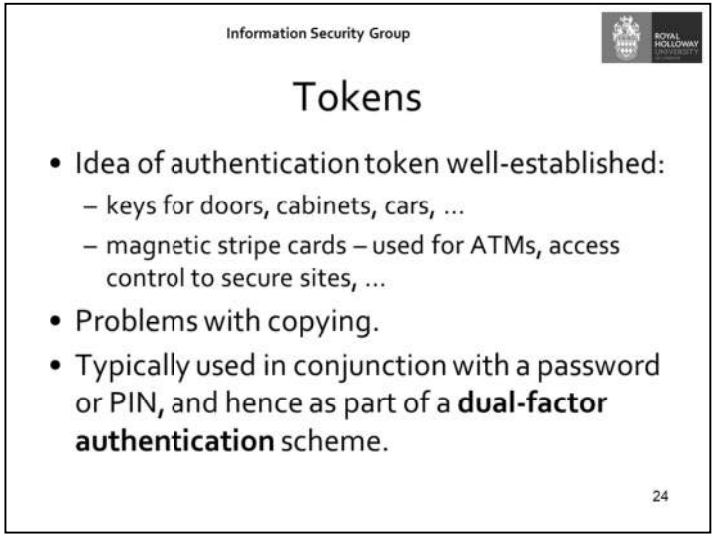


The one-way function f must have the property that knowledge of $f(R,P)$, R and f itself does not compromise the value of P (at least in any feasible amount of time). This must remain true even if the interceptor knows a series of such values.

The system is clearly insecure if the set of possible passwords P is not large enough. If there are too few possibilities for P then the interceptor can try all possible passwords until one is found which gives the correct output when f is given as input an intercepted challenge and the guessed password.

This identification scheme requires the user to have the means to compute f reasonably quickly and easily. Hence this is an example of an identity verification scheme of the 'verification by something possessed' type. This naturally leads us to a discussion of *tokens*, i.e. user-specific devices which are used to support authentication.

[Non-examinable note: The situation can be improved further through the use of **Password-Authenticated Key Establishment (PAKE)** protocols. These protocols enable a secure key to be established between two parties sharing a password in such a way that brute force attacks on the password cannot be performed, even by an active eavesdropper. The most such an attacker can do is to engage in the protocol with a legitimate party using a single password guess, and the attacker can learn whether or not the guess is correct. Such attacks can be mitigated by limiting the number of failed protocol interactions.]



The idea of giving users authentication tokens by which they can identify themselves is a very old one, and one that is used very widely. For example, we all regularly use physical keys to access doors, cars, filing cabinets, etc.

Unfortunately physical keys, and many other kinds of identity verification token (e.g. ID cards, passports, credit cards), can be lost, borrowed, copied and/or tampered with.

Tokens are typically used in conjunction with a password or PIN, and hence are one part of a **dual-factor authentication** scheme, i.e. where the user needs to demonstrate knowledge or possession of two different objects to be successfully authenticated.

Information Security Group

Magnetic stripe cards

- Very widely used type of token.
- ISO/IEC 7810 specifies card dimensions and magnetic stripe format.
- User ID on magnetic stripe.
- Usually used with PIN.
- Off-line systems – PIN check data on card. N.b. PIN check data must be 'salted'.
- On-line systems – PINs verified centrally.

25

The plastic magnetic stripe card is a very widely deployed type of token. It is used for the identification and authentication of users for:

- ATMs (Automatic Teller Machines) = 'holes in the wall',
- POS (Point Of Sale) operations in shops, etc.,
- access control to secure sites, etc.

ISO/IEC 7810 is an international standard specifying the dimensions of cards and the format of the magnetic stripe on the back.

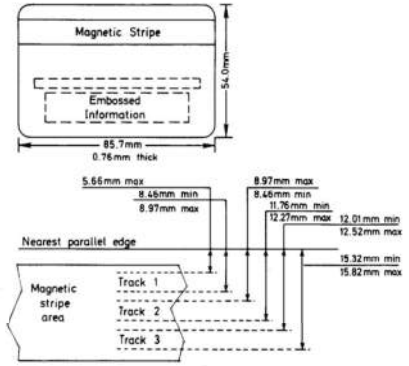
Typically, user identification information (e.g. bank account number) will be held on the magnetic stripe. Such a card is usually used with some kind of PIN to verify a user's claimed identity.

In off-line systems PIN verification information must be stored on the card. This could mean that the PIN is held in enciphered form on the card, or that the result of applying a one-way function to the PIN is written there. In either case the PIN must be combined with user-variable information (such as all or part of the user identification number) prior to encipherment, to prevent attackers of the system compiling matching lists of PINs and enciphered PINs.

In on-line systems user PINs can be verified centrally and hence do not need to be written on to the card.

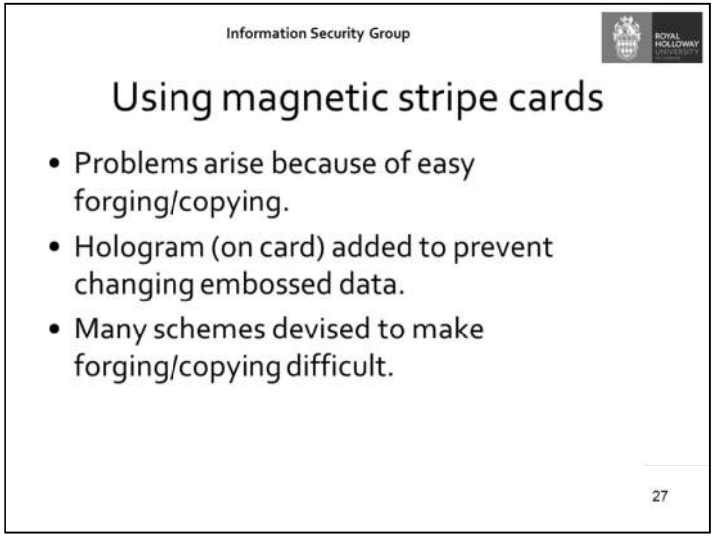
Information Security Group

Layout of magnetic stripe card



26

The complete layout of the card is specified in ISO/IEC 7810.

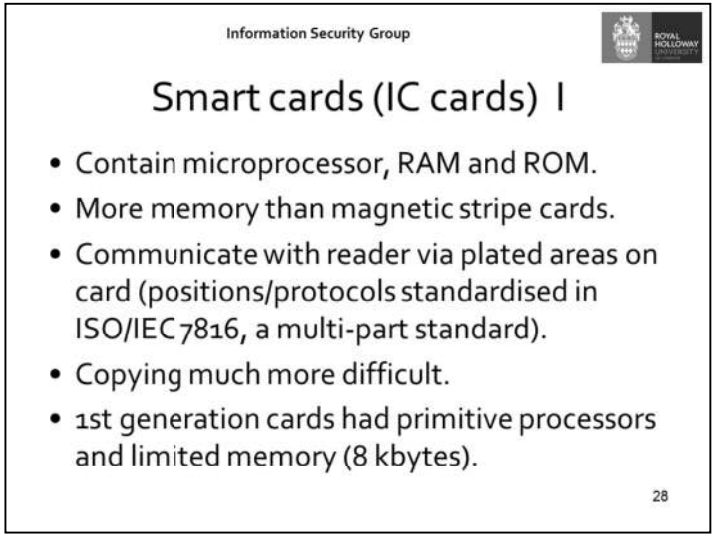


Problems can arise because of the easy forging and/or copying of magnetic stripe cards.

The hologram written onto the front of many of today's credit and debit cards is present as an anti-forgery device to prevent changes being made to the information embossed on the card.

Many proprietary schemes have been devised to try and devise special types of card which are much more difficult to forge and/or copy.

However, ultimately the problems will always exist with a 'dumb' device, where all the data on the card can be read by any device (friendly or hostile).

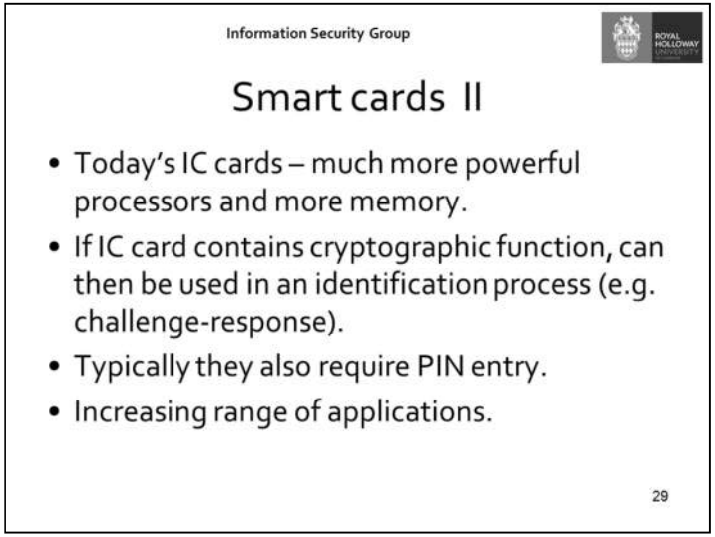


In the 1980s, cards with on-board microprocessors, RAM and ROM were developed, generally known as *smart cards* (or chip/IC cards) Smart cards typically have much more memory than magnetic stripe cards (which usually only store around 250 bytes of data), and have the enormous advantage of on-board processing power. They also offer physical protection to stored data.

The contacts to the internal circuitry are via plated areas on the card surface, including contacts for the supply of external power. The positioning of these contacts, the card size, and the protocols used for IC card - card reader communications are all governed by a multi-part international standard: ISO/IEC 7816.

One of the most important properties of smart cards is that they are extremely difficult to copy. In fact, manufacturers normally keep the internal design details highly confidential in order to make learning the secrets within the chip hard, and hence make copying/cloning much more difficult.

The first generation of smart cards (available from the mid-late 1980s) had primitive 8-bit processors and limited memory (typically up to 8 kbytes). Some also offered limited built in cryptographic functionality.

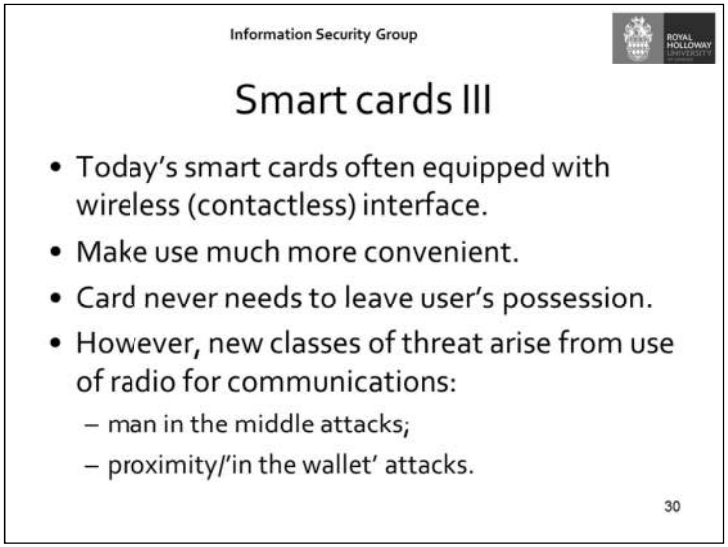


Today's IC cards offer much more powerful processors, more memory, and a variety of possible cryptographic functions. Some current IC cards are capable of performing digital signature calculations in a fraction of a second.

On-board cryptographic processing is enormously attractive, since, when combined with the physical security provided for stored secret data, it enables them to be used as interactive identification devices. Specifically, if the secret user key (or part of it) is stored in the card, then the card can participate in a challenge-response user identification scheme by computing the chosen one-way function.

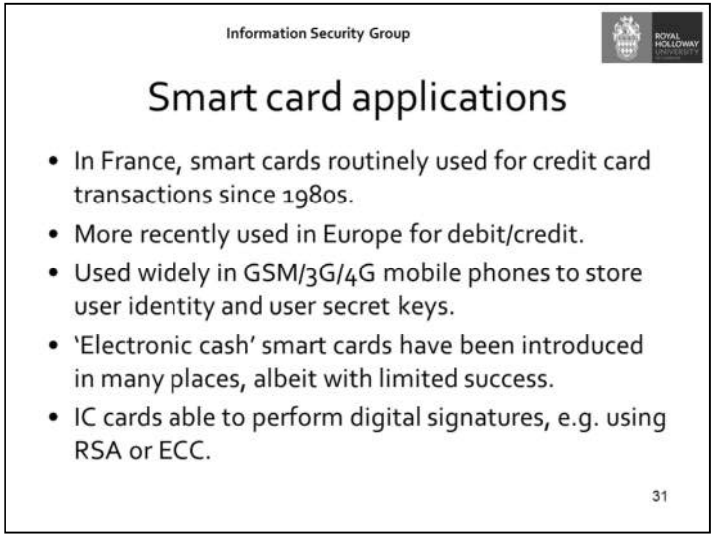
One enhancement of such a scheme requires the user to enter a PIN (via the terminal equipment and smart card reader) before the card will perform its function. This protects against card theft.

As the available processing power and memory has grown, increasing numbers of identification-related applications have opened up for smart cards.



Today's smart cards are often equipped with a wireless (contactless) interface. The deployment of such an interface makes use of the card much simpler and more convenient. In particular, the card never needs to leave user's possession.

However, new classes of threat apply to such cards, arising from the use of radio for communications. In particular it may be possible for a malicious party to interfere with the communications between a card and a legitimate terminal. In addition, it might be possible for an attacker in close proximity to a card to cause it to complete a transaction without the knowledge of the cardholder.



Examples of the growing number of smart card applications include the following.

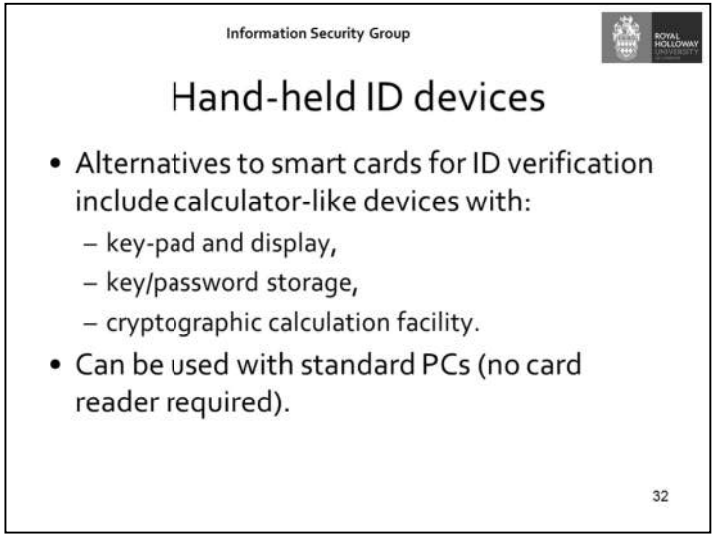
• In some countries (e.g. France) smart cards replaced magnetic stripe cards for use as credit cards back in the 1980s.

• In the UK and across Europe, IC cards have replaced Magnetic stripe cards for debit and credit applications (first pilot started back in 1997).

• GSM, 3G and 4G mobile phones all require the entry of a SIM (Subscriber Identity Module) before they will work. These SIMs store the user identity information and the user secret key (the GSM/3G/4G terminals are not tied to a particular user). These SIMs are 'cut down' standard smart cards.

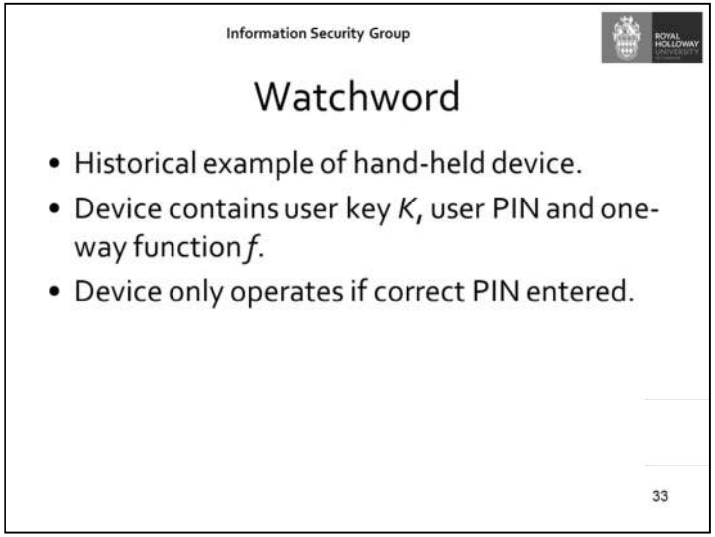
• Geldkarte in Germany and Proton in Benelux support smart card based electronic cash schemes.

• Smart cards capable of performing digital signatures quickly (e.g. using RSA or elliptic curve crypto (ECC)) have been a reality since the late 1990s. This makes all kinds of applications possible.

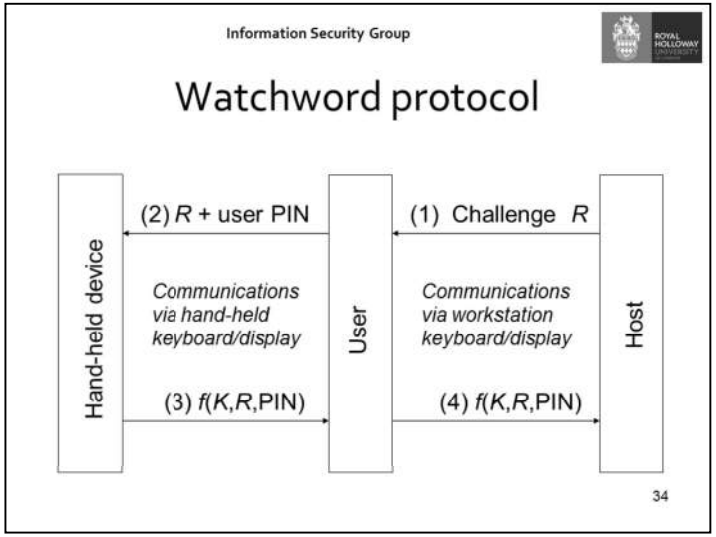


Alternatives to smart cards for user identification include 'calculator-like' devices with a keypad and display. These are of great value, especially in the short term, because they typically can be used with standard PCs, with no special card-readers, etc. These devices need to be capable of storing a secret key and performing cryptographic calculations.

A typical modern device of this type will be credit card or keyfob sized.



One historical example of such a scheme is the *Watchword* system. In this scheme users are equipped with a hand-held device containing a user key K, a user PIN and a one-way function f. To protect against theft, the device will not operate unless the correct PIN is entered.



The device operates in the following way.

(1) The host issues a challenge R to the user, which is displayed on the user's workstation.

(2) The challenge R is entered by the user into the hand-held device, together with the user PIN.

(3) If the PIN is correct then the hand-held device displays the output of the function f when given as input the challenge R, the secret user key K, and the user PIN.

(4) The user then types this response into the workstation for presentation to the host (which can check it since it also possesses the key K and the PIN). In fact the host will typically store the user secret keys in a physically secure sub-system.

Information Security Group

Use of EMV cards with tokens

- Many banks have issued accountholders with card readers for use with an EMV debit card.
- Card and reader jointly form a challenge-response authentication token.
- The reader sends challenge to the card, which generates an account-specific response (using onboard secrets known only to bank).

35

In recent years many banks have issued their accountholders with card readers designed to be used with an EMV (i.e. a 'chip and PIN') debit card.

The card and reader jointly form a challenge-response authentication token.

The reader (which is not user-specific) sends the challenge to the card, which generates an account-specific response (using onboard secrets known only to the bank that issued the card).

Information Security Group

EMV Cards: CAP

- Uses of EMV cards for authentication conform to industry standard known as the Chip Authentication Programme (CAP).
- A CAP reader is shown.

36

Most uses of EMV cards for authentication conform to an industry standard known as the Chip Authentication Programme (CAP). An example of a CAP reader is shown.

These CAP readers are typically provided to cardholders by the card issuers.

Moves in some countries (e.g. Belgium) to promote use of national ID cards with such readers.

Information Security Group

S/KEY

- S/KEY is a public domain one-time password scheme (Internet RFC 1760).
- Based on repeated application of a one-way function *f* of a secret key.
- First apply one-way function *N* times to secret key (to get 1st password), then apply *N*-1 times (to get 2nd password), and so on – giving *N* one-time passwords.

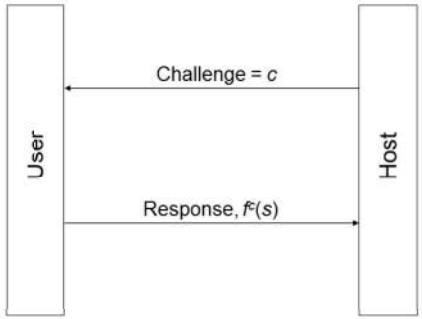
37

The S/KEY system is another scheme of this general type. Devised by Bellcore in the US, it has been made publicly available, and a specification has been published as an Internet *Request For Comments* (RFC 1760).

The user and host also need to share an implementation of a one-way function *f* based on the MD4 hash-function (which is specified in Internet RFC 1320). Although MD4 is no longer considered adequate for use with digital signatures, it would seem perfectly adequate for use in S/KEY. This one-way function takes a 64-bit input and gives a 64-bit output (although MD4 itself gives a 128-bit output).

Information Security Group

S/KEY system

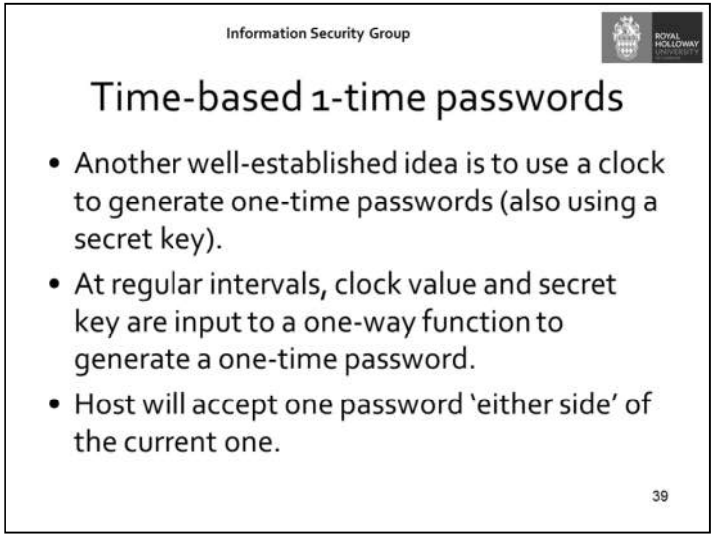


38

The host and user agree on a 64-bit key *S*. The host stores the values *k* and $f^k(s)$, where *f* is a one-way function (f^k simply means apply *f* a total of *k* times), but does not need to store *S*. The user remembers a password from which *S* can be derived. Thus the secret *S* is not stored long-term in either the host or user systems.

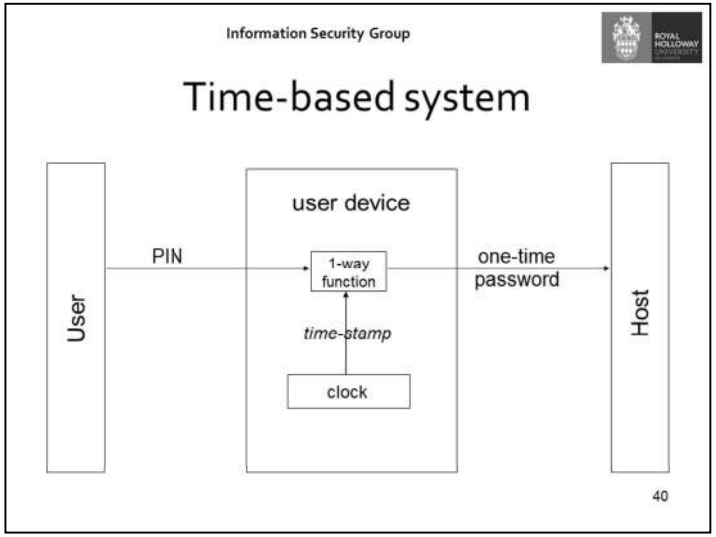
When the user identity is to be verified, the host subtracts one from its current value of *k*, and sends this value as the challenge *c* (=*k*-1). The user responds with the response $r = f^c(S)$, which the host verifies by checking that $f(r) = f^k(S)$. If this check works, then the host replaces its stored value of $f^k(S)$ with $f^c(S)$.

The 'one-time passwords' $f^c(S)$ must be used in the correct order, since knowledge of the final password reveals all the others! This means that, if an attacker can impersonate the host and send the challenge *c*=1, then the user will respond with $f(S)$, which can be used to impersonate the user.



Yet another scheme is based on the idea of using an accurate clock (in conjunction with a secret key) to generate one-time passwords. At regular intervals, e.g. once a minute, the time and secret key are combined within the user device to generate the current 'one-time password'.

To avoid clock drift causing loss of synchronism between the host and the token, the host will typically accept one password 'either side' of the current one. The host can also use the information gained from this process to update a 'clock offset' to adjust for user clock drift.



As in the previous schemes, the host needs to replicate the functionality of the user device so that it can compute the one-time passwords.

Information Security Group

SecurID

- SecurID is a currently marketed token system.
- Generates one-time passwords as a function of an internal clock and a secret.

41

SecurID is a currently marketed token system – available from RSA.

It generates one-time passwords as a function of an internal clock and a secret.

Information Security Group

Using a phone as a token

- Given ubiquity of smart phones, much interest in developing ways to use a phone as an authentication token.
- Many proprietary proposals.
- Over last two or three years, the FIDO industry alliance has developed a standard.
- Idea: user device (e.g. a smart phone) can itself authenticate a user, and can then tell a remote *relying party* (*RP*) about the result.

42

Given the ubiquity of smart phones, in recent years there has been much interest in developing ways to use a phone as an authentication token.

There are many proprietary proposals of this type.

Over the last two or three years, the FIDO industry alliance has developed an industry standard.

The main idea is that a user device (e.g. a smart phone) can itself authenticate a user, and can then inform a remote server, a *relying party* or *RP*, about the result.

Information Security Group

FIDO – introduction

- FIDO alliance is an industry standards body.
- FIDO specifications are authentication technology agnostic – can use biometrics, ...
- Two protocols between user device and RP:
  - **UAF – universal authentication factor**: intended to get rid of passwords;
  - **U2F – universal second factor**: intended to complement a password with a second factor;

43

The FIDO alliance is an industry standards body – unlike ISO, IETF or OASIS.

The FIDO specifications are authentication technology agnostic. That is, FIDO can use biometrics or any other authentication method.

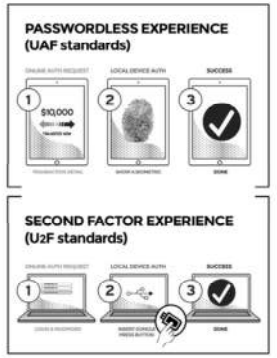
FIDO has specified two protocols, both of which execute between the user device and the RP:

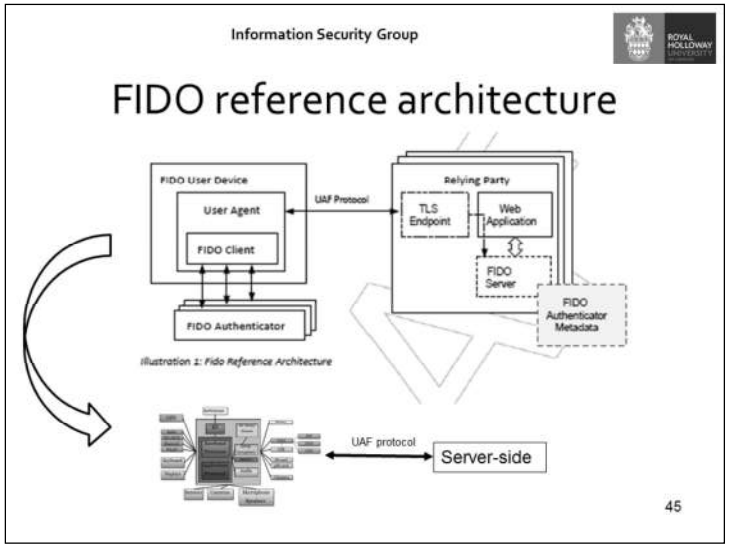
- *UAF (universal authentication factor*) is intended to get rid of passwords;
- *U2F (universal second factor)* is intended to complement a password with a second factor.

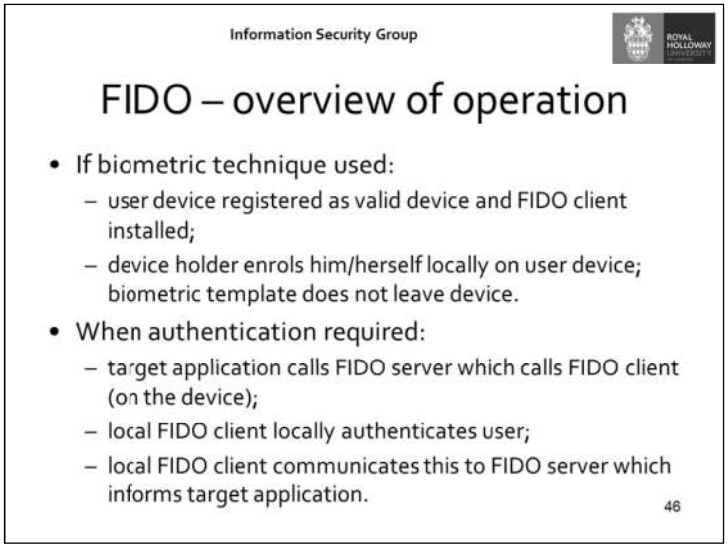
Information Security Group

FIDO – two schemes

- **UAF** – universal authentication factor (replace password by e.g. biometric).
- **U2F** – universal seccnd factor (complement password by second factor).





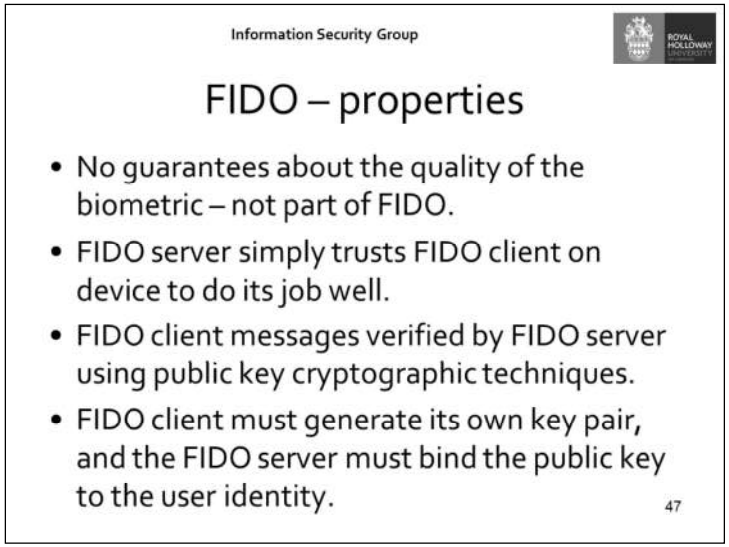


If a biometric authentication technique is used:

- the user device is registered as a valid device and the FIDO client software is installed;
- the device holder enrols him/herself locally on the user device, i.e. establishes a biometric template; the biometric template does not leave the device.

When authentication required:

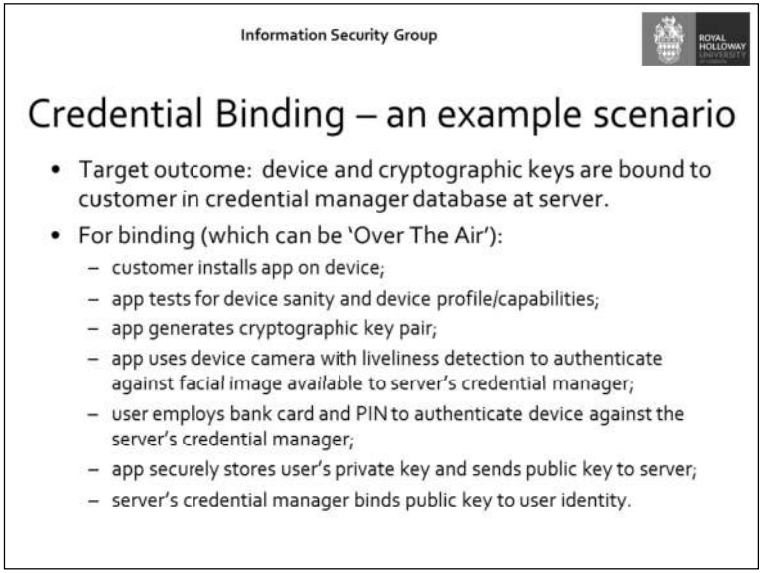
- the target application calls FIDO server which calls the FIDO client application (on the device);
- the local FIDO client locally authenticates user;
- the local FIDO client communicates this to FIDO server which informs target application.



The system offers no guarantees about the quality of the biometric – this is not part of FIDO. The FIDO server simply trusts the FIDO client on the user device to do its job well.

FIDO client messages verified by FIDO server using public key cryptographic techniques.

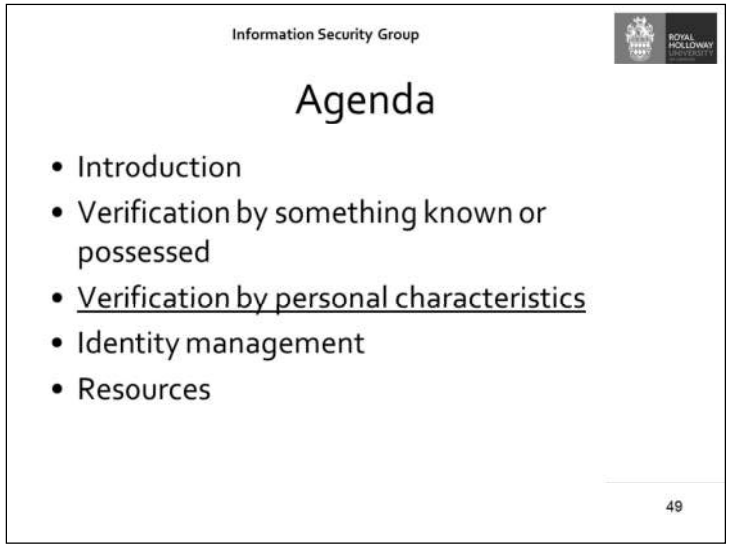
The FIDO client application must generate its own key pair, and the FIDO server must bind the public key to the user identity.



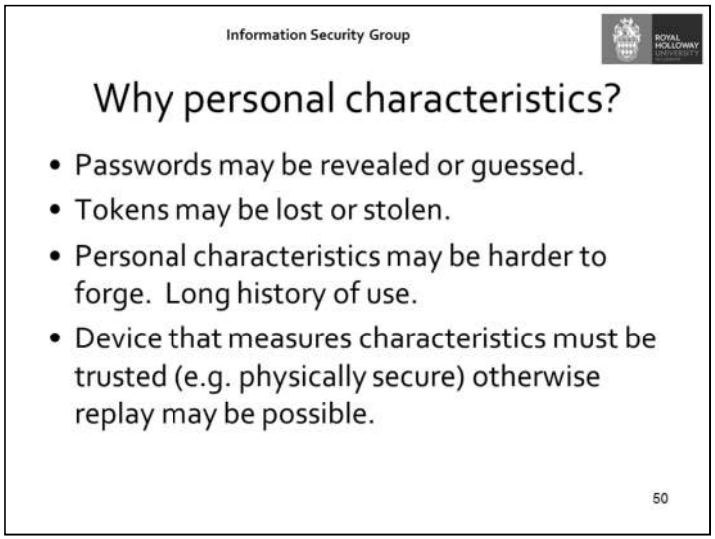
We look at one example of a way in which the FIDO client public key can be bound to the user identity. The target outcome of the binding process is that the device and the user's cryptographic key are bound to a unique customer identity in the credential manager database at server.

For binding (which can be 'Over The Air'):

- the customer installs app on device;
- the app tests for device sanity and device profile/capabilities;
- the app generates a cryptographic key pair for use by the FIDO client on this device;
- the app uses device camera with liveliness detection to authenticate against facial image available to server's credential manager [server authenticates user];
- the user employs his/her bank card and PIN to authenticate device against the server's credential manager [server authenticates device];
- the app securely stores user's private key and sends the public key to the server;
- the server's credential manager binds the received public key to the user identity.



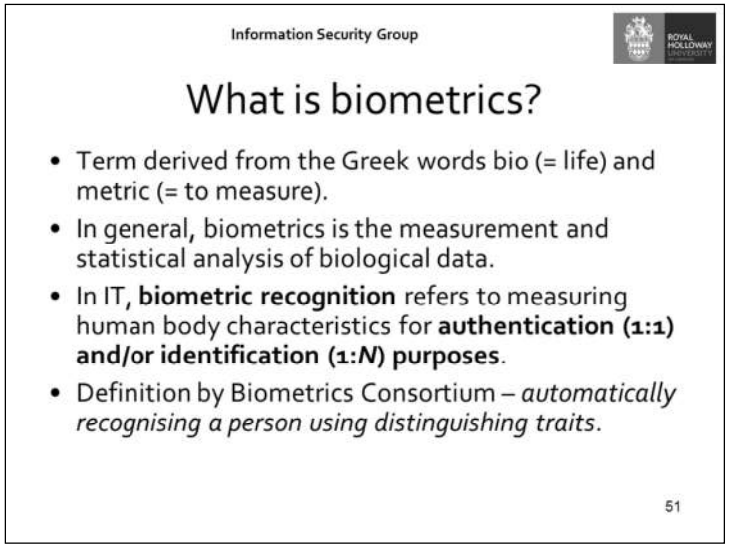
We next consider biometric-based user authentication schemes.



It is clear that passwords may be revealed or guessed, and that tokens may be lost or stolen.

Hence in recent years much attention has been given to automatic identity verification systems based on user characteristics. This is in the belief that such characteristics are harder to forge than user tokens. There is, of course, a long history of manual identification systems based on personal characteristics, from recognition of faces, voices and signatures, to the use of fingerprints in criminal investigations.

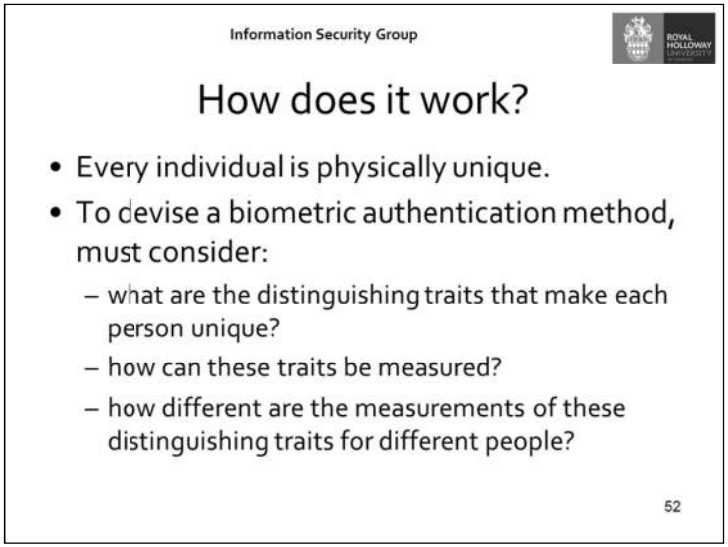
It is important to note that the device actually carrying out the physical measurement must be physically secure, since otherwise a previously enacted (valid) measurement could be replayed to the host. In particular, if use of the measuring device is monitored by a trusted person (as, for example, occurs at US immigration), a number of serious risks are reduced.



The term biometrics is derived from the Greek words bio (= life) and metric (= to measure).

Biometrics refers to the measurement and statistical analysis of biological data. In IT, **biometric recognition** refers to technologies for measuring and analysing human body characteristics for authentication (1:1) and/or identification (1:*N*) purposes. In fact, typically the technology is used for one or the other – it is important to understand the difference between the two!

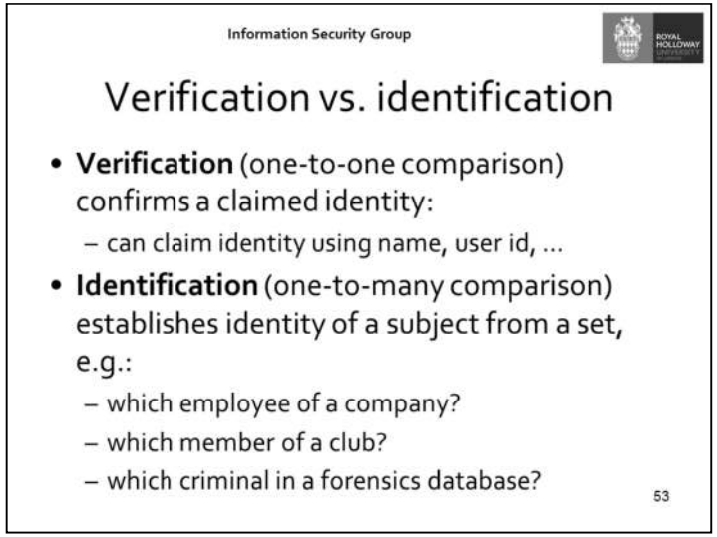
A definition of the use of the term in IT by the Biometrics Consortium is: *automatically recognising a person using distinguishing traits*.



The use of biometrics start from the observation that every person is physically unique.

In order to devise effective biometric identification/authentication systems, we need to consider the following questions:

- What are the distinguishing traits that make each person unique?
- How can these traits be measured?
- How different are the measurements of these distinguishing traits for different people?



There are two basic modes of operation for a biometrics system:

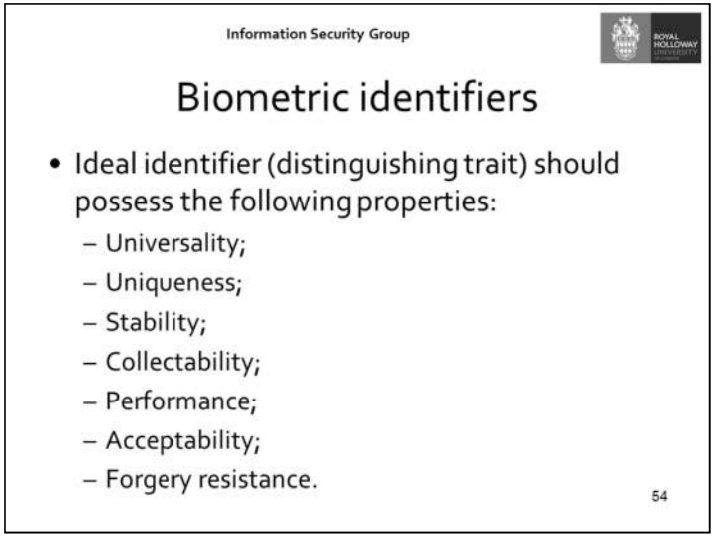
• Verification or authentication: tries to answer the question 'Is the claimant the person who he or she claims to be?' The user claims an identity and the system verifies his/her identity by comparing the biometric information rendered by the user with a reference for the claimed identity stored in the system (for example, in a smartcard). It is a one-to-one comparison.

• Identification: tries to answer the question 'Is the claimant an enrolled user and who is he/she?' The user simply provides his/her biometric information and the system compares his/her biometric data with templates stored in the system database. It is a one-to-many comparison.

Practical questions deriving from the above two classes of application include:

• is the claimant the owner of this card/computer/document? (verification)

• should this individual be given access to the system/room/file? (identification)



An ideal identifier (i.e. the selected distinguishing trait) should possess the following properties:

• universality: nearly all people in the target population should have the characteristic.

• uniqueness: the characteristic of each individual should be unique, i.e. the biometric feature of each individual in the population should be different from that of every other individual.

• stability: the characteristic should neither change with time nor allow alteration.

Any physiological or behavioural characteristic having these properties can be used for personal identification. However, for the purpose of automatic personal identification, the biometric feature should have one more property:

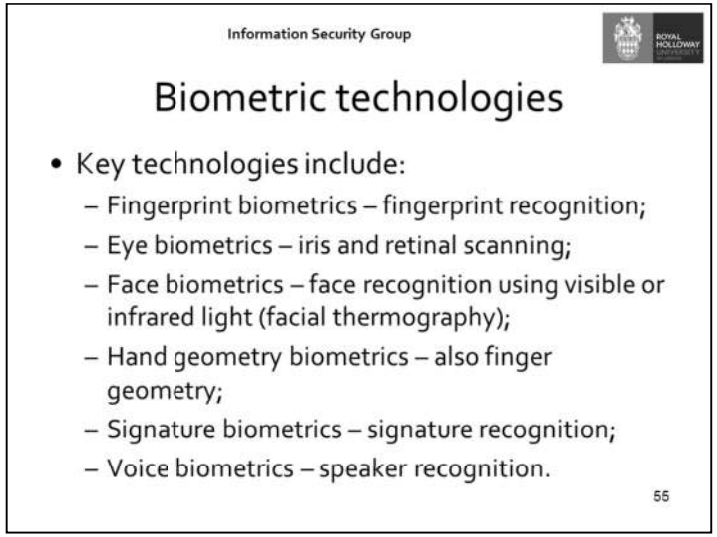
• collectability: it should be possible to measure the characteristic quantitatively.

There are yet some other issues to be considered when a biometric system is being developed:

• performance: achievable identification accuracy, speed, memory requirements.

• acceptability: the extent to which people are willing to accept a particular biometric system in their daily lives.

• forgery resistance: how easy it is to fool the biometric system by fraudulent methods (particularly relevant when technique used for authentication).

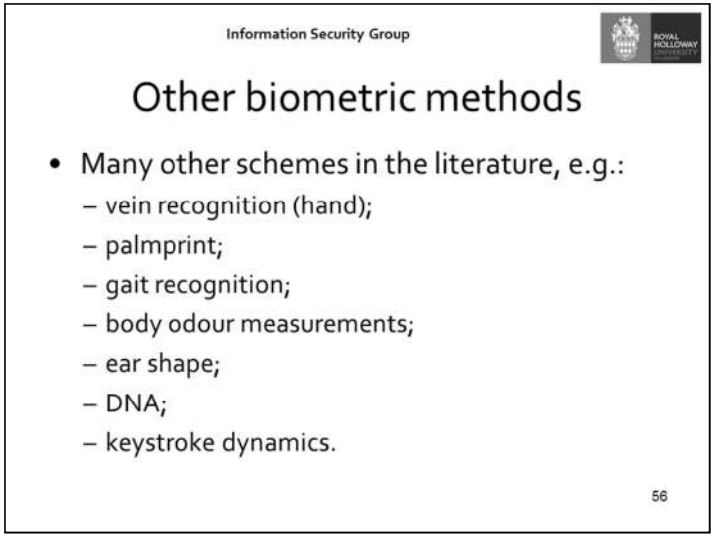


Many different automatic identification schemes have been proposed and used, including:

- recognition of signatures,
- fingerprint analysis,
- voice recognition,
- retinal scan,
- iris scan,
- face recognition,
- hand geometry.

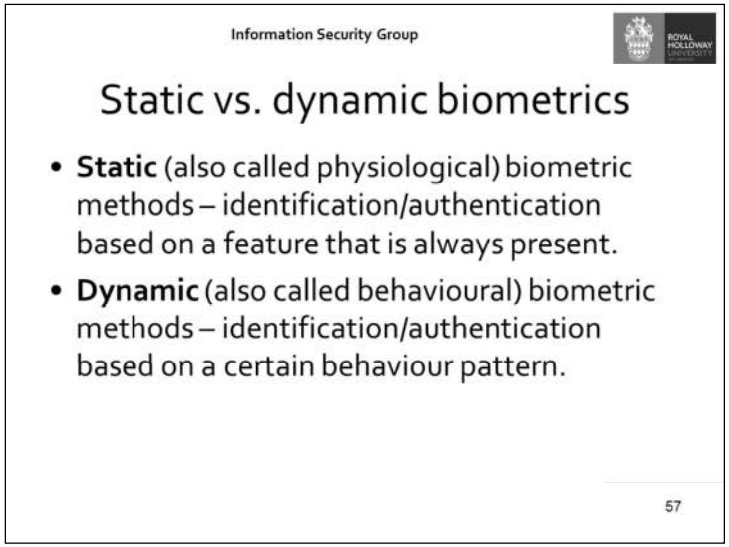
These technologies will be considered in more detail in a subsequent lecture.

IY5512: Part 5



Many less obvious schemes have been proposed in the literature, including:

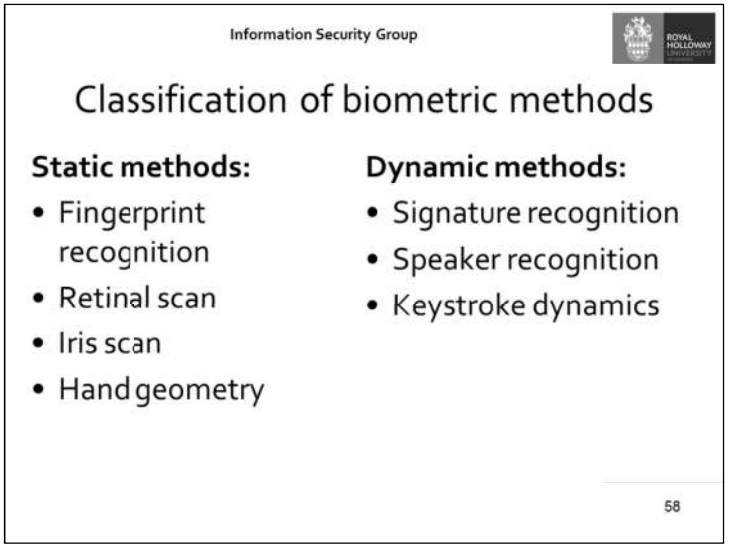
- vein recognition: the vein pattern of the back of the hand.
- palmprint: a third approach to hand identification.
- gait recognition: in which people are recognised by the way they walk
- body odour measurement: automated methods for odour measurements are required in industrial processes and other applications.
- ear shape: possesses the property required from a biometric identifier.
- DNA: clearly our DNA uniquely identifies us (except for twins).
- keystroke dynamics: uses rhythm patterns, such as the time between keystrokes, hold times, finger placement, and the pressure applied on the keys.



Biometric techniques can be divided into static and dynamic methods.

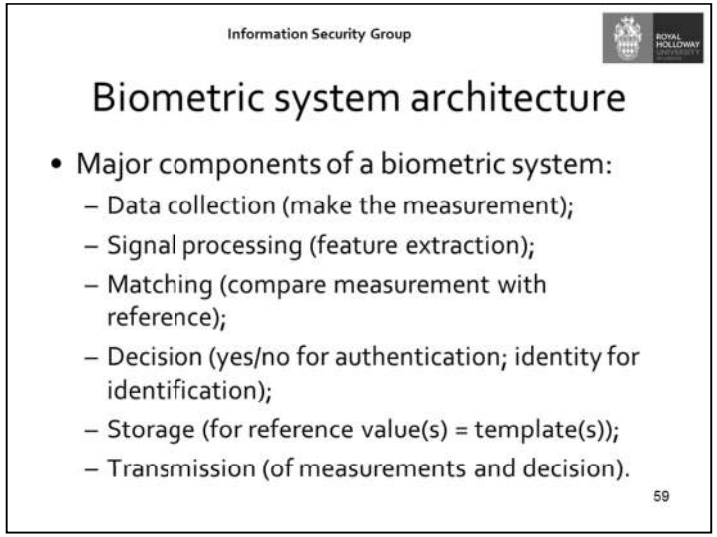
- In **static** (also called physiological) biometric methods, identification/authentication is based on a feature that is always present.
- In **dynamic** (also called behavioural) biometric methods, identification/authentication is based on a certain behaviour pattern.

IY5512: Part 5



Examples of the two classes of biometric method are given.

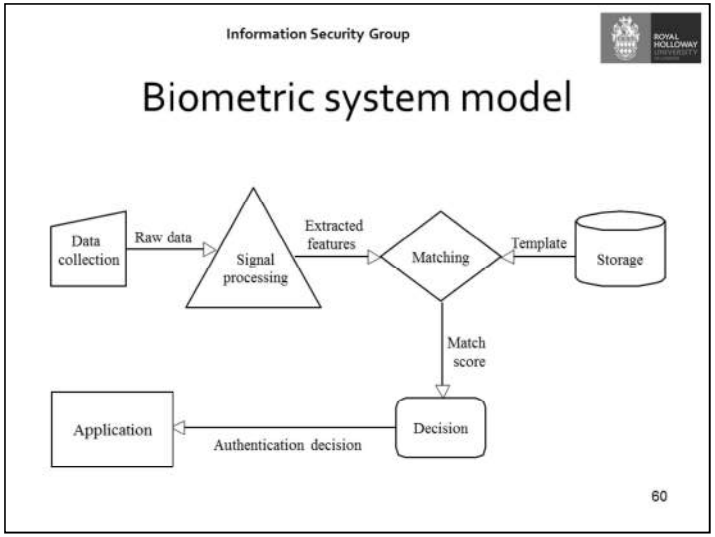
Note that fingerprint recognition is by far the most mature and reliable technique, pioneered by the FBI.



All biometric systems achieve automatic personal identification/authentication by comparing the biometric feature presented by a user with the stored feature of the claimed user (or potential users). Regardless of the biometric feature being used, the biometric system architecture will incorporate the following logical components:

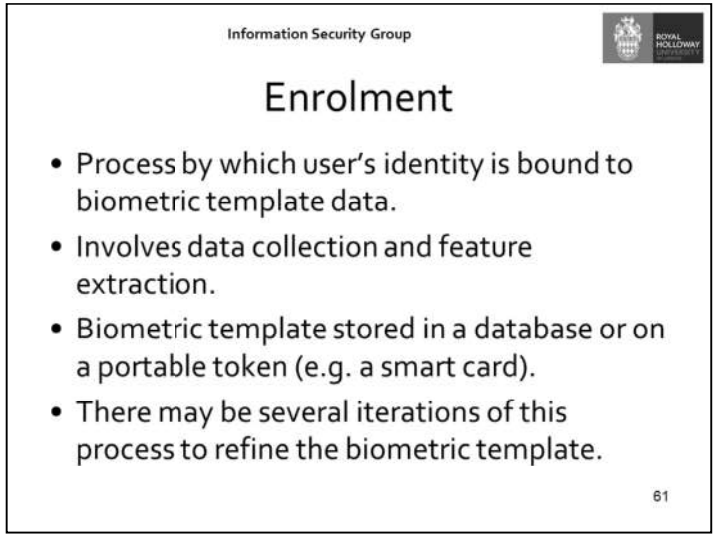
- **data acquisition**: captures the biometric data presented by the individual.
- **feature extraction**: signal processing module that extracts and codes the distinguishing features from the biometric data.
- **matching**: makes a statistical comparison between the extracted features and the reference data stored in the system.
- **storage**: maintains the reference templates of the enrolled users.
- **decision**: interprets the result from the matching module.

This model for a system architecture helps us to analyse the use of biometrics. However, there may be some systems where several of these components are integrated into one unit or are not even present (for example, some systems use raw biometric data for comparison and do not have the feature extraction module).



The diagram shows the model for authentication (identification is similar, except that multiple templates will be involved, and the nature of the decision will be different).

Prior to use of this model, it is necessary to **enrol** a user, in order to create a **biometric user template**.



Enrolment is the process of collecting biometric samples from a user, processing of them to create a biometric reference (template), and storage of this reference in a system database or portable token. Several iterations may be required to achieve a template of appropriate quality.

It is advisable for users to be enrolled at an authorised enrolment site. There must be verification that the reference template being recorded is that of the actual user. Collateral material binding the enrolee to their biometric template should be generated. It is also necessary to ensure that the recorded biometric template is able to properly verify the enrolee's identity. A live matching of the enrolee's reference template should be made before the template is stored in the system.

The key requirements for enrolment are:

• the procedure should be secure, including robust authentication of the enrolee (i.e. the individual being enrolled);

• the quality of the template, including how easily it can be used to identify/authenticate the individual, should be checked;

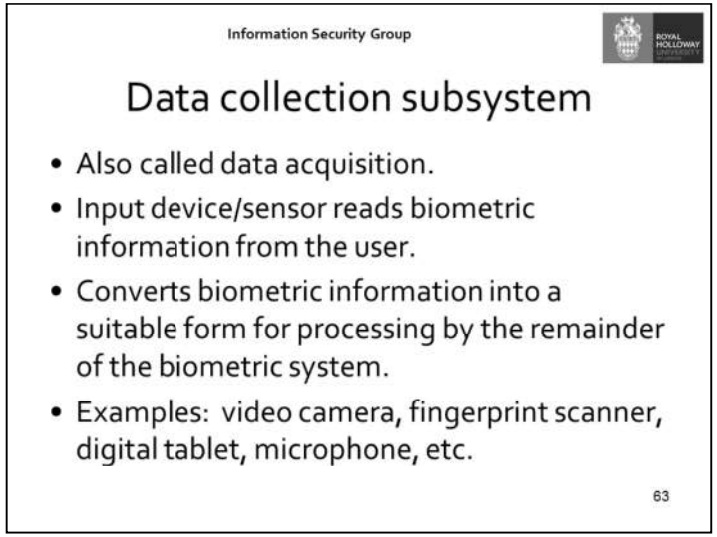
• the enrolment procedure should produce a secure binding between the template and the individual.

There are other requirements for the enrolment procedure.

It should be ensured that the person performing the enrolment – the enroller – has the proper permission to access the enrolment functions, e.g. access control, command authentication, etc. Note that this includes legal authorisation (depending on the jurisdiction) to capture, process and store the biometric characteristics. This means that conducting a test or pilot may be non-trivial.

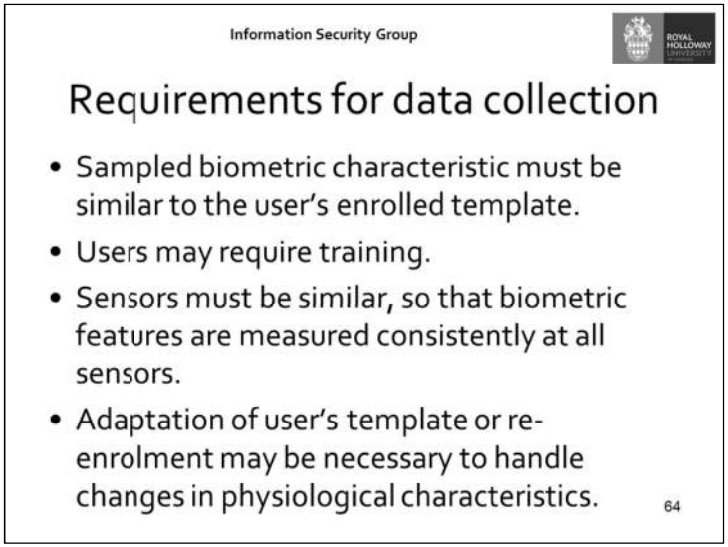
It is also advisable to check if the enrolee has already been registered with the system and/or if a different enrolled user has a biometric template matching the current enrolee's. Until recently doing such a search at enrolment time was not feasible for large databases – however, today vendors claim that they can do such a search in minutes, even for very large databases.

The procedure is rather similar to issuing a public key certificate!



The data collection (or data acquisition) module contains the input device or sensor that reads the biometric information from the user. It is the link between the physical domain and the logical domain.

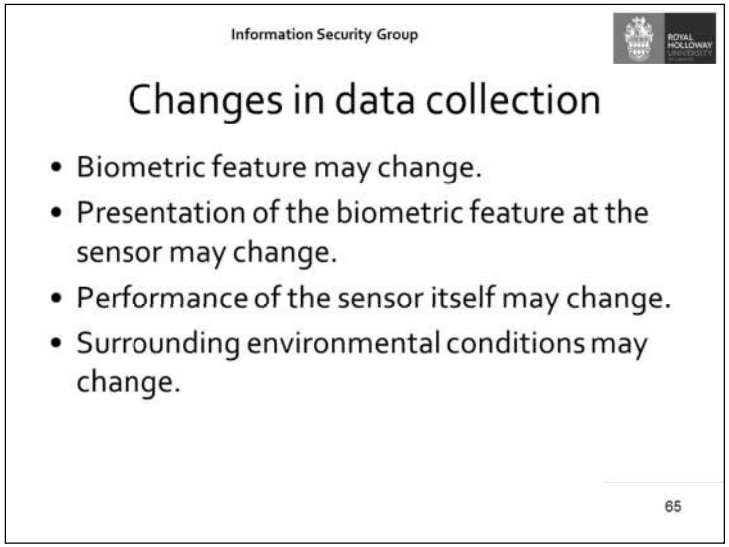
It converts the user's biometric information into a suitable form for processing by the remainder of the biometric system. Examples of data acquisition systems include: video cameras, fingerprint scanners, digital tablets, microphones, etc.



In order to recognise a user successfully, the sampled biometric characteristic must be similar to the user's reference template, to which it is compared. This imposes requirements on the data collection sensor, and may impose training requirements for the users to help them deliver consistent readings. All sensors in a given system must be similar enough that a feature collected by one sensor will closely match the same feature collected at other sensors – and also that collected during enrolment – so that the user can be recognised at any location.

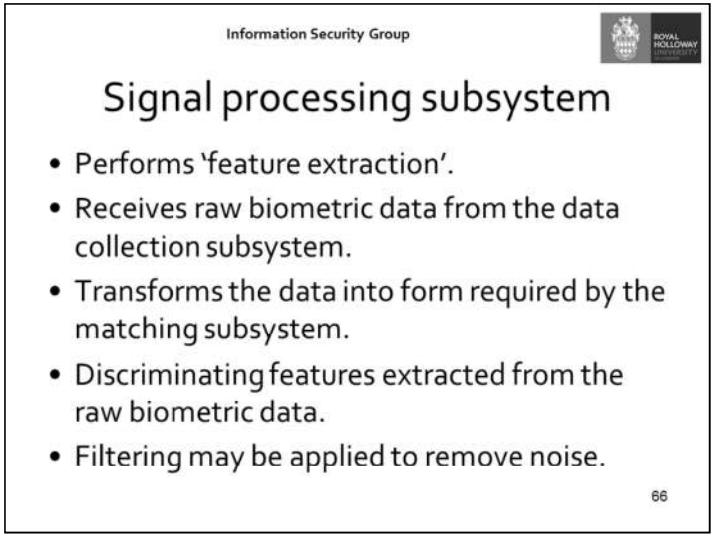
Depending on the biometric technology being used, environmental conditions such as lighting, background noise, weather, can impact on the performance of the data acquisition module.

Adaptation of a user's template or re-enrolment may be necessary to handle changes in a user's physiological characteristics.



In practice, a biometrics system must cope with changes in the data collection environment, and in the data being collected:

- The biometric feature may change.
- The presentation of the biometric feature at the sensor may change.
- The performance of the sensor itself may change.
- The surrounding environmental conditions may change.



The feature extraction module receives the raw biometric data from the data acquisition module and extracts the **distinguishing features** from the raw data, transforming it into the form required for storage and matching. Even for the same biometric characteristic, there are various ways of extracting the distinguishing features. These are often proprietary.

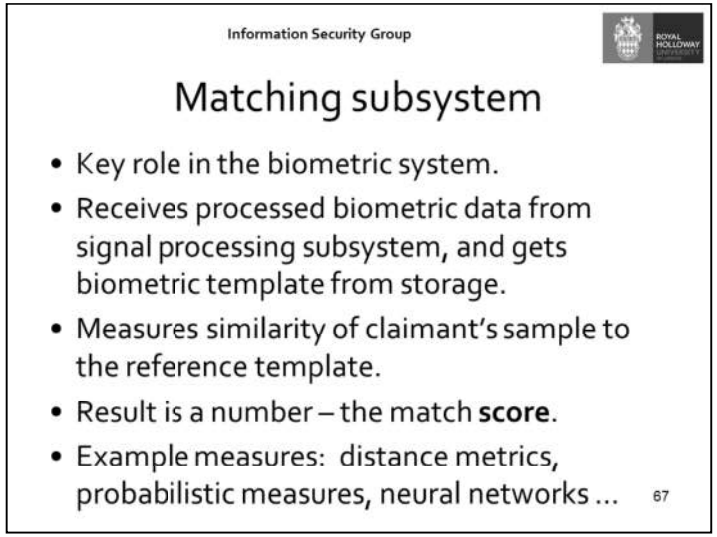
This module may perform a quality analysis of the raw data to determine if it is satisfactory for use. If the data fails the quality test, the user may need to supply the biometric characteristic again.

The raw biometric data may be pre-processed prior to feature extraction in order to remove noise or to be normalised in some way.

Typically, it is not possible to reconstruct the raw data from the extracted features.

Some biometric systems compare raw data, in which case this module is not required.

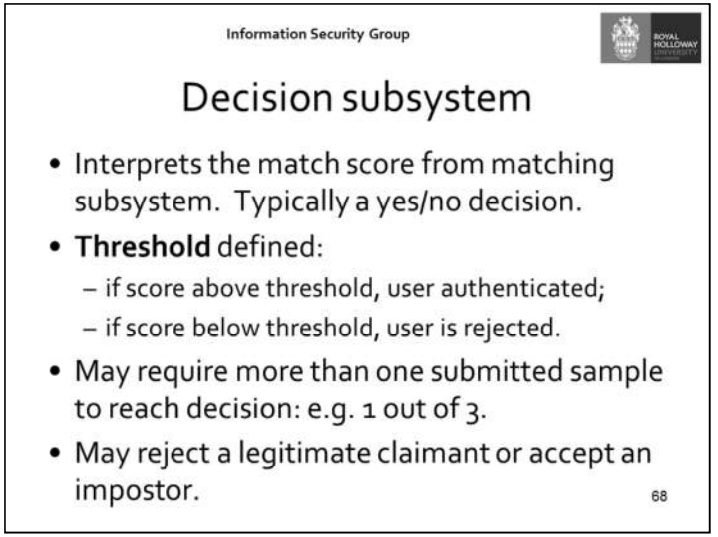
[The EU Schengen VISA system stores both the original picture (a jpeg) and the derived template. Templates are typically proprietary, so the only way to avoid vendor lock-in is to also store the original pictures.]



The matching module has a key role in the biometric architecture. It receives the processed data from the feature extraction system and compares it with the biometric template from the storage module.

The matching module measures the similarity of the claimant sample to the template generated at the time of enrolment. Each comparison yields a **score**, which is a numeric value indicating how closely the sample and the template match. There are different methods for computing the score and some typical examples are: distance metrics, probabilistic measures, and neural network-based methods.

[In most cases the operation of this component is highly proprietary. NIST provides input datasets, vendors provide test results, and NIST compares the 'black boxes' in terms of results.]



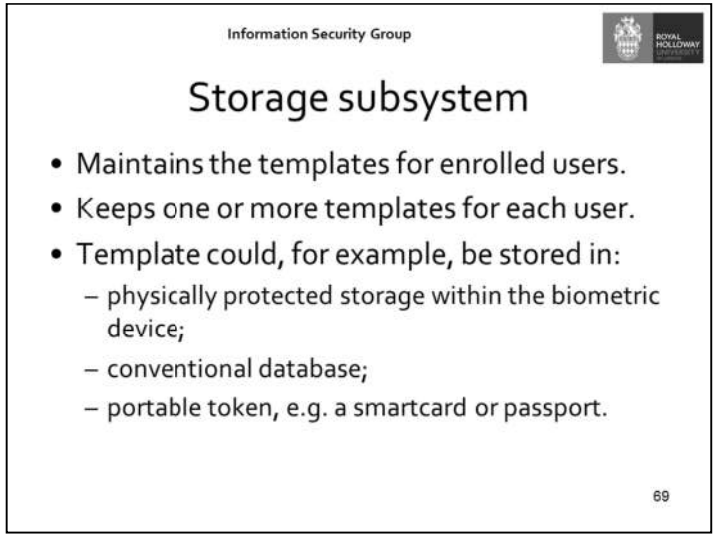
The decision module receives a score from the matching module and, using a confidence value based on security risks and risk policy, interprets the result of the score. (In our discussions below, we assume a high score means a close match and a low score means a poor match).

The decision module usually returns a binary yes or no. In the most common case, the decision is based on a single threshold. If the score is above the threshold, the module concludes that the user is indeed the individual owing the template. If not, the module indicates the user is not that individual. In more complicated cases, the decision is made based on more than one matches and a yes decision is taken if, for example, 2 out of 3 submitted samples match.

Note that it is possible that a legitimate claimant is rejected by the biometric system due to the very nature of the biometric data. The data acquisition module does not collect exactly the same biometric information at every attempt to use the system and so it is possible that a legitimate user is rejected or an impostor is admitted by the biometric system.

The output may not always be a binary decision. There might be a 'grey zone', when a challenge response scheme or a second authentication method is used.

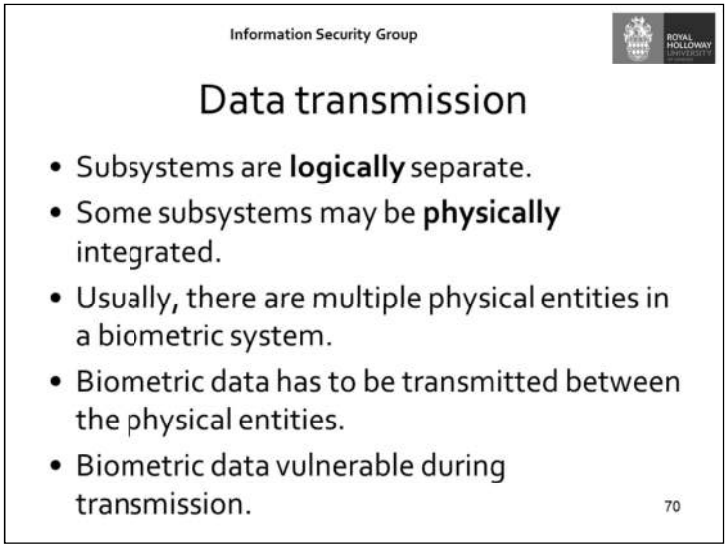
In many cases this subsystem would be integrated with the matching subsystem.



The storage module maintains the reference templates for enrolled (registered) users. It may contain a single template for each user or thousands of templates depending on the system architecture or intended use.

The template may be physically stored in physically protected storage in the biometrics device, in a conventional database on a computer, or in a portable token such as a smartcard or passport.

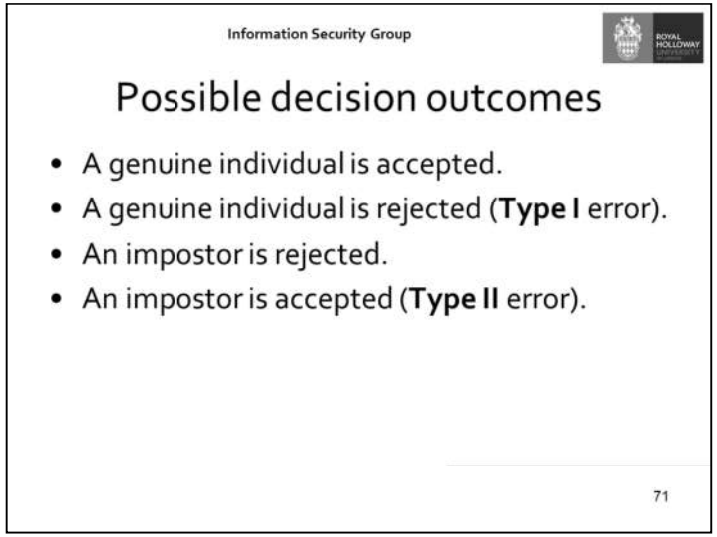
Collateral information, such as name, identification number, etc, binding the owner to his/her reference template may also be stored together with the reference template.



The subsystems within the model are **logically** separate. However, some subsystems may in practice be **physically** integrated.

However, usually, there are a number of separate physical entities in a biometric system. As a result, biometric data has to be transmitted between the different physical entities

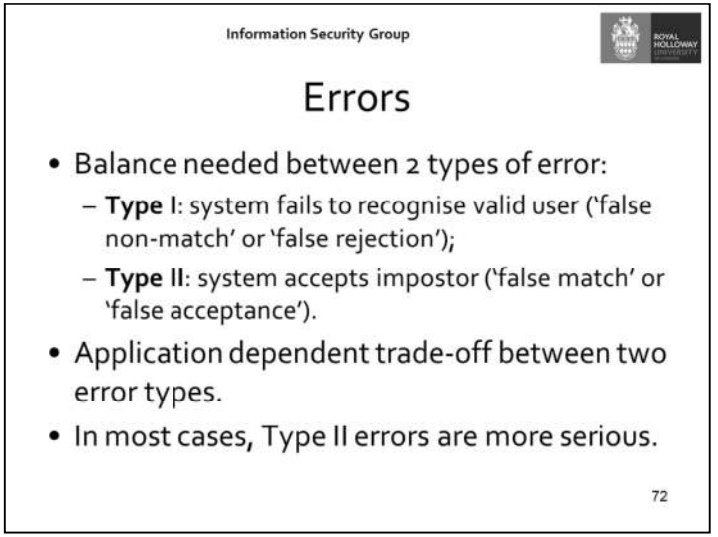
Biometric data is vulnerable during transmission; e.g. a measurement could be replaced by another, previously recorded, measurement, thereby enabling impersonation.



Unfortunately, biometric characteristics are not exactly the same every time they are collected. Your voice changes depending on the time of day, your emotional state, or you simply don't utter the same sentence EXACTLY the same way every time you do it. The fingerprint image captured by a sensor is not always the same because your finger may be greasy at one time and clean at another, the skin dryness changes or you simply put your finger on sensor in a slightly different position.

The matching module rates the similarity between the collected biometric data and the reference template. If the match score is above a tolerance (or acceptance) threshold, the claimant is accepted. If it is below the threshold, the claimant is rejected. Biometric systems can therefore generate two types of errors:

- **Type I error:** where the system fails to identify a valid user ('false non-match' or 'false rejection'),
- **Type II error:** where the system accepts an impostor ('false match' or 'false acceptance').



The value of the acceptance threshold is crucial to the performance of the system and depends on the security requirements of the application. If the threshold is relatively high (i.e. it is tough to meet), more valid users will be rejected (the false non-match rate will be high) but less impostors will be accepted (the false acceptance rate will be low). On the other hand, if the threshold is relatively low (i.e. it is easy to meet), more impostors will be accepted (the false match rate will be high) but less valid users will be rejected (the false non-match rate will be low). There is thus a trade off between these two types of errors; that is, the threshold setting will depend on the security requirements of the application.

In most applications, false acceptance (Type II) is the more important type of error, and this is minimised. False rejection (Type I) is not a security issue but more a usability issue, generating extra 'manual work' for the second line of defence (e.g. border guards).

Information Security Group

Match score threshold

- Acceptance (match score) threshold is crucial and application dependent:
  - threshold too high (i.e. tough) causes Type I errors (legitimate users rejected).
  - threshold too low (i.e. relaxed) causes Type II errors (impostors admitted).
- Equal error rate (EER): false non-match rate (FRR) = false match rate (FAR).

73

In any biometric scheme, the key parameter is the match score (acceptance) threshold. To summarise:

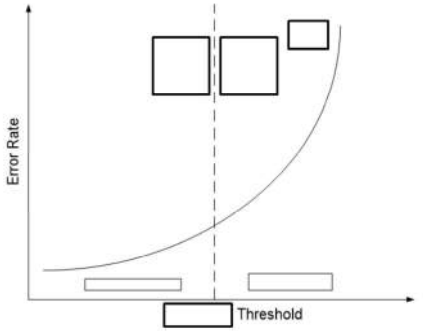
- if the tolerance threshold is relatively high (i.e. it is relatively tough to meet the threshold), then more valid users will be rejected (the false non-match rate will be high) but less impostors will be accepted (the false match rate will be low);
- if the threshold is relatively low (i.e. it is relatively easy to meet the threshold), fewer valid users will be rejected (the false non-match rate will be low) but more impostors will be accepted (the false acceptance rate will be high).

There is a trade off between these two types of errors that depends on the security requirements of the application.

The equal error rate is defined when the tolerance threshold is such that the false non-match rate is equal to the false match rate. This value is usually used as a criterion to compare different biometric systems, rather than as a means of setting the threshold. The actual tolerance threshold is determined by the needs of the given application.

Information Security Group

Type I / FRR error curve

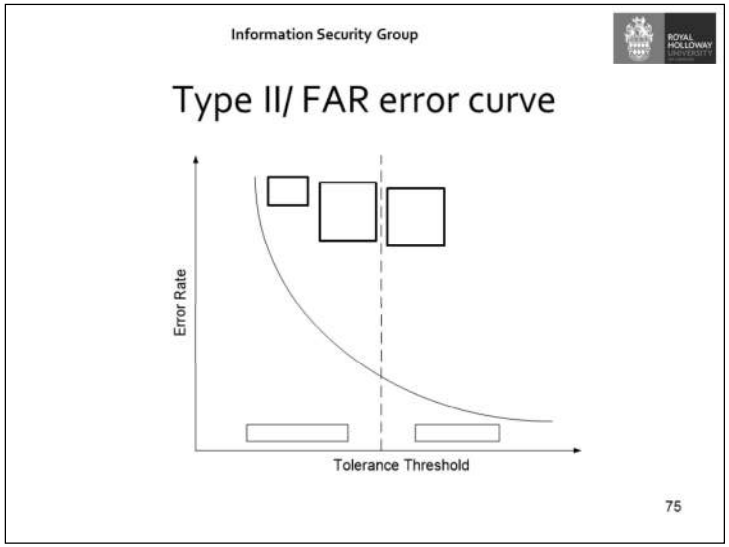


74

The curve shows the false rejection rate (FRR).

Threshold low: relatively low FRR = relatively few legitimate users rejected

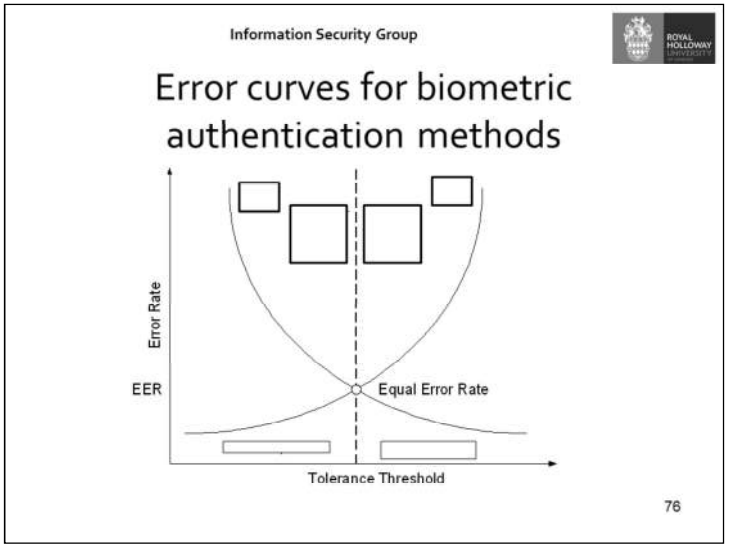
Threshold high: relatively high FRR = relatively large number of legitimate users rejected



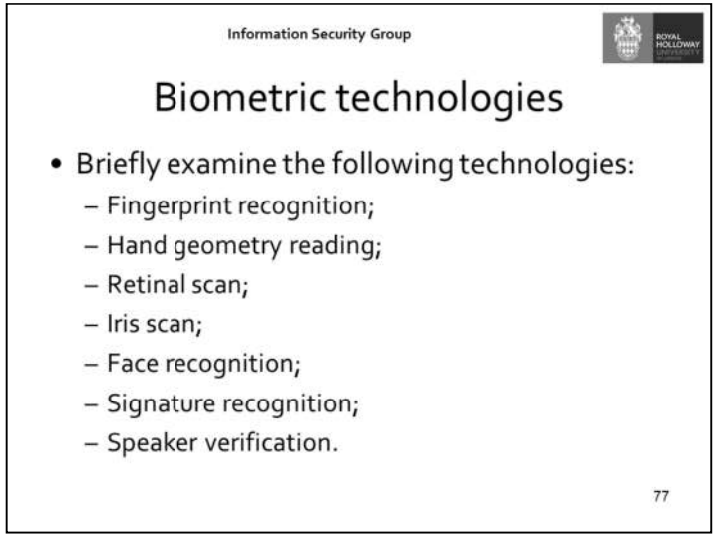
The curve shows the false acceptance rate (FAR).

Threshold low: relatively high FAR = relatively large number of impostors accepted.

Threshold high: relatively low FAR = relatively few impostors accepted.

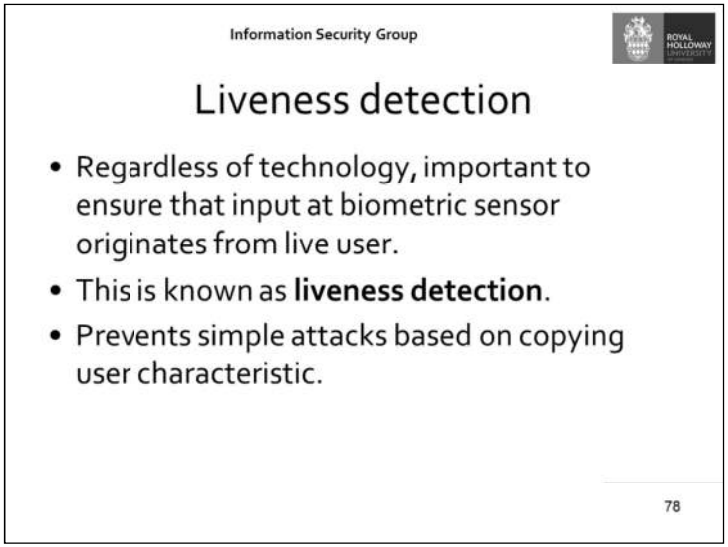


One way of measuring the performance of the system is to specify the error rate at which the FAR and the FRR are equal. This is known as the equal error rate. However, this is **not** a way of setting the threshold – as mentioned before, the choice of threshold will depend on the application.



We conclude by briefly looking at some of the most prominent biometric technologies.

We analyse their advantages and disadvantages.



Regardless of the technology in use, it is important to make sure that the input at a biometric sensor originates from a live user.

This is known as *liveness detection*.

It prevents simple attacks involving copying a user's characteristics.

Information Security Group

Fingerprint recognition I

- Ridge patterns on fingers uniquely identify people.
- Classification scheme devised in 1890s.
- Major features: arch, loop, whorl.
- Each fingerprint has at least one of the major features and many 'small features' (so-called **minutiae**).

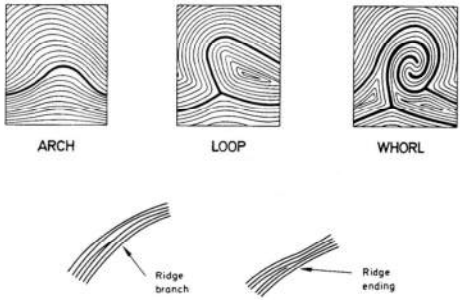
79

It has long been known that the skin ridge patterns on fingers (and elsewhere) can be used to uniquely identify people. A classification scheme to aid in matching people against records was devised in the 1890s.

This scheme is based on the recognition of certain types of 'feature'. The major features are the arch, loop and whorl. Each finger has at least one major feature. The small features, or minutiae, are even more important. For example the positions of ridge ends and ridge bifurcations are very important. There will be between 50 and 200 such minor features on every finger.

Information Security Group

Features of fingerprints



80

Examples of major and minor fingerprint features are shown.

In a machine recognition system, the reader must first minimise the rotation of the image with respect to the stored images. However, some residual rotation will always be present, and must be coped with.

The recognition system must then look for minutiae and compare with stored values. Minor injuries to finger ends can be a major problem.

Users can sometimes be unhappy about using such systems because of their association with crime detection.

Liveness detection is important because of the threats from detached real fingers and 'gummy bear' fingers, i.e. fake fingers made from gelatine – see:

http://www.theregister.co.uk/2002/05/16/gummi_bears_defeat_fingerprint_sensors/

One problem is that we all leave copies of our fingerprints wherever we go, which can potentially be used to manufacture fake fingers.

The basic steps in any automated fingerprint authentication are as follows.

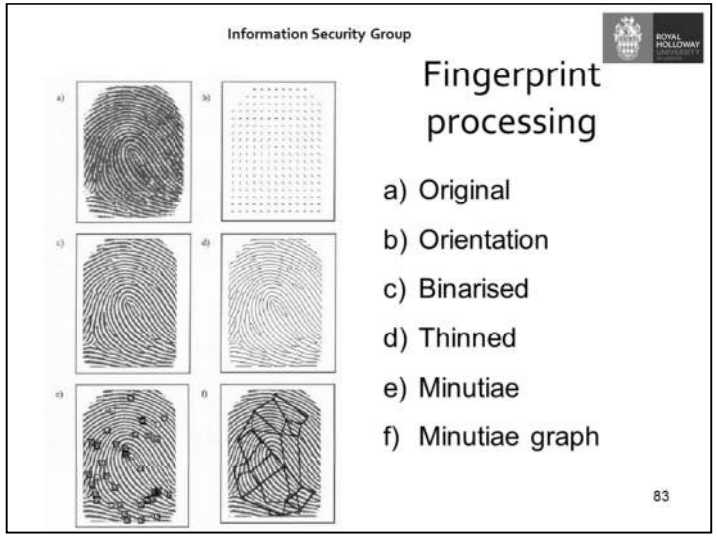
• Image acquisition: acquire image of fingerprint from sensor.

• Noise reduction: noise occurs due to dirty, dry, cut, wet finger tips. Filters are used to eliminate noise and enhance the ridges.

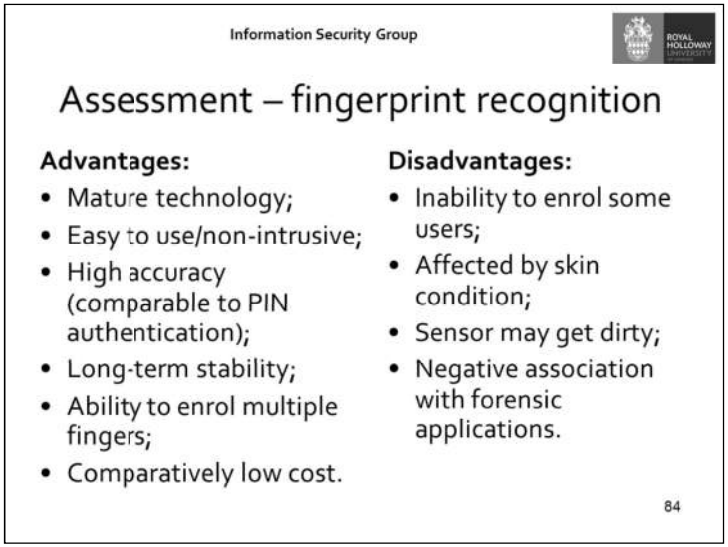
• Image enhancement: **binarisation** (transformation of grey-scale images in binary images) and **thinning** (reduction of the width of the ridges to a single pixel).

• Feature extraction: detection of termination points of the lines and ridge bifurcation.

• Matching: comparison between the extracted features and a previously stored biometrics template, analysing the neighbouring relationship of the minutiae.



The main steps in a fingerprint image processing procedure are shown.

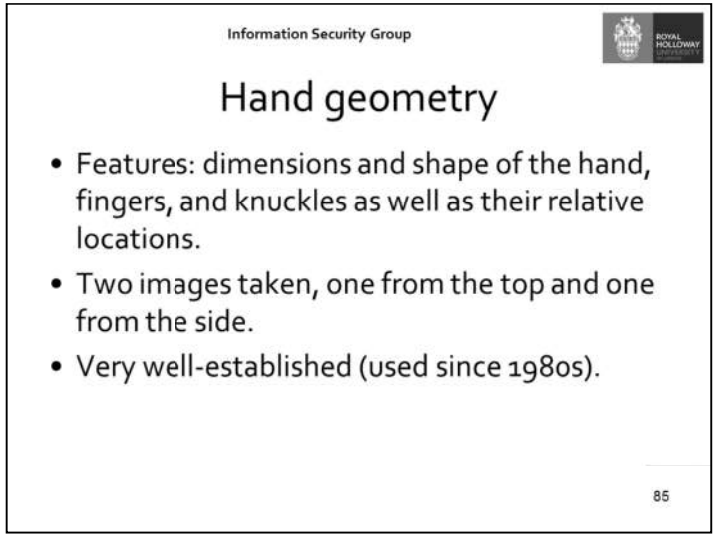


Advantages of fingerprint recognition include:

- It is a mature technology;
- It is easy to use/non-intrusive;
- It potentially has high accuracy (comparable to PIN authentication);
- Fingerprints have long-term stability;
- Multiple fingers can be enrolled;
- Readers are of comparatively low cost.

Disadvantages include:

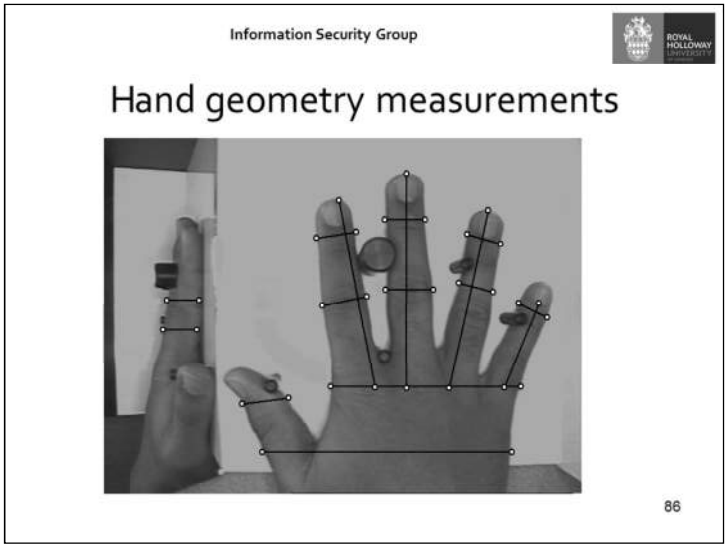
- Some users cannot be enrolled;
- Measurements can be affected by skin conditions;
- The sensor may get dirty;
- The technique has an association with forensic (criminal) applications.



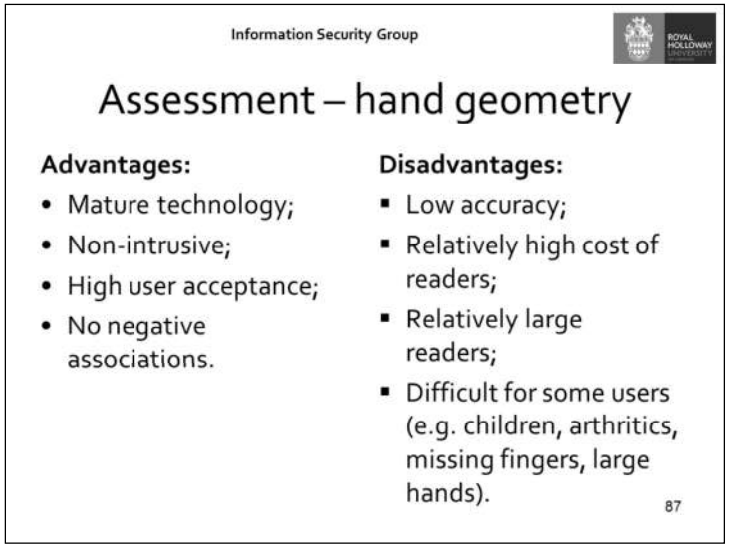
Hand geometry has been used for physical access control since the late 1980s. Hand geometry systems use two cameras to capture two different images, one from the top and one from the side, and determine the dimensions and shape of the hand, fingers, and knuckles and their relative position. A related biometric technique is finger geometry, which uses features only from a few fingers as opposed to the entire hand.

There is a useful link here:

http://www.biometrics.gov/Documents/HandGeometry.pdf



Hand geometry identification involves taking a range of measurements, e.g. of widths and lengths of fingers.

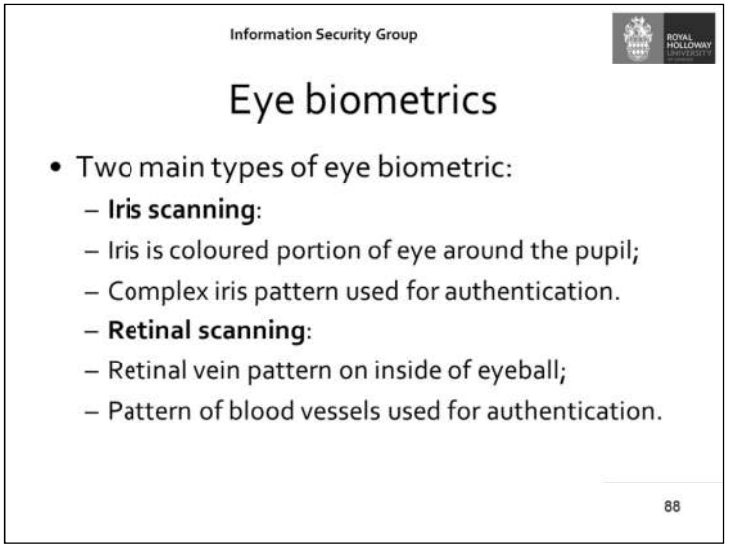


Advantages of hand geometry include:

- It is a mature technology;
- It is non-intrusive;
- It has high user acceptance;
- It has no negative associations.

Disadvantages include:

- Low accuracy;
- The readers have relatively high cost;
- Relatively large readers;
- It is difficult to use for some users (including children, and subjects with arthritis, missing fingers or large hands).



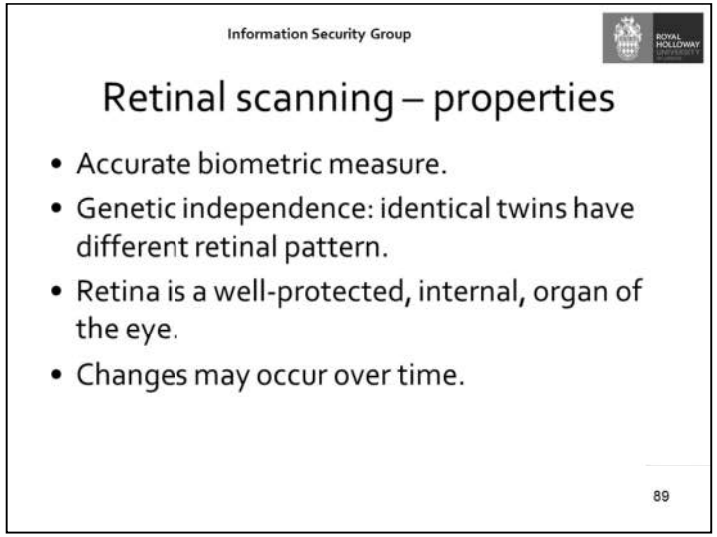
There are two main types of eye-based biometric: iris scanning and retinal scanning.

Iris scanning:

- The iris is the coloured portion of the eye surrounding the pupil; it is formally known as the trabecular meshwork;
- The complex iris pattern can be used for authentication.

Retinal scanning:

- This uses the retinal vascular pattern on the back of the inside of the eyeball;
- The pattern of blood vessels can be used for authentication.



The retinal pattern is highly distinctive so it is generally regarded as an accurate biometric measure.

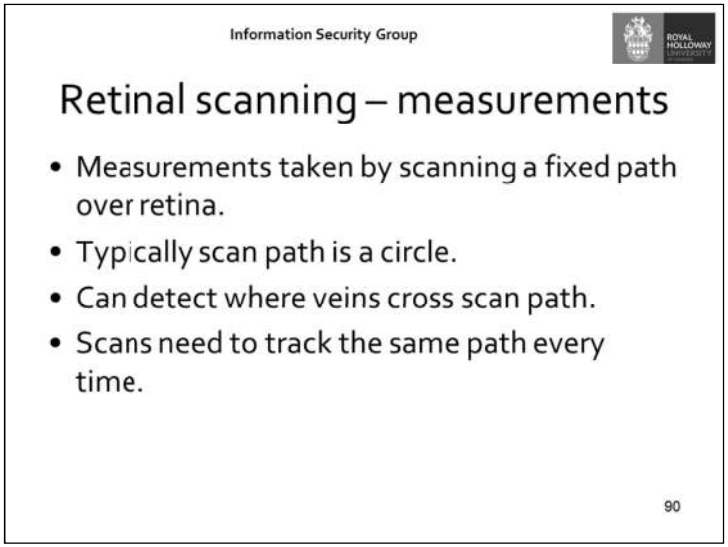
It is genetically independent, and hence even two identical twins have different retinal patterns.

It is a internal organ of the eye; therefore is well-protected and not affected by environmental conditions.

The pattern may, however, change during the life of a person.

There is a helpful wikipedia article here:

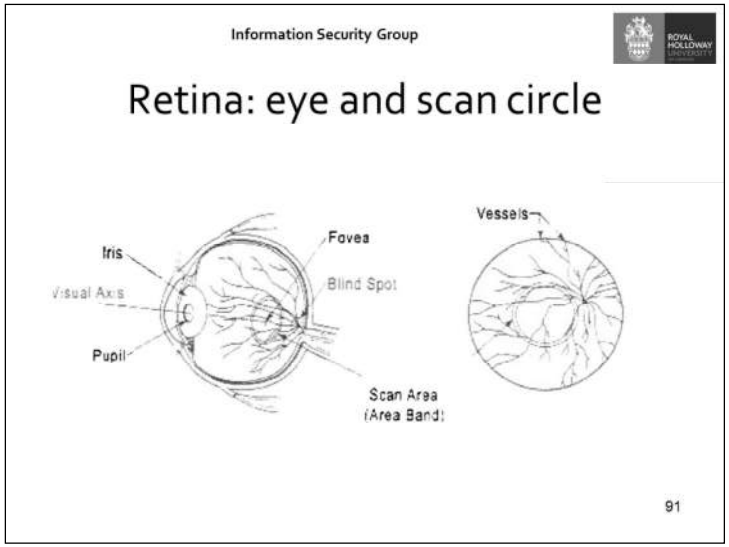
http://en.wikipedia.org/wiki/Retinal_scan



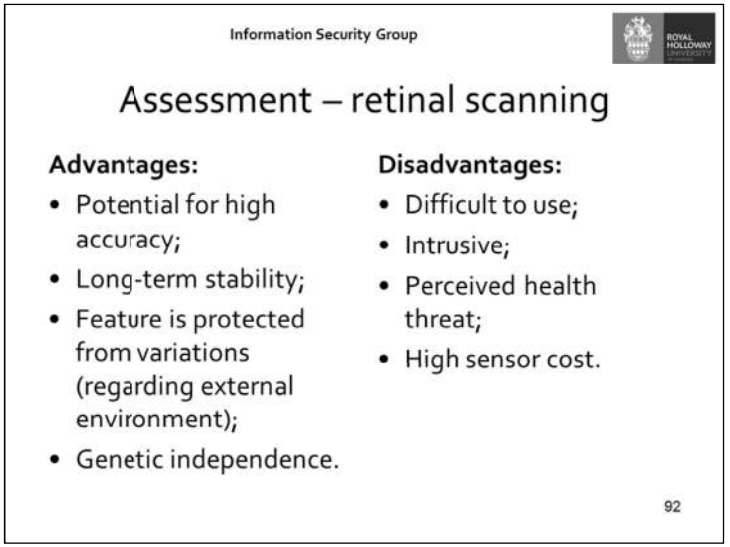
Retinal scan measurements are taken by scanning a fixed path over the retina. The typically scan path is a circle.

The scanner can detect where veins cross the scan path.

Obviously scans will need to track the same path every time.



The eye is shown on the left, and the scan circle on the retina is shown on the right.

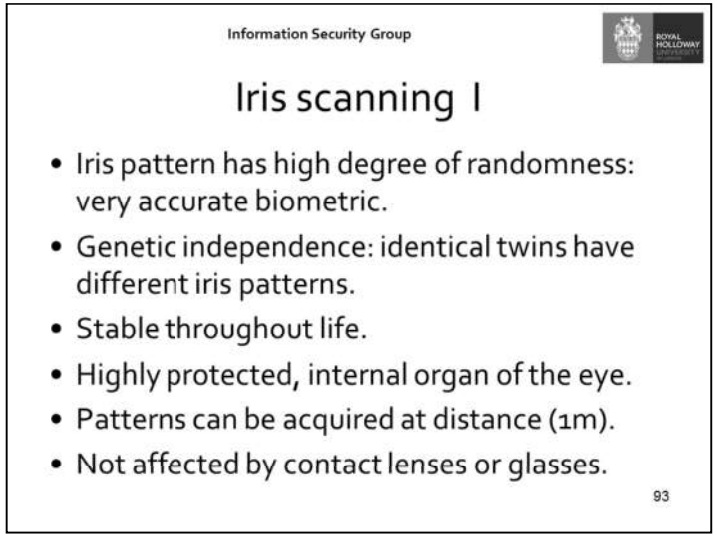


Advantages of retinal scanning include:

- It has the potential of high accuracy;
- The eye has long-term stability;
- The measured feature is protected from variations (regarding external environment);
- It has genetic independence.

Disadvantages include:

- It is difficult to use;
- Intrusive;
- Perceived health threat;
- High sensor cost.



The iris pattern is completely formed by the 7th month of pregnancy. It possesses the following properties:

• High degree of randomness: 244 degrees of freedom – it is a very accurate biometric measure.

• Genetic independence.

• Apparently stable throughout life.

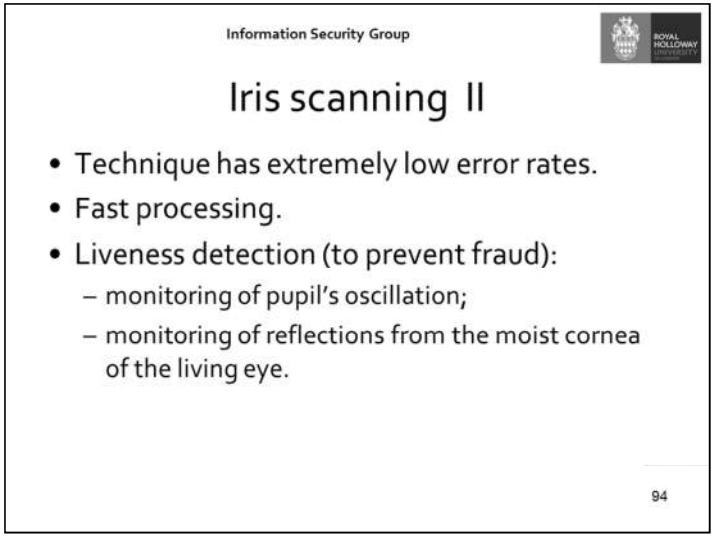
• Physically protected.

• Images can be acquired from a distance of 1m: as a result it is reasonably non-intrusive.

• The pattern can be encoded into 256 bytes making storage of template and matching process suitable for practical applications.

For more information, visit Daugman's website:

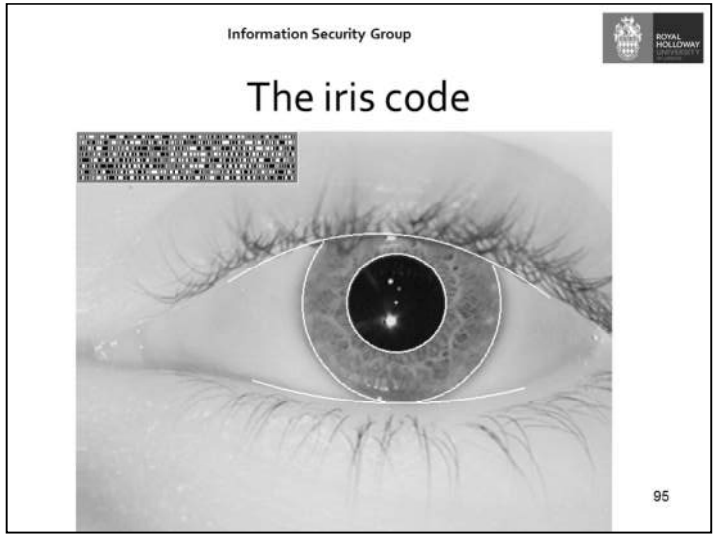
http://www.cl.cam.ac.uk/~jgd1000/



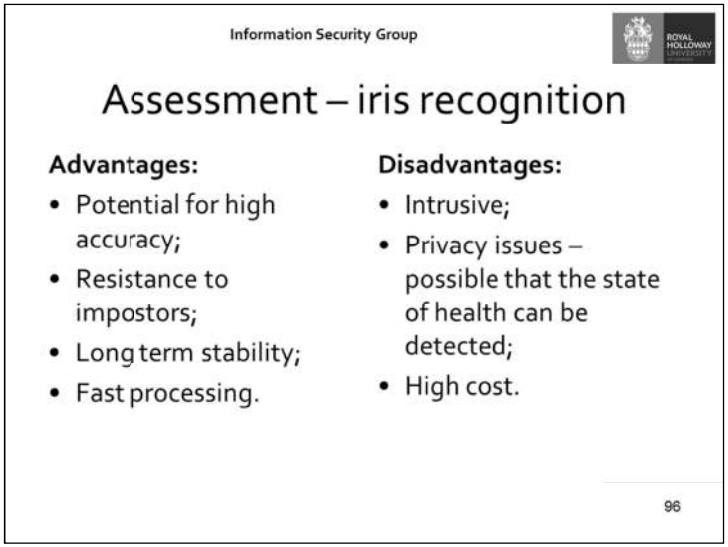
After the iris scan is processed so as to extract its distinguishing features (in the feature extraction module), an iris code is generated which consists of 256 bytes viewed as a two-dimensional barcode. This algorithm was developed by Daugman at Cambridge University.

Iris recognition has very low error rates and is the most accurate biometric in use today. Both processing and storage requirements are low, making it a most suitable technique for practical applications.

To prevent fraud and provide liveness detection, both the pupils' oscillations and the reflections of the moist cornea of the living eye are monitored to ensure that a living eye is present before the scanner (camera).



Shown is an example of an iris pattern, imaged at the distance of about 35 cm. The bit stream in the top left is the iris code.

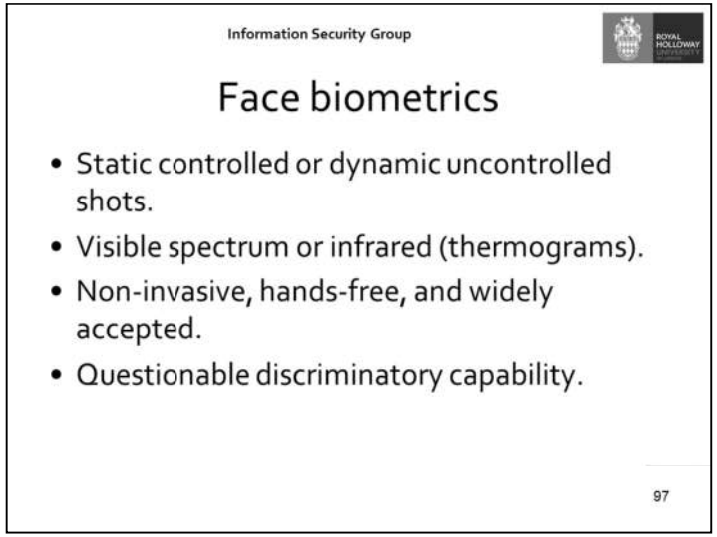


Advantages of iris scanning include:

- It potentially gives high accuracy;
- Resistance to impostors;
- Long term stability;
- Fast processing.

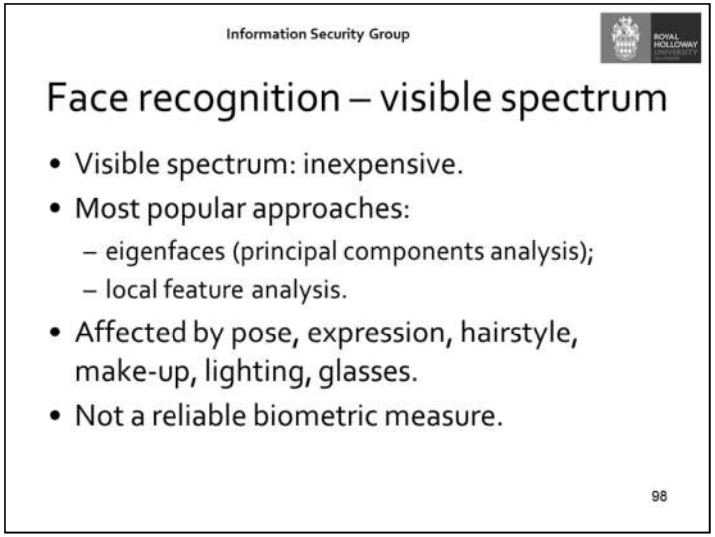
Disadvantages include:

- Intrusive;
- Potential privacy issues – some people think that the state of health of the subject can be detected;
- High cost.



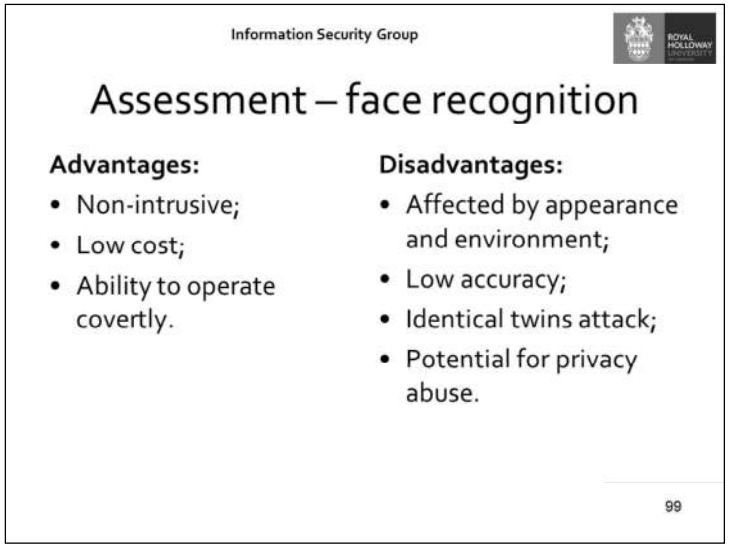
Personal identification by facial image can be done in a number of different ways. Images can be collected in the visible spectrum using an inexpensive camera or using infrared to capture facial heat emission (thermograms). Depending on the sophistication of the system, some applications may require a stationary user in order to capture the image while others may use motion image. The major advantage of facial identification is the fact that it is non-invasive, hands-free, and therefore widely accepted (and it is also how we usually recognise others – looking at their faces). However this biometric technique has questionable discriminatory capability and is generally not recommended for applications with high security requirements.

IY5512: Part 5



Face recognition typically uses an inexpensive digital camera to capture the facial image. The feature of the face used for identification/authentication is then extracted using one of several approaches. The most common approaches are known as 'principal components' and 'local feature' analysis. Principal components analysis models a particular face as a weighted combination of predefined 'basis' faces. Local feature analysis locates certain facial features, such as nose, mouth, eyes, etc, and assesses their individual geometry and their relative position.

This technique is highly affected by pose, expression, hairstyle, make-up, lighting, eyeglasses, facial hair, and is therefore not a reliable biometric measure.

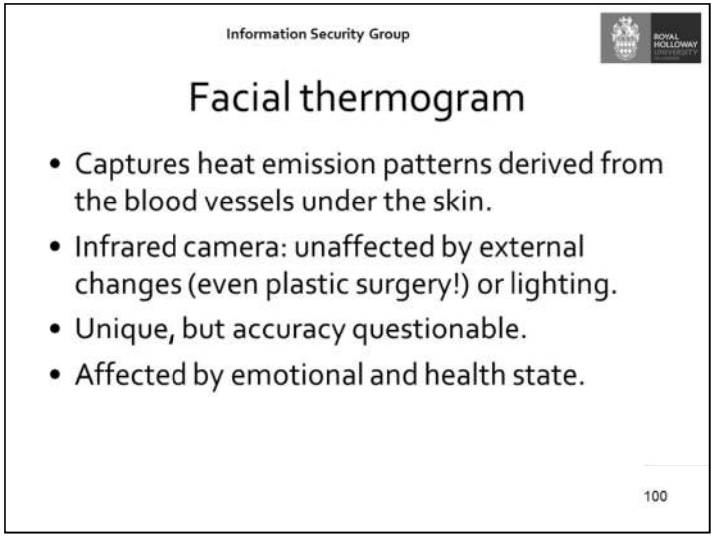


Advantages of face recognition include:

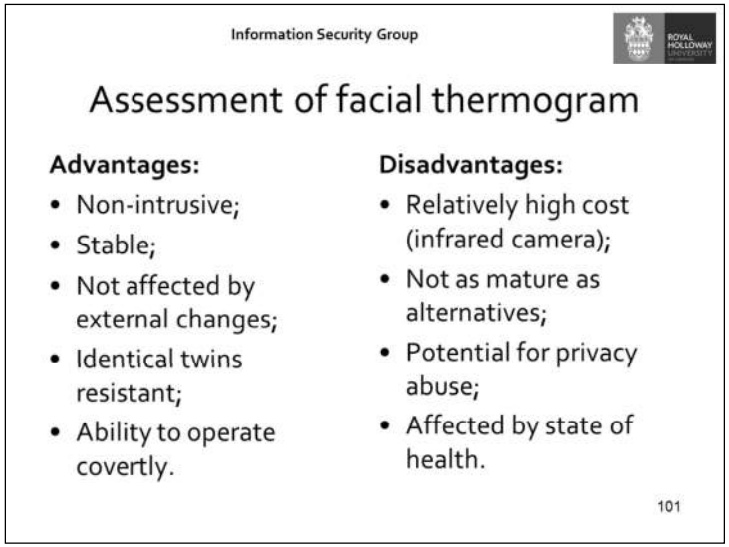
- Non-intrusive;
- Low cost;
- Ability to operate covertly.

Disadvantages include:

- Affected by appearance and environment;
- Low accuracy;
- Identical twins attack;
- Potential for privacy abuse.



A facial thermogram captures the heat emission patterns derived from the blood vessels under the skin. It uses an infrared camera and so is not affected by lighting (the thermogram can be done in the dark). It is also unaffected by external changes such as hair, eyeglasses, make-up, and even plastic surgery. Although the heat emission patterns are unique for each individual its accuracy is questionable (but is not as bad a biometric technique as face recognition). The heat emission are affected by emotional and health state.

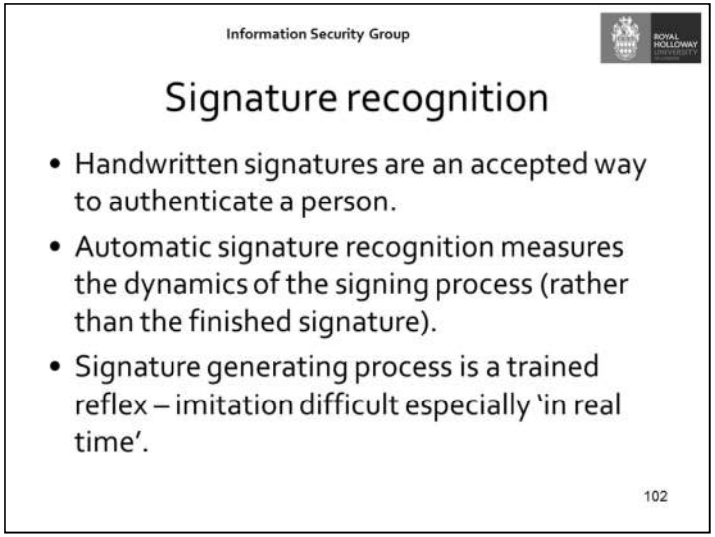


Advantages of facial thermograms include:

- Non-intrusive;
- Stable;
- Not affected by external changes;
- Identical twins resistant;
- Ability to operate covertly.

Disadvantages include:

- Relatively high cost (infrared camera);
- It is a relatively new technology;
- Potential for privacy abuse;
- Affected by state of health.

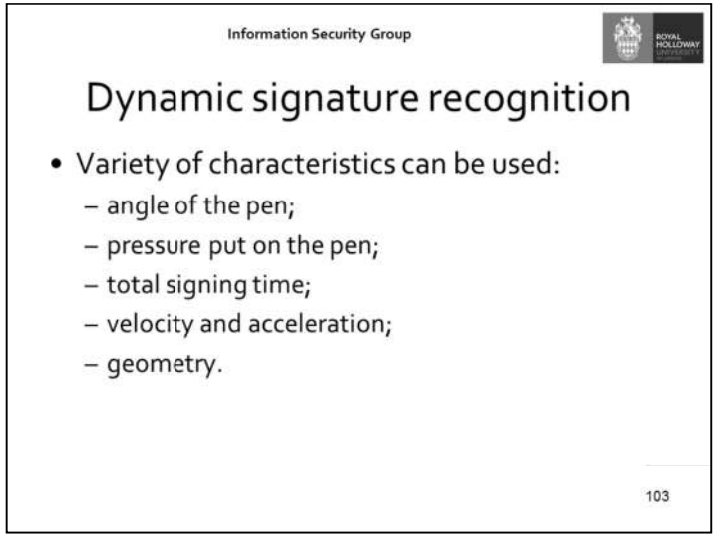


Hand-written signatures are in very wide use; the traditional verification technique is based on the visual inspection of a written signature. The process of generating a signature becomes a 'trained reflex', which is not subject to conscious muscular control. Thus signature imitation is difficult, especially at normal writing speed (i.e. in 'real time'); this explains why bank clerks often ask for documents to be signed while they are watching.

Automatic verification can either be based upon:

- signatures already produced, or
- observation of the actual signature process.

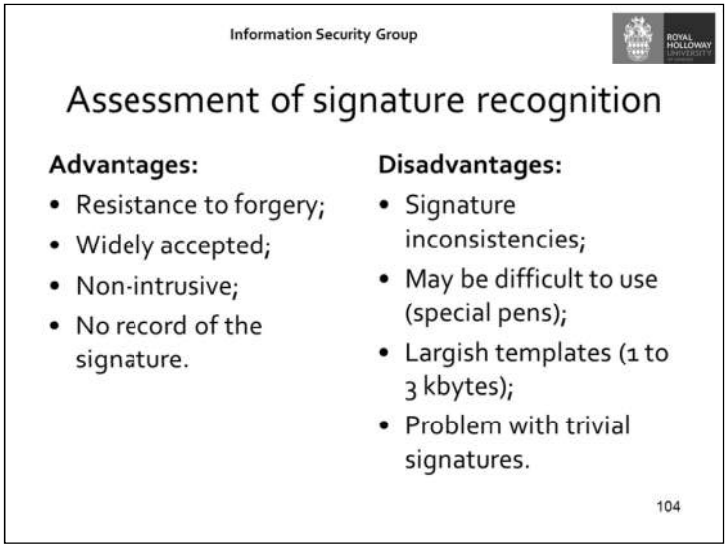
The second approach is more common, although the first approach has some relevance, for example for automatic cheque signature verification. However, the second approach offers greater security, since the first approach cannot detect copied or 'traced' signatures.



If the dynamics of the signature process can be used, i.e. as in the second approach, then forgery is clearly more difficult.

A variety of dynamic signature characteristics can be used, including:

- the angle of the pen;
- pressure put by user on the pen, and hence by pen on a surface while signing;
- the time to complete the signature;
- velocity and/or acceleration of the pen,
- how often the pen is lifted from the page;
- the way the signature looks (geometry).



Advantages of signature recognition include:

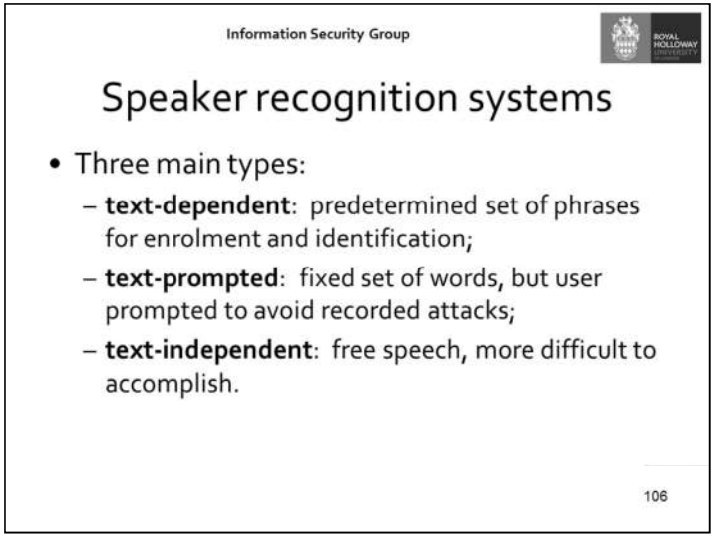
- It offers resistance to forgery;
- It is widely accepted;
- It is non-intrusive;
- There is no record of the signature.

Disadvantages include:

- Signature inconsistencies;
- It may be difficult to use if a special pen is required, or the writing angle is constrained;
- Largish templates (1 to 3 kbytes);
- Some users have 'trivial', i.e. very simple, signatures, which do not give enough discrimination.

The speech signal carries both linguistic and speaker acoustic patterns. The linguistic patterns represent what is being said and the speaker patterns reflect both anatomy (size and shape of the throat and mouth) and learned behavioral patterns (voice pitch, speaking style).

Speaker recognition requires heavy signal processing such as spectral analysis, periodicity, filtering, etc.

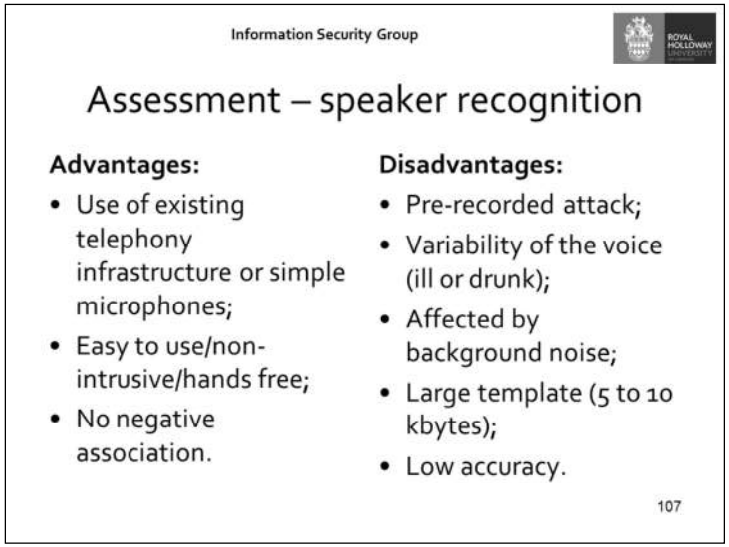


Speaker recognition systems can employ three types of spoken input:

• **text-dependent**: uses a predetermined set of phrases for enrolment and identification. Most speaker verification applications use this mode of operation.

• **text-prompted**: asks the user to repeat specific words, phrases, or numbers. It is used when there is a concern about taped recorded impostors.

• **text-independent**: no fixed set of words, but free flowing speech. More difficult to accomplish than the other modes.

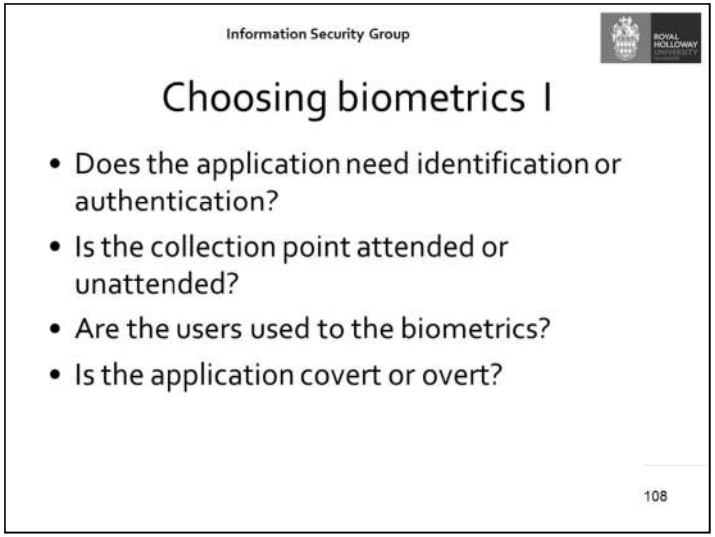


Advantages of speaker recognition include:

- Use of existing telephony infrastructure or simple microphones;
- Easy to use/non-intrusive/hands free;
- No negative association.

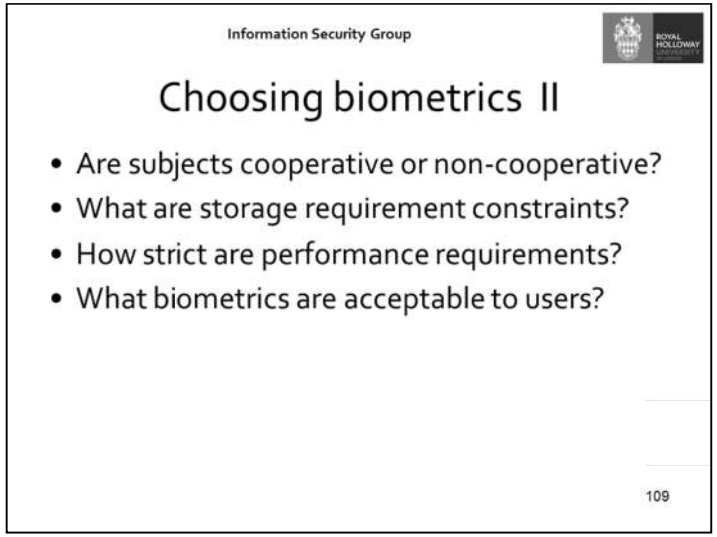
Disadvantages include:

- Pre-recorded attack;
- Variability of the voice (ill or drunk);
- Affected by background noise;
- Large template (5 to 10 kbytes);
- Low accuracy.



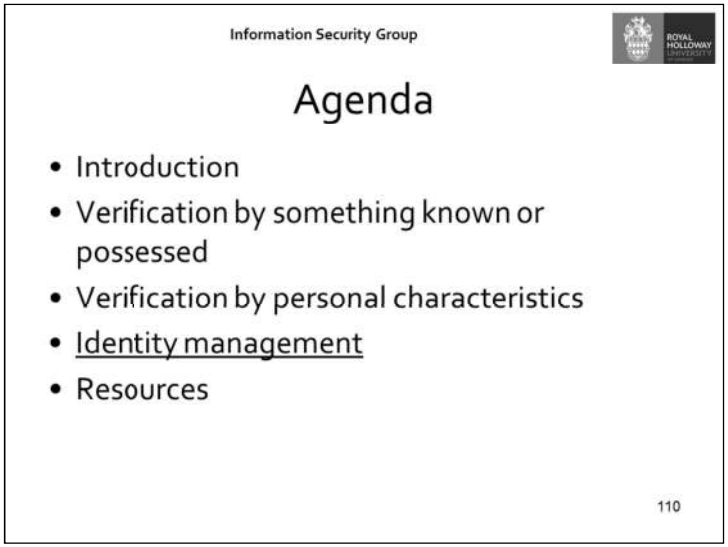
Each biometric technology has its strengths and its limitations. No single biometrics is expected to meet the needs of all applications effectively. The choice of biometrics employed in a personal identification system depends on the requirements of the application and the profile of the target population. Some general properties of biometrics-based identification systems can be described by the following questions:

- Does the application need identification or authentication? Authentication achieves better performance than identification for the same biometrics – authentication systems may therefore resort to less accurate biometrics. Applications requiring the identification of a subject from a large database of identities need relatively more accurate biometrics.
- Is the collection point attended (semi-automatic) or unattended (completely automatic)? An application which can afford a human operator at the acquisition point makes forgery more difficult.
- Is the application covert or overt? Not all biometrics can be captured without the knowledge of the subject to be identified – so this reduces the options for covert applications.
- Not attended? High maintenance.

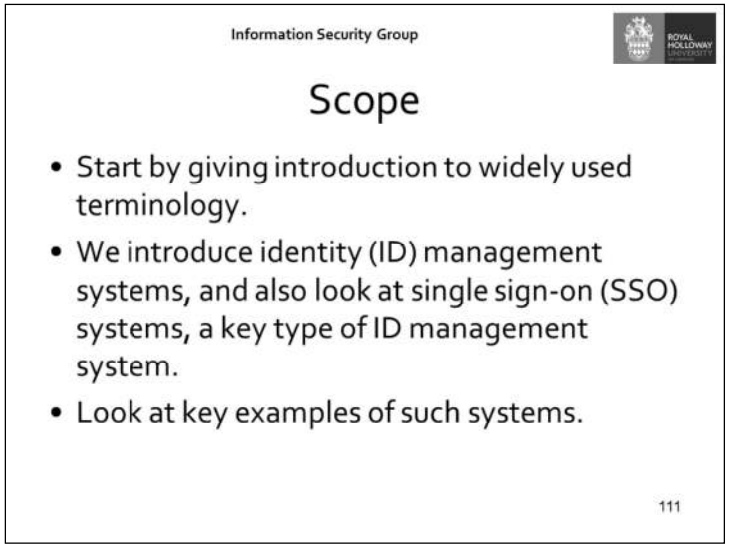


Further questions that need to be considered include the following.

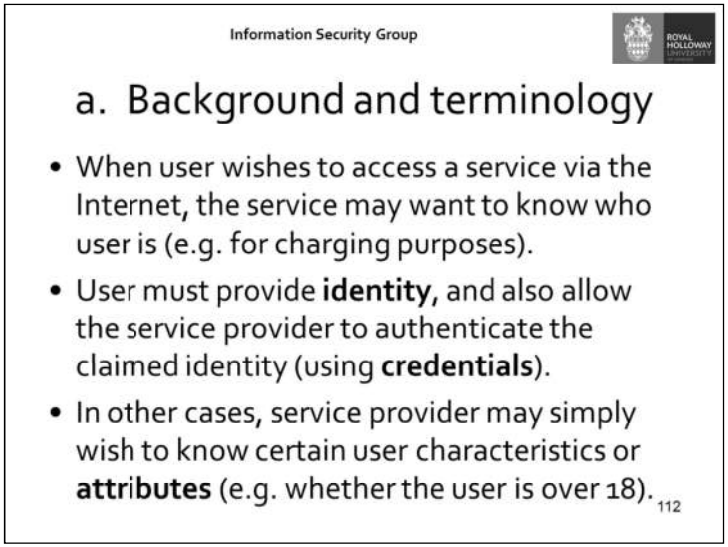
- Are the subjects cooperative or non-cooperative?
- What are the storage requirement constraints?
- How strict are the performance requirements?
- What types of biometrics are acceptable to the users?



We conclude our discussion of user authentication by introducing the topic of identity management.



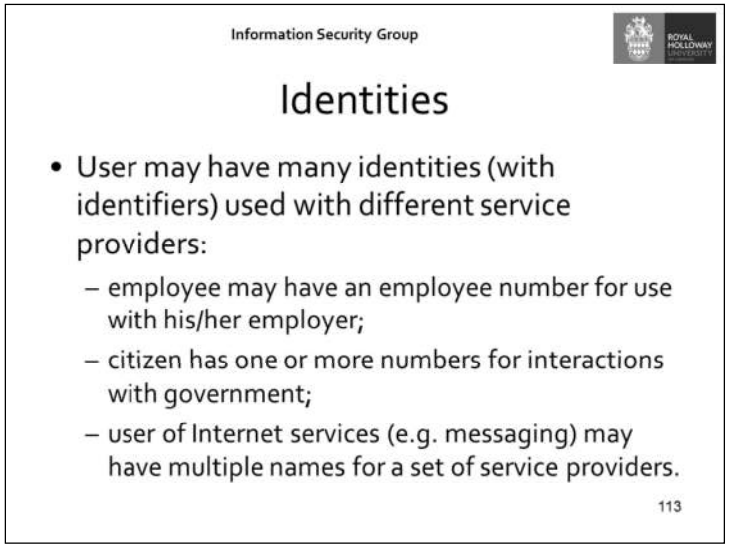
We conclude our discussion of user authentication by introducing identity management systems. We start by introducing a range of identity/privacy-related terminology, before introducing identity management systems. In practice, identity management systems are often implemented as single sign-on (SSO) systems, a key category of identity management systems. We look at key examples of such systems.



When a user wishes to make use of an Internet service, the service will typically wish to be sure of the identity of the user (e.g. to ensure the user is authorised to access the service and/or for charging purposes). This requires the user to provide an *identity*, and also to give the means for the service provider to *authenticate* the claimed identity (that is, verify in some way that the user is entitled to use the provided identity). This is achieved by the user deploying one or more *credentials*.

In other cases, the service provider may simply wish to know certain that the user has certain characteristics or *attributes* (e.g. that they are over 18).

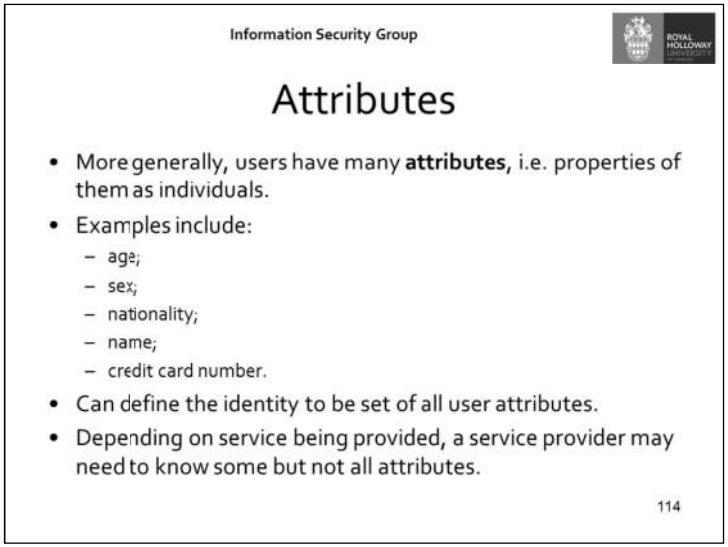
Note that the discussion here is of Internet identity management – identity management can also apply in other networking environments, e.g. corporate networks.



A user may have many identities (with associated identifiers) for use with different service providers. For example:

- an employee may have an employee number for use with his/her employer;
- a citizen has one or more numbers for interactions with government;
- a user of Internet services (e.g. messaging) may have multiple names, each used with one or more service providers.

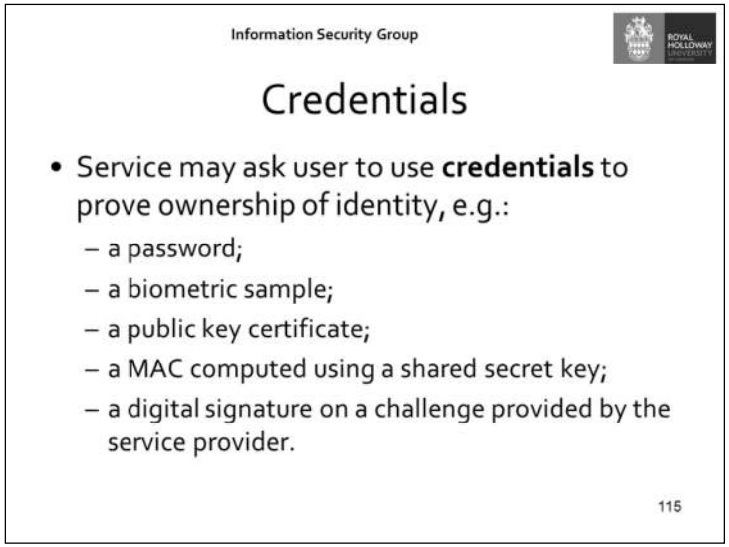
IY5512: Part 5



More generally, users have many *attributes*, i.e. properties that they possess as individual human beings. Examples of such attributes include:

- age;
- sex;
- nationality;
- first name;
- last name;
- credit card number;
- passport number;
- ethnicity;
- religion;
- ...

We can define the identity of a user to be the collection of all the user's attributes. Depending on the service that is being provided, a service provider may need to know some but not all of a user's attributes. For example, in some cases a unique name may be required, whereas in others merely selected attributes, e.g. their age, may be sufficient.



To enable a service provider to authenticate a user as a legitimate holder of an identity (or, more generally, a set of attributes), the user may be required to use one or more *credentials*, and possibly to prove that they 'own' the credentials. Possible credentials include:

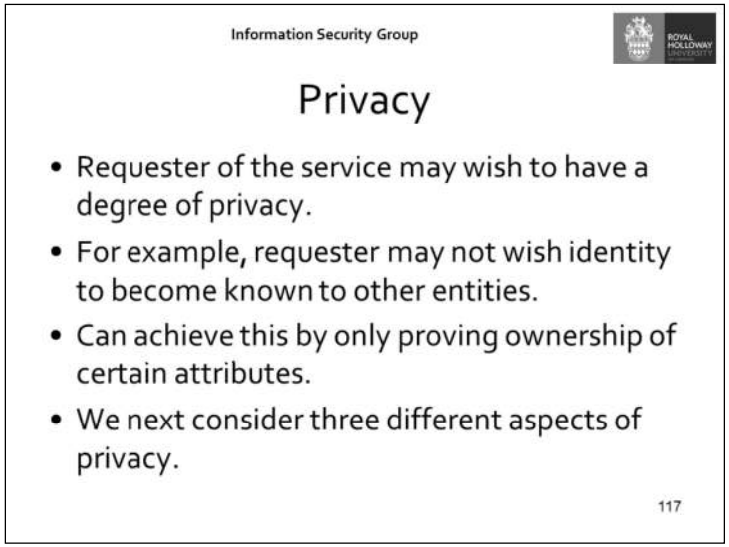
- a password;
- a biometric sample;
- a public key certificate;
- a MAC (message authentication code) computed using a shared secret key;
- a digital signature on a challenge provided by the service provider, computed using the user's private signing key.

IY5512: Part 5

Once an entity has been authenticated, the service provider needs to decide whether or not to grant the requested service. This is referred to as *authorisation*, i.e. is the holder of this identity (or a user with this set of attributes) authorised to access this service?

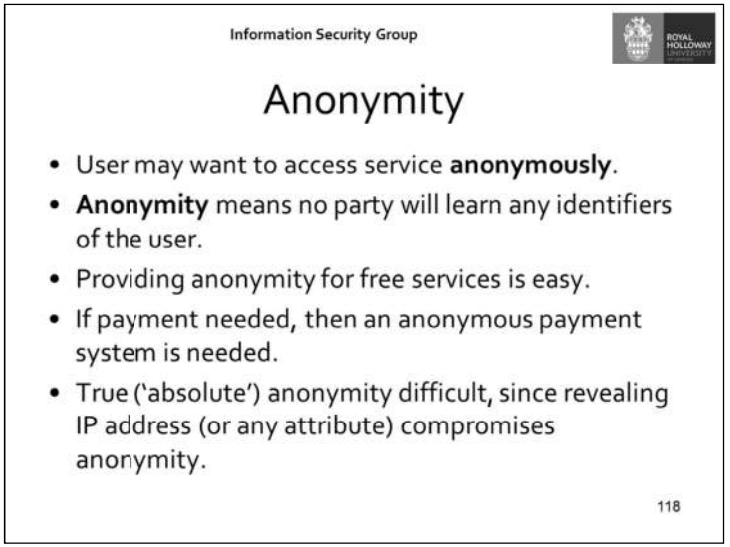
This could, for example, be supported using server-held Access Control Lists (ACLs). Alternatively, the requester of service might provide a statement signed by the resource-owner, saying that requester should be granted access (often called an *authorisation statement*).

Authorisation (access control) is the main focus of part 6 of this course.



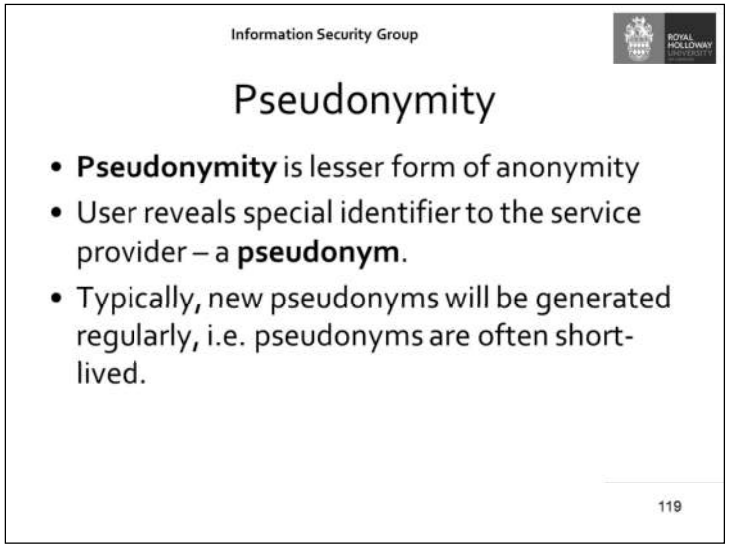
In some cases, the requester of the service may wish to have a degree of privacy provided. For example, the requester may not wish his/her identity to become known to other entities. This could be achieved by the service requester only proving ownership of certain attributes.

We next consider three different aspects of privacy.



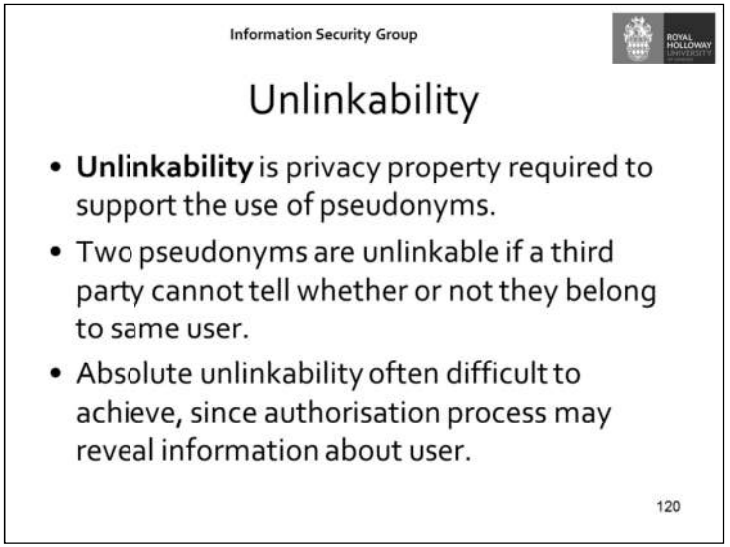
A user may wish to be able to access a service in an *anonymous* way. *Anonymity* means that no party will learn any of the identifiers of the user. Providing anonymity for free services without any authorisation requirements is relatively simple. If payment is needed, then an anonymous payment system is needed, e.g. cash or e-cash.

True ('absolute') anonymity is difficult to achieve, since even revealing an IP address (or any attribute) to some extent compromises anonymity.



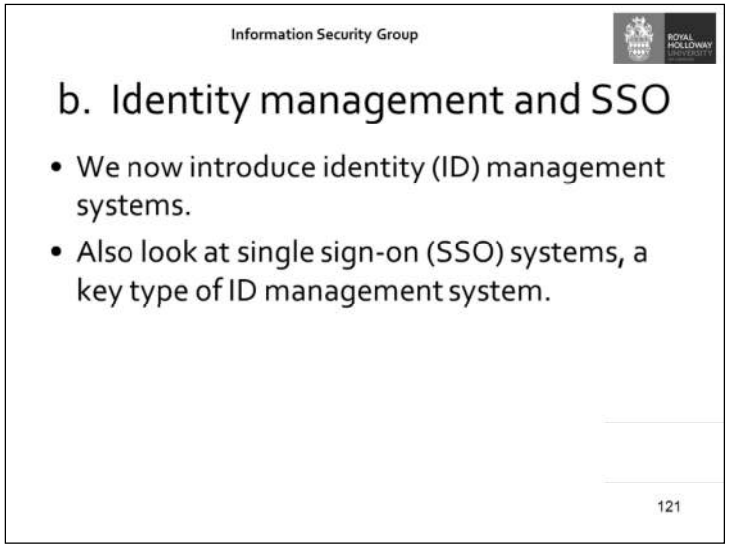
Pseudonymity is a lesser form of anonymity, in which the user reveals a special type of identity to the service provider known as a *pseudonym*. Typically, new pseudonyms will be generated regularly, i.e. pseudonyms are often short-lived.

An example of short-lived pseudonyms is provided by the Temporary Mobile Subscriber Identities (TMSIs) used by GSM (and also by 3G and 4G mobile systems).

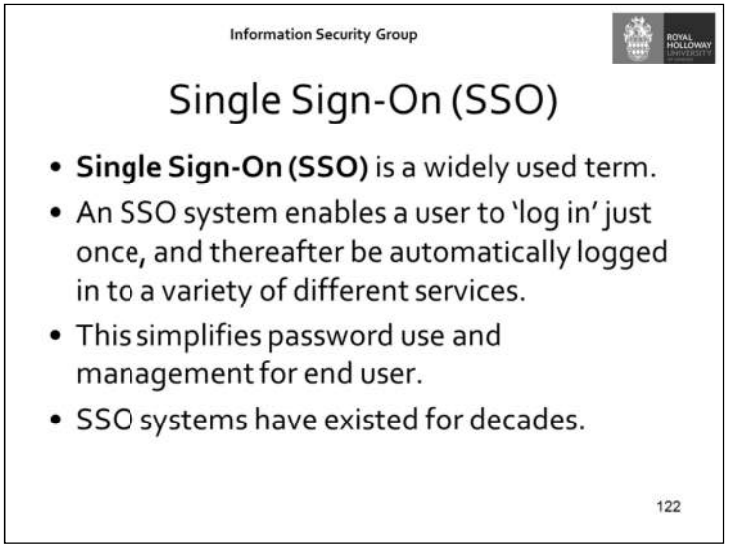


Unlinkability is a privacy property required to support the effective use of pseudonyms. Two pseudonyms are unlinkable if a third party cannot tell whether or not they belong to the same user.

In practice, absolute unlinkability is often difficult to achieve, since the authorisation process may reveal information about the user. This is because the access rights given to a user may help to identity the user.

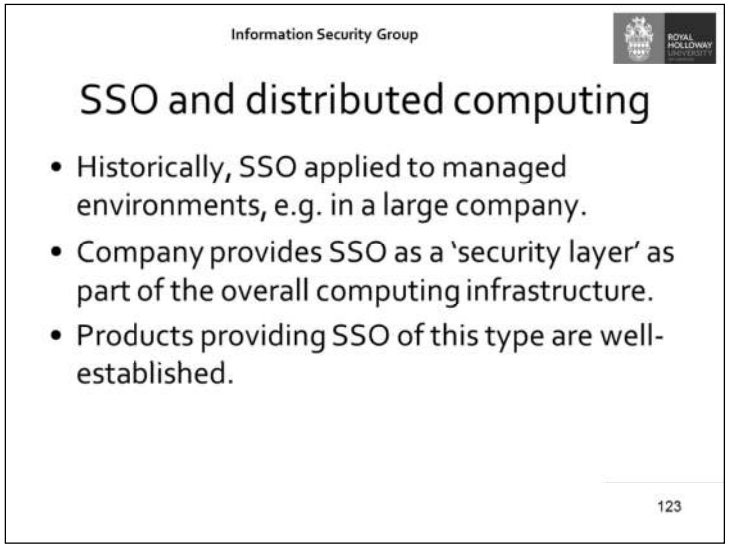


We now introduce identity (ID) management systems. Also look at single sign-on (SSO) systems, a key type of ID management system.

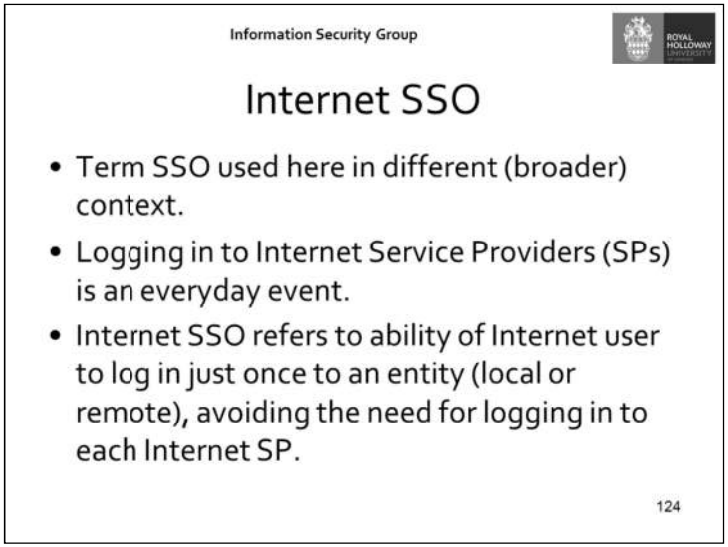


Single Sign-On (SSO) is a widely used term. An SSO system enables a user to 'log in' just once, and thereafter be automatically logged in to a variety of different services. This greatly simplifies credential management (including password management) and credential use for the end user. In a general context, SSO systems have existed for decades.

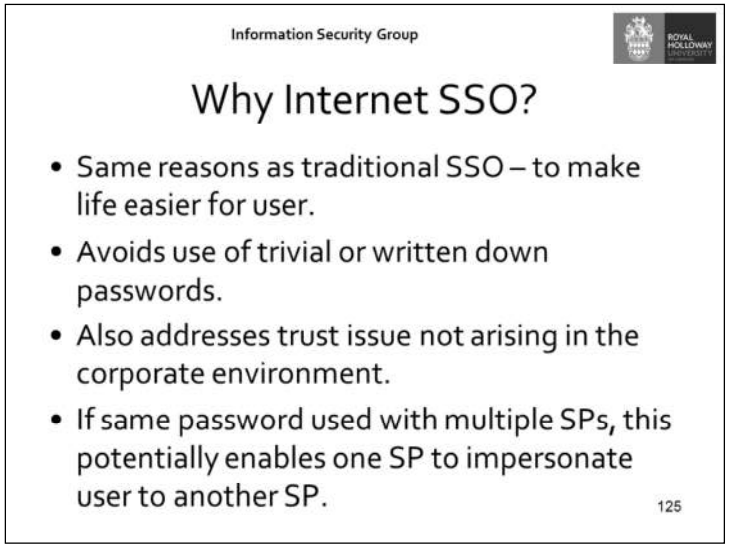
Windows Server provides an SSO service in managed domains. This is discussed in part 7a of this course. Windows server is built on Kerberos (as discussed in IY5511), which is a type of identity management protocol.



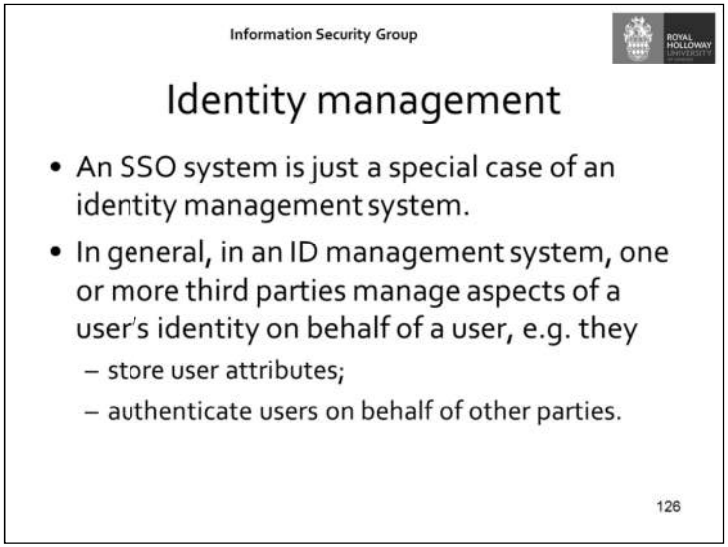
Historically, SSO is a service that has been provided in managed environments, e.g. within a large company. The company would typically provide SSO as a 'security layer' as part of the overall computing infrastructure. Products to provide SSO of this type are well-established (including Windows, as we discuss in part 7b of the course).



In this course, the term SSO is used in a somewhat different (broader) context. Logging in to Internet Service Providers (SPs) is an everyday event. Internet SSO refers to the ability of an Internet user to log in just once to an entity (local or remote), which then avoids the need for logging in to every Internet SP.



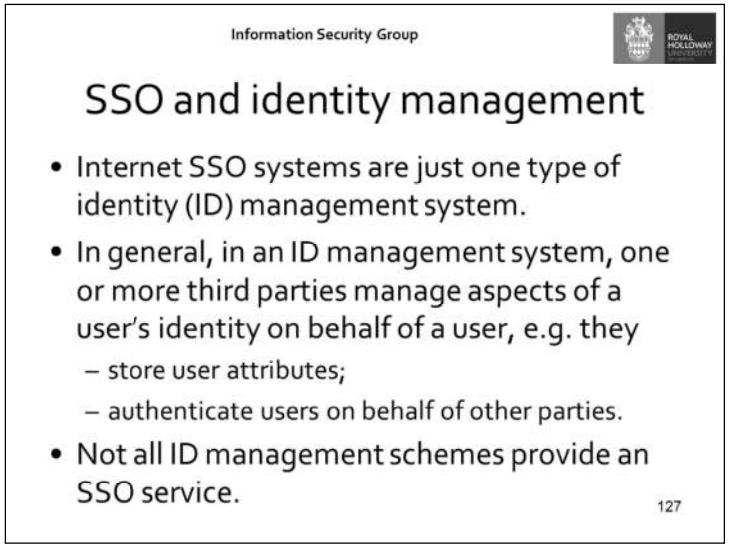
Internet SSO has been introduced for the same reasons as traditional SSO – to make life easier for the user. However, apart from avoiding the use of trivial or written down passwords, it also addresses a trust issue not arising in the corporate environment. That is, if the same password is used with multiple SPs, then this potentially enables one SP to impersonate the user to another SP. This is clearly an undesirable scenario.



An SSO system is just a special case of an identity management system.

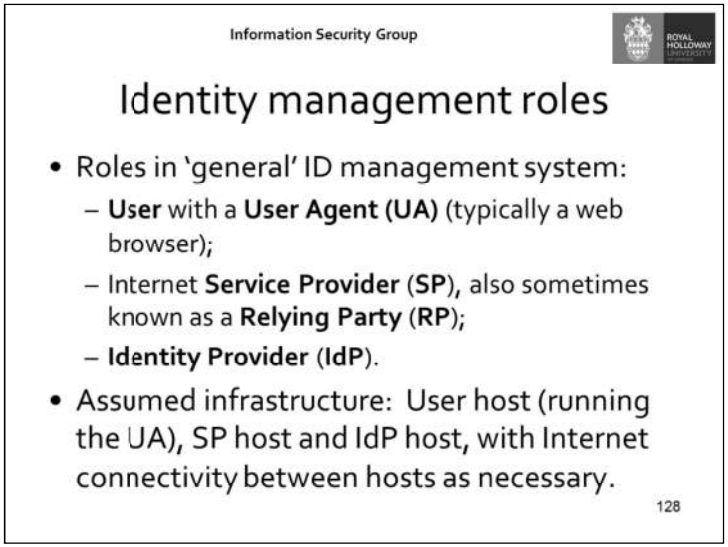
In general, in an ID management system, one or more third parties manage aspects of a user's identity on behalf of a user, e.g. they

- store user attributes;
- authenticate users on behalf of other parties.



Internet SSO systems are today regarded as just one type of identity (ID) management system. In general, an identity management system provides a framework within which third parties can manage aspects of a user's identity on behalf of a user; in particular they can store user attributes, and authenticate users on behalf of other parties.

It is important to note that not all ID management schemes provide an SSO service.



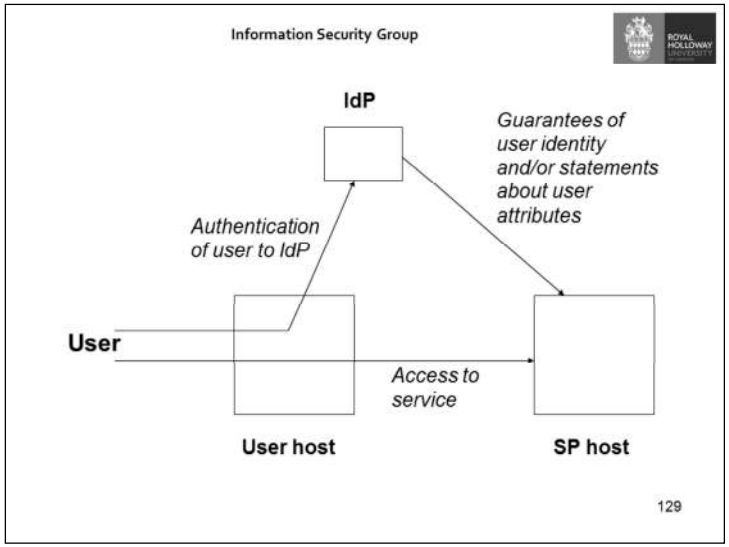
The principal roles in a 'general' Internet ID management system are:

• the *User*, who is equipped a with a means of accessing services in the form of a *User Agent (UA)* (typically a web browser),

• an Internet *Service Provider (SP)*, sometimes also referred to as the *Relying Party (RP)*, which is providing services that the suer wishes to access;

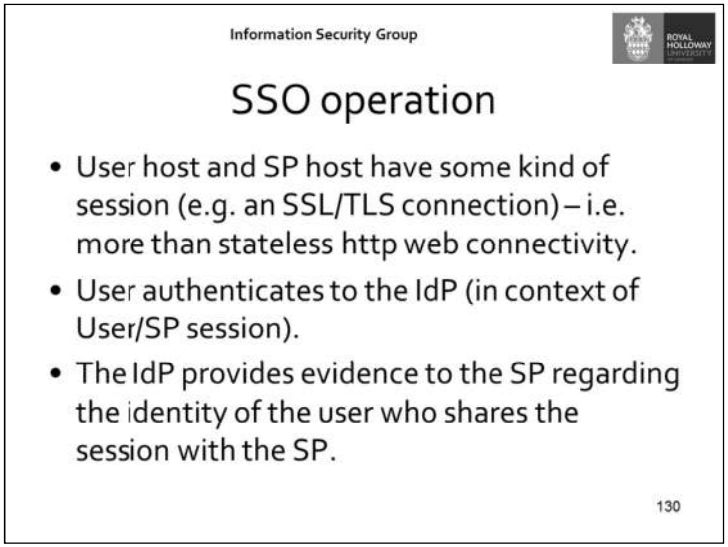
• an *Identity Provider (IdP)*, which stores information about the user, e.g. means to authenticate the user.

The assumed infrastructure to support ID management contains: a User host, an SP host and an IdP host, with Internet connectivity between hosts as necessary.



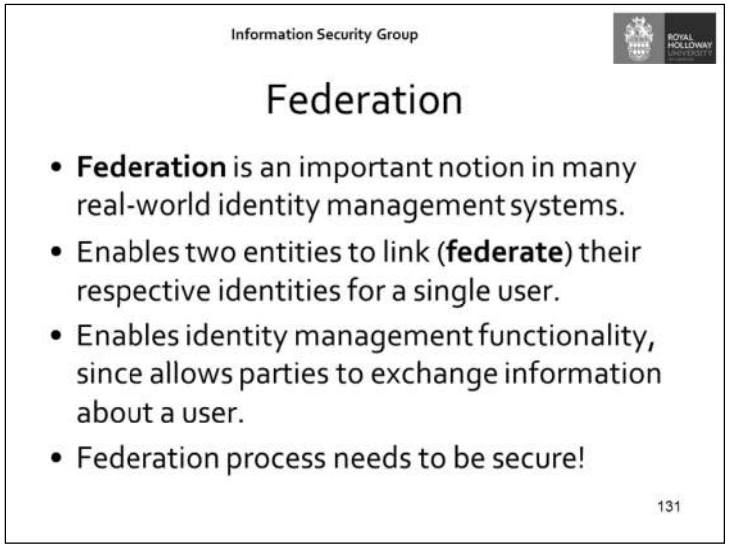
The diagram shows a simple model for an Internet identity management system.

IY5512: Part 5



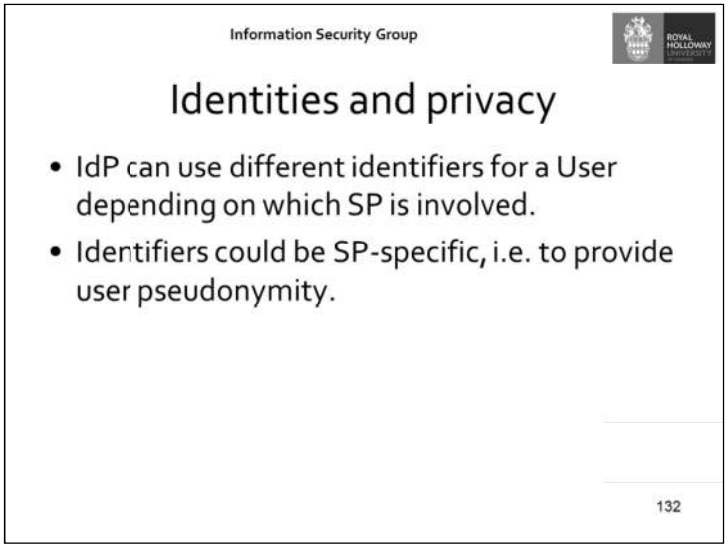
The User host and the SP host are assumed to share some kind of session (e.g. an SSL/TLS connection) – i.e. more than stateless http web connectivity. The User authenticates to the IdP (in the context of a User/SP session). The IdP provides evidence to the SP regarding the identity of the user with whom the SP shares the session.

The reference to 'in the context of a User/SP session' captures the need for a binding between the user communicating with the SP and the user which has been authenticated to the IdP. How this binding is achieved is system-dependent.



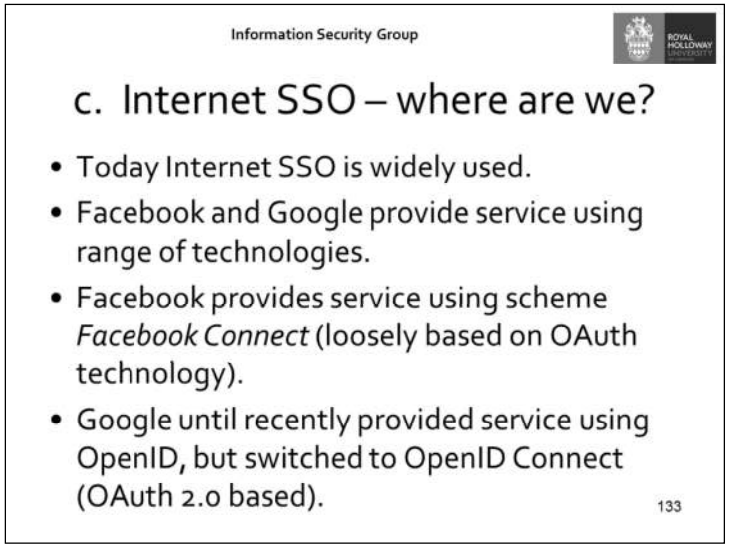
Federation is an important notion in many real-world identity management systems. It enables two third parties, e.g. an RP and an IdP, to link (*federate*) their respective identities for a single user.

Federation enables identity management functionality, since it allows third parties to exchange information about a user and to know who they are talking about. Obviously, the federation process needs to be secure, or else the basis of identity management can be undermined.



An IdP can use different identifiers for a User, depending on which SP is involved.

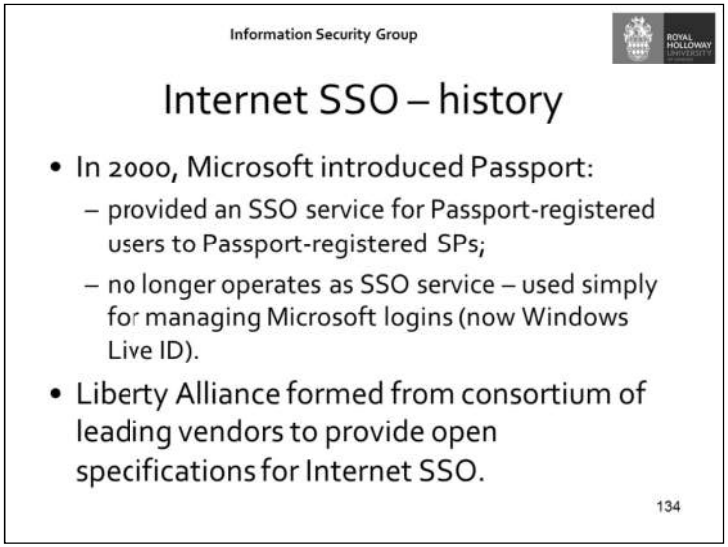
These identifiers could be SP-specific, i.e. to provide user pseudonymity. That is, the IdP could use a different pseudonym for a user with each SP.



Today Internet SSO is very widely used, in particular through the services provided by Facebook and Google. They have provided SSO services for several years using a range of technologies.

Facebook provides its SSO service using a scheme called *Facebook Connect*, which is loosely based on OAuth technology.

Google until recently provided its SSO service using the OpenID technology, but has now switched to use of OpenID Connect (which is OAuth 2.0 based).

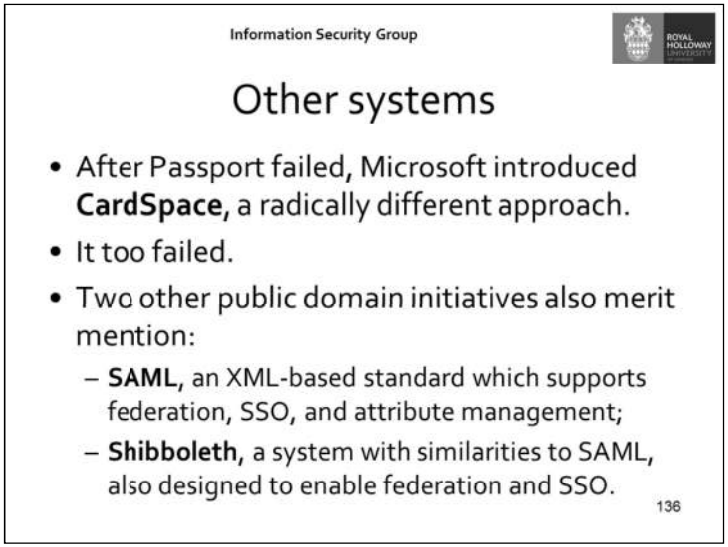


Around 2000/01, Microsoft introduced the Passport scheme, which provided an SSO service for Passport-registered users to Passport-registered SPs. It was widely criticised and suffered from poor take-up. It is still in operation, but it no longer operates as an SSO service; it is just used as a means of managing Microsoft logins, and has been transformed into Windows Live ID.

As a rival initiative, the Liberty Alliance was formed from a consortium of leading vendors to provide open specifications for Internet SSO.

The *Liberty Alliance* was a consortium of companies interested in SSO and identity management. It published a series of specifications for an 'open' XML-based SSO system.

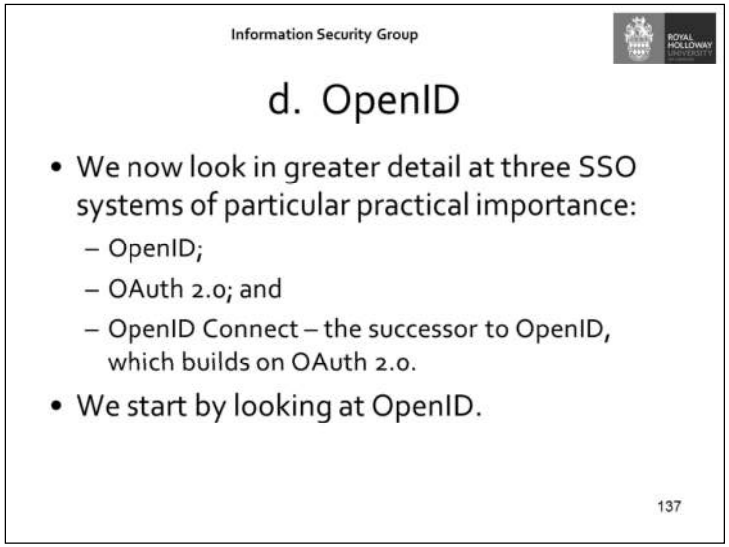
The *Kantara Initiative* is the successor to the Liberty Alliance, and has inherited its specifications.



After Passport failed, Microsoft introduced CardSpace, a radically different approach. Interestingly, CardSpace was not an SSO system and did not support federation. It's main aim was to enable users to retain control over their attributes and over use of their credentials. It too failed. CardSpace had an open source analogue, known as *Higgins*.

Two other public domain initiatives also merit mention:

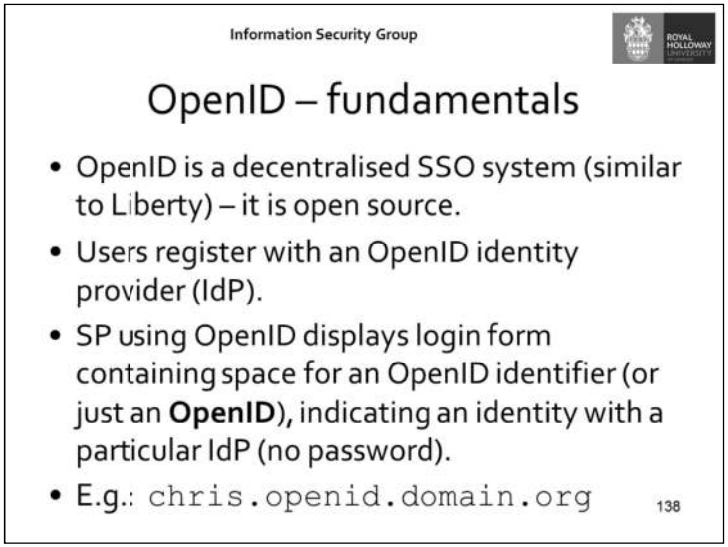
- SAML, an XML-based standard which supports federation, SSO, and attribute management;
- Shibboleth, a system with similarities to SAML and Kantara, was also designed to enable federation and SSO.



We now look in greater detail at three SSO systems of particular practical importance:

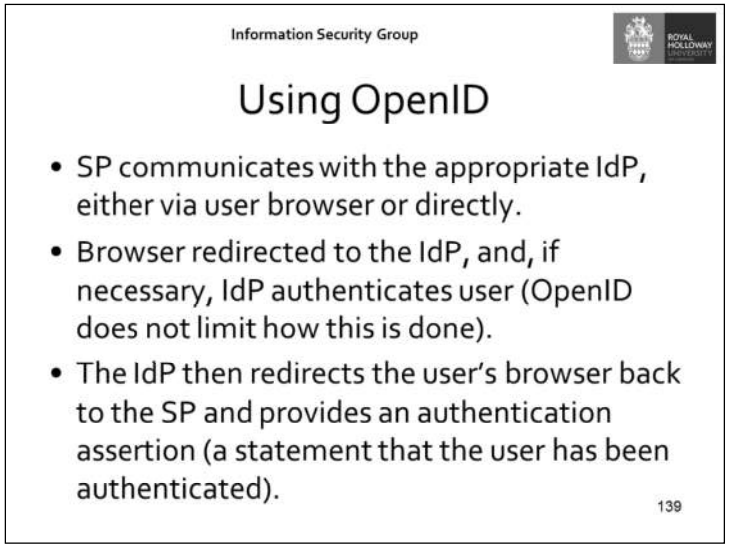
- OpenID;
- OAuth 2.0; and
- OpenID Connect – the successor to OpenID, which builds on OAuth 2.0.

We start by looking at OpenID.



OpenID is a decentralised SSO system (with some similarities to Liberty). OpenID is an open source project. Users register with an OpenID identity provider (IdP), and obtain an identifier specific to that IdP.

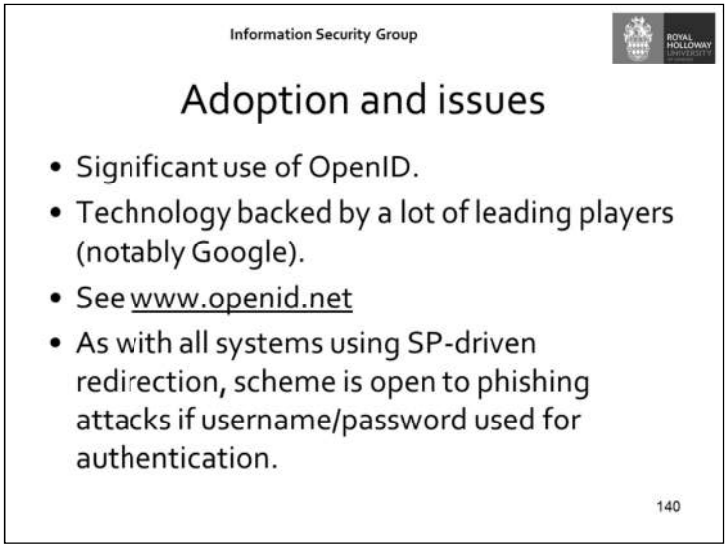
A service provider (the Replying Party, or RP) using OpenID displays a login form containing a space for an OpenID identifier (known as an **OpenID**), indicating a particular identity with a particular IdP (no password). An example of an OpenID might be something like `chris.openid.domain.org`



The SP then communicates with the appropriate IdP, either via the user's browser or directly.

The user's browser is redirected to the IdP, and, if necessary, the IdP then authenticates the user (OpenID does not restrict how this is done).

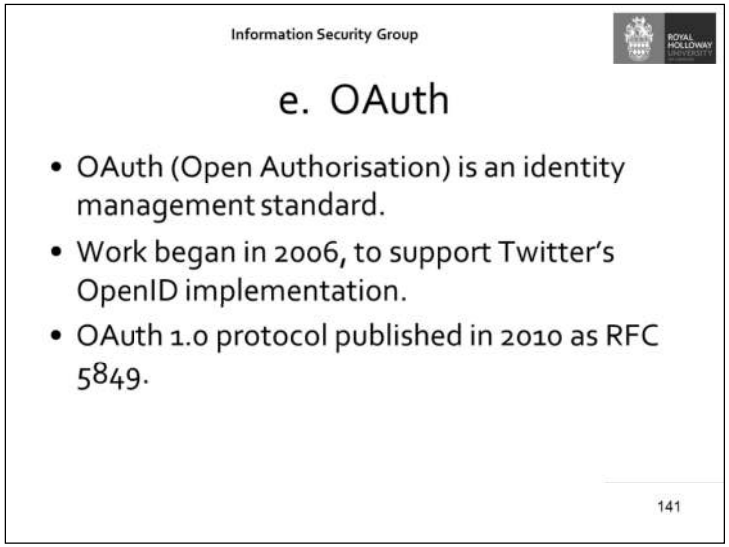
The IdP then redirects the user's browser back to the SP and provides an authentication assertion, a statement to the SP that the user has been authenticated.



There is significant use of OpenID. The technology is backed by a lot of leading players (notably Google, although Google has now moved on to using its successor technology, OpenID Connect).

For further details see www.openid.net

As with all systems relying on redirection at the behest of the RP, the scheme is open to phishing attacks if username/password used for authentication.

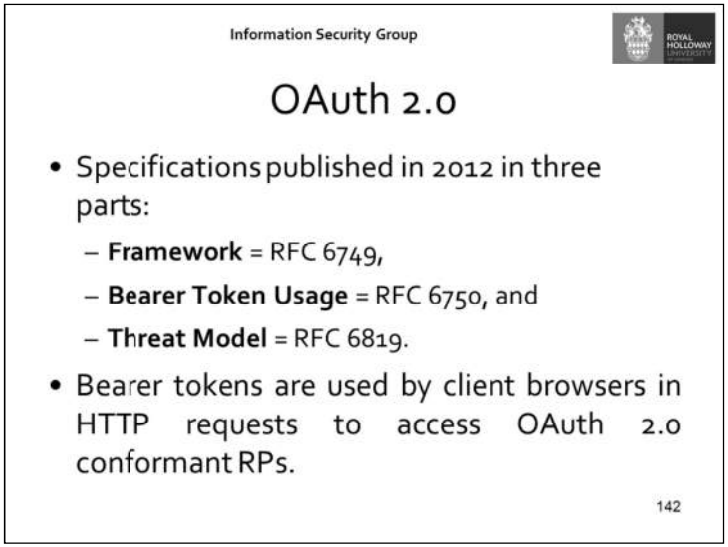


OAuth (Open Authorisation) is an identity management standard.

Work on OAuth began in November 2006, and was designed to support the Twitter OpenID implementation.

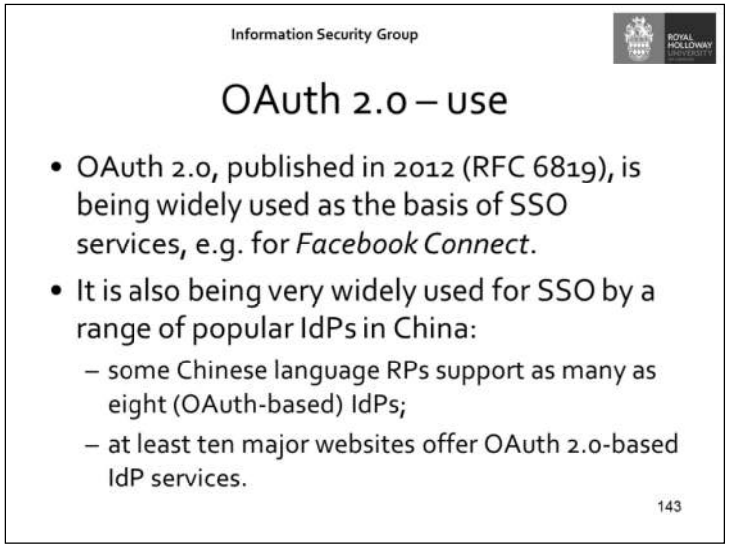
The OAuth discussion group was created in April 2007 to write the draft proposal for an open protocol.

On October 3, 2007, the OAuth Core 1.0 final draft was released. By 2008 there was sufficient support for formally chartering an OAuth working group within the IETF. The OAuth 1.0 Protocol was published as RFC 5849, in 2010.



In 2012, the OAuth 2.0 specifications were published in three parts:

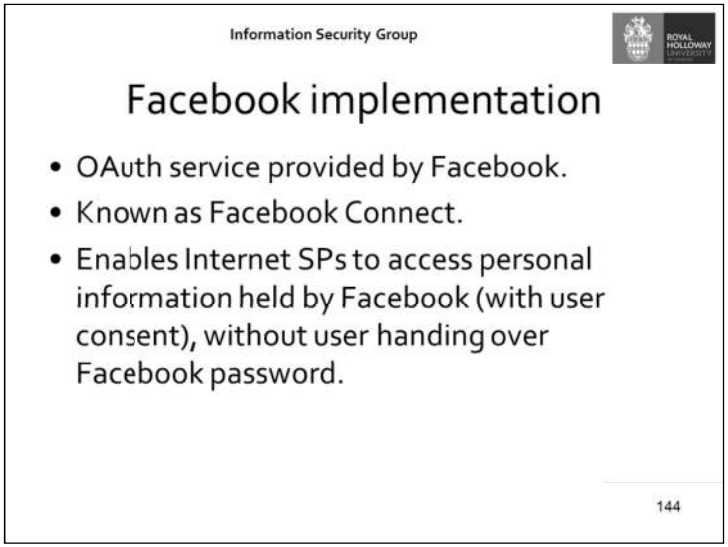
- the Framework was published as RFC 6749,
- the Bearer Token Usage as RFC 6750, and
- the Threat Model and Security Considerations as RFC 6819.



OAuth 2.0, published in 2012 (RFC 6819), is being widely used as the basis of SSO services, e.g. for *Facebook Connect*.

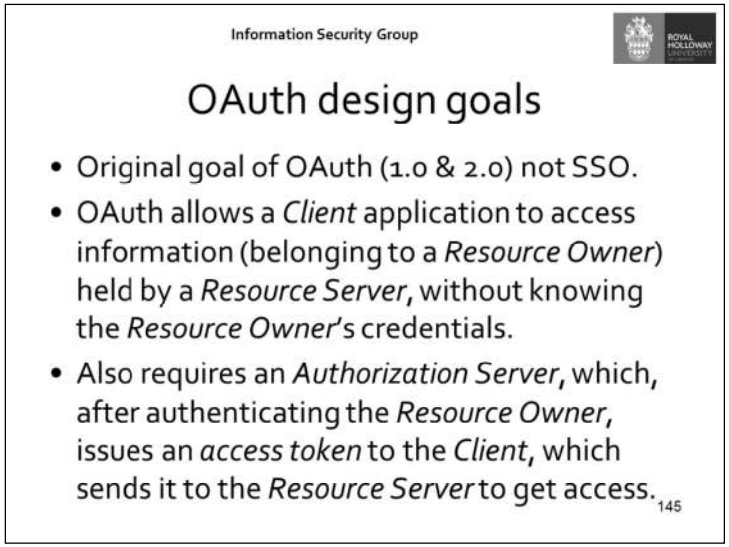
It is also being very widely used for SSO by a range of popular IdPs in China:

- some Chinese language RPs support as many as eight (OAuth-based) IdPs;
- at least ten major websites offer OAuth 2.0-based IdP services.



An OAuth service is provided by Facebook. This service is known as Facebook Connect.

It enables Internet SPs to access personal information held by Facebook (with user consent), without the user handing over his or her Facebook password.

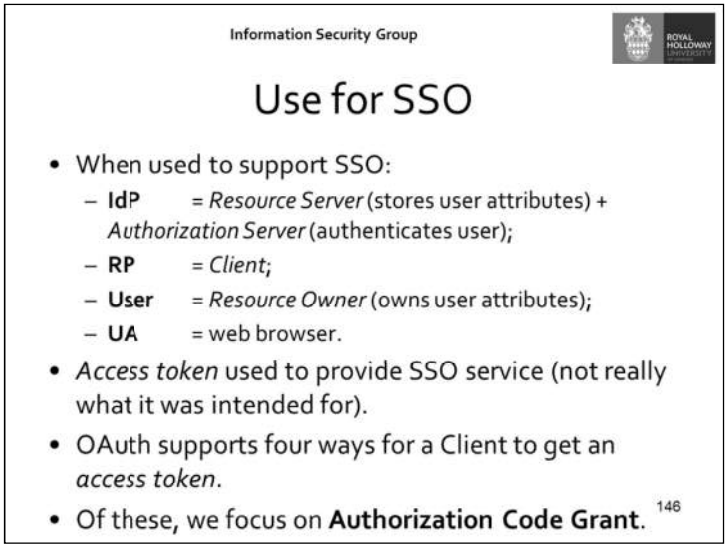


The original goal of OAuth (1.0 & 2.0) was not SSO. Instead it is designed to enable an end-user to grant an Internet application controlled access to personal information (e.g. user attributes, photos, contact lists, etc.) stored at a third party site, without divulging long-term credentials such as passwords.

In the absence of a system like OAuth, applications must request user credentials in order to access user information held by a third party, which is clearly undesirable.

More specifically, OAuth allows a *Client* application to access information (belonging to a *Resource Owner*) held by a *Resource Server*, without knowing the *Resource Owner*'s credentials.

It also requires an *Authorization Server*, which, after authenticating the *Resource Owner*, issues an *access token* to the *Client*, which sends it to the *Resource Server* to get access.

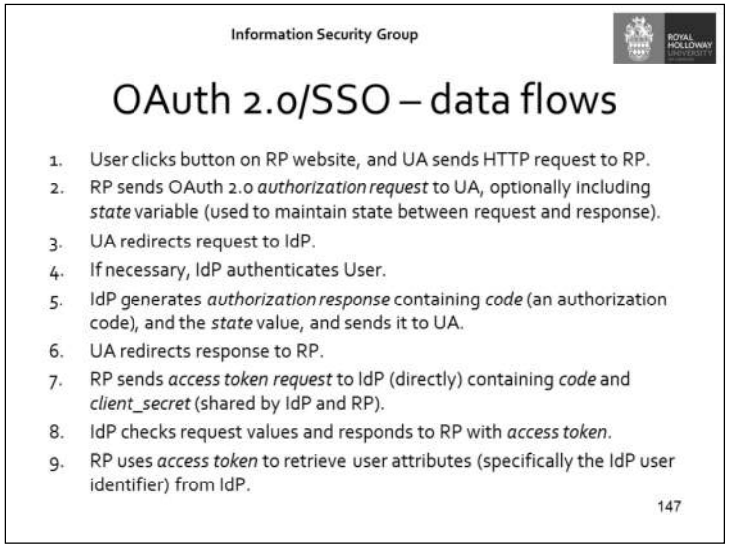


When used to support SSO:

- **IdP** = *Resource Server* (stores user attributes) + *Authorization Server* (authenticates user);
- **RP** = *Client*;
- **User** = *Resource Owner* (owns user attributes);
- **UA** = web browser.

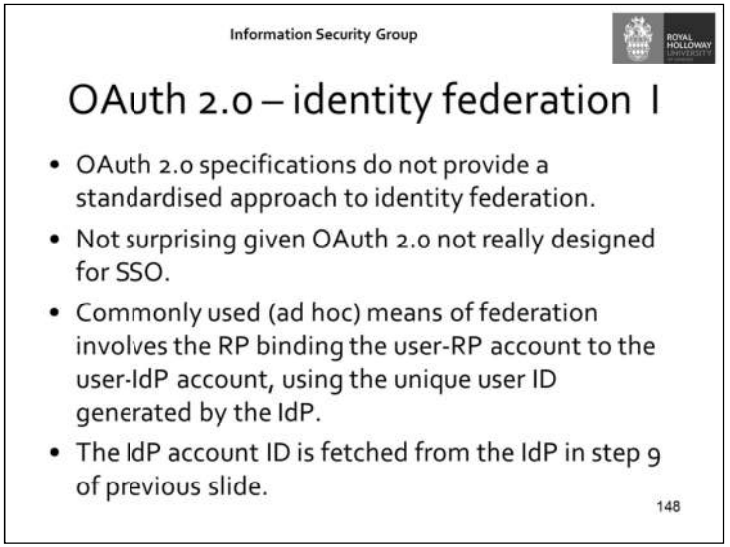
The *Access token* is used to provide the SSO service (it is not really what it was intended for).

OAuth supports four ways for a Client to get an *access token*. Of these, we focus on *Authorization Code Grant*.



When using the Authorization Code Flow, the following exchanges take place.

1. The User clicks a button on the RP website, and the UA sends an HTTP request to the RP.
2. RP sends OAuth 2.0 *authorization request* to UA, optionally including *state* variable (used to maintain state between request and response).
3. UA redirects request to IdP.
4. If necessary, IdP authenticates User.
5. IdP generates *authorization response* containing *code* (an authorization code), and the *state* value, and sends it to UA.
6. UA redirects response to RP.
7. RP sends *access token request* to IdP (directly) containing *code* and *client_secret* (shared by IdP and RP).
8. IdP checks request values and responds to RP with *access token*.
9. RP uses *access token* to retrieve user attributes (specifically the IdP user identifier) from IdP.

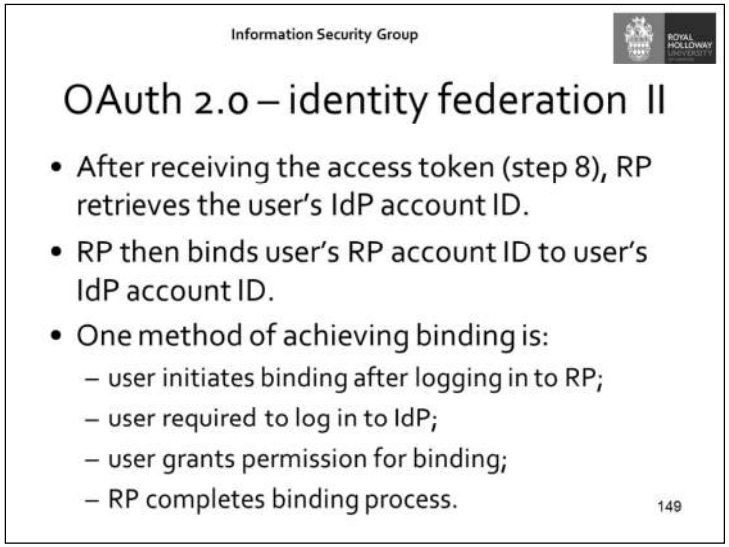


The OAuth 2.0 specifications do not provide a standardised approach to identity federation.

This is hardly surprising given that OAuth 2.0 was not really designed for SSO.

Commonly used (ad hoc) means of federation involves the RP binding the user-RP account to the user-IdP account, using the unique user ID generated by the IdP.

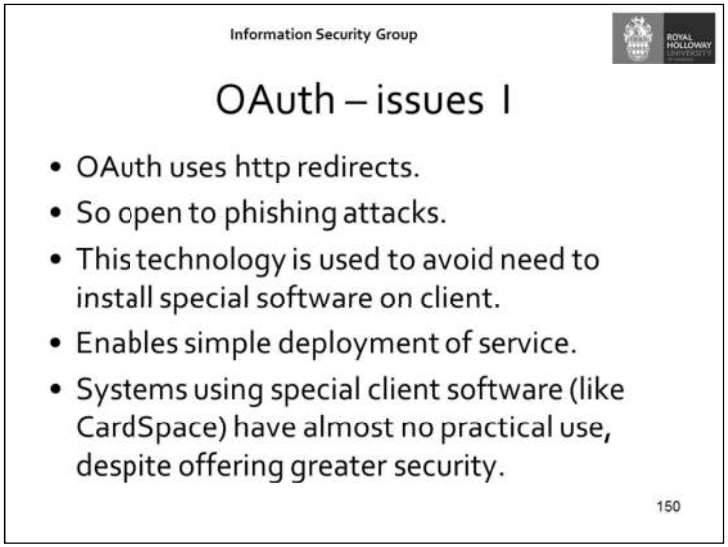
The IdP account ID is fetched from the IdP in step 9 of previous slide.



After receiving the access token (step 8), the RP retrieves the user-IdP account ID. The RP can then bind the user's RP account ID to the user's IdP account ID.

One method of achieving binding is as follows:

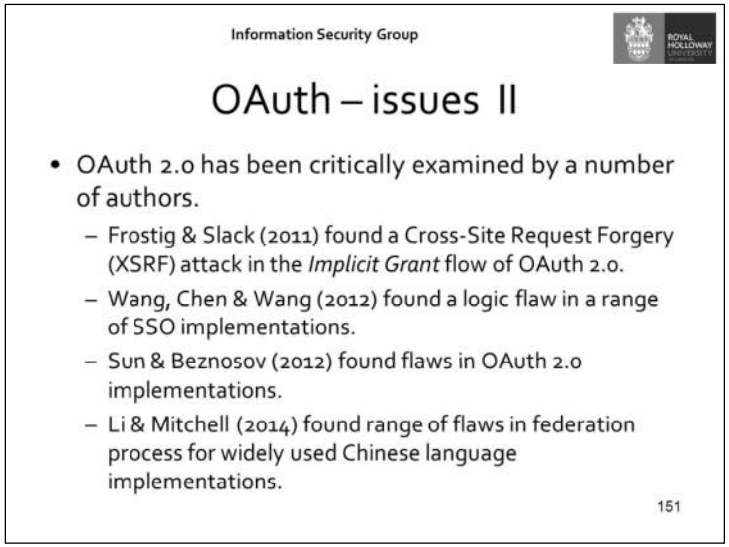
- the user initiates binding after logging in to RP;
- the user is required to log in to IdP;
- the user grants permission for binding;
- the RP completes binding process.



Like OpenID and some versions of Liberty, OAuth uses http redirects to manage interactions between the SP and the IdP. As a result it is open to phishing attacks.

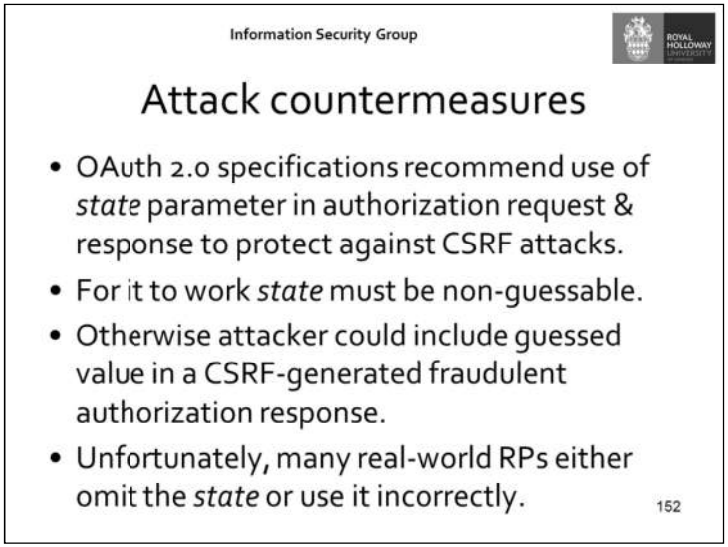
This redirect technology is used to avoid the need to install special software on the client platform. As a result it enables simple deployment of the service.

Indeed, systems using special client software (like CardSpace) have almost no practical use, despite offering greater security (and, in particular, resistance to phishing).



OAuth 2.0 has been critically examined by a number of authors.

- Frostig & Slack (2011) found a Cross-Site Request Forgery (XSRF) attack in the *Implicit Grant* flow of OAuth 2.0.
- Wang, Chen & Wang (2012) found a logic flaw in a range of SSO implementations.
- Sun & Beznosov (2012) found flaws in OAuth 2.0 implementations.
- Li & Mitchell (2014) found a range of flaws in federation process for widely used Chinese language implementations.

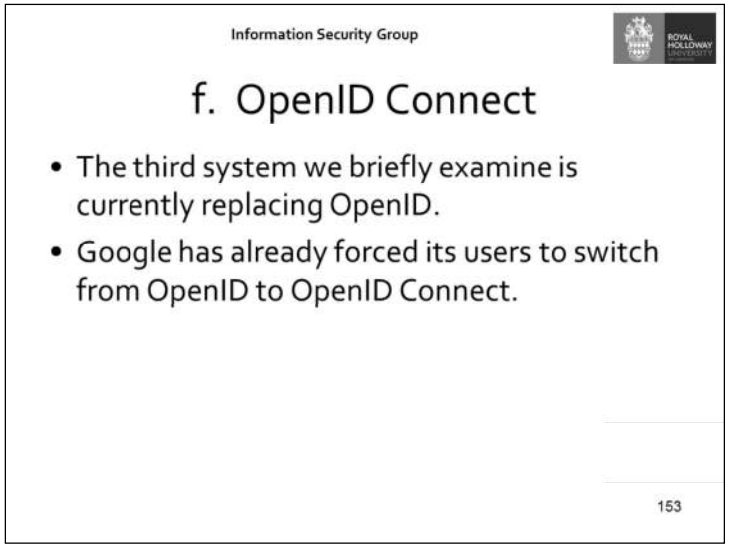


OAuth 2.0 specifications recommend use of *state* parameter in authorization request & response to protect against CSRF attacks.

For it to work *state* must be non-guessable.

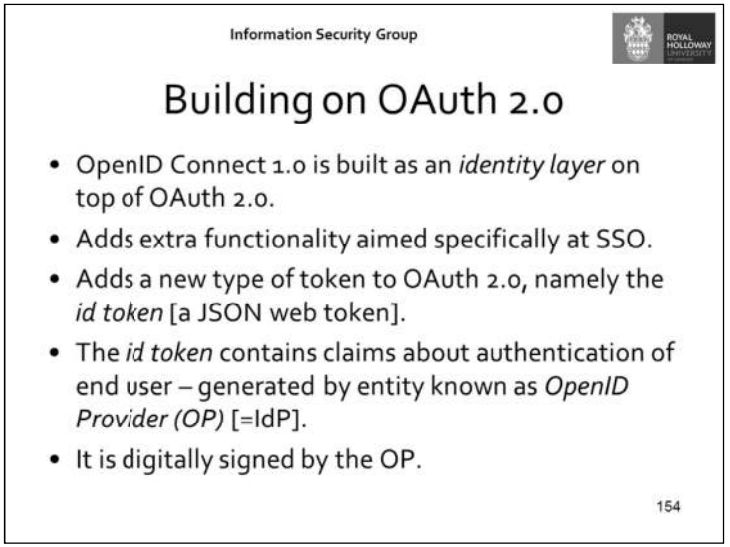
Otherwise attacker could include guessed value in a CSRF-generated fraudulent authorization response.

Unfortunately, in practice it seems that many real-world RPs either omit the *state* or use it incorrectly.



The third system we briefly examine is currently replacing OpenID.

Google has already forced its users to switch from OpenID to OpenID Connect.



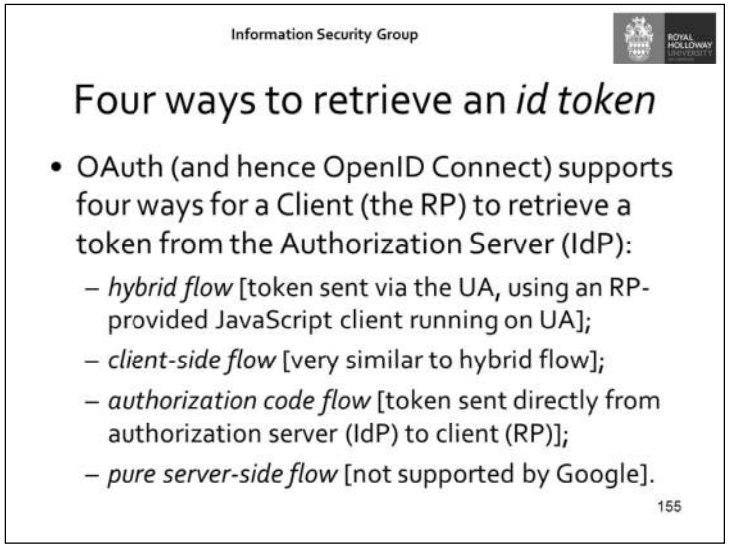
OpenID Connect 1.0 is built as an *identity layer* on top of OAuth 2.0.

Adds extra functionality aimed specifically at SSO.

Adds a new type of token to OAuth 2.0, namely the *id token* [a JSON web token].

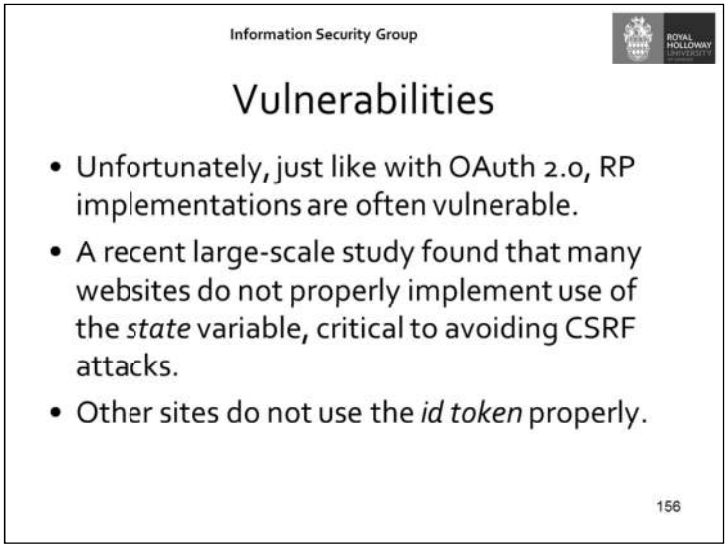
The *id token* contains claims about authentication of end user – generated by entity known as *OpenID Provider (OP)* [=IdP].

It is digitally signed by the OP.



OAuth 2.0 (and hence OpenID Connect) supports four ways for a Client (the RP) to retrieve a token from the Authorization Server (IdP):

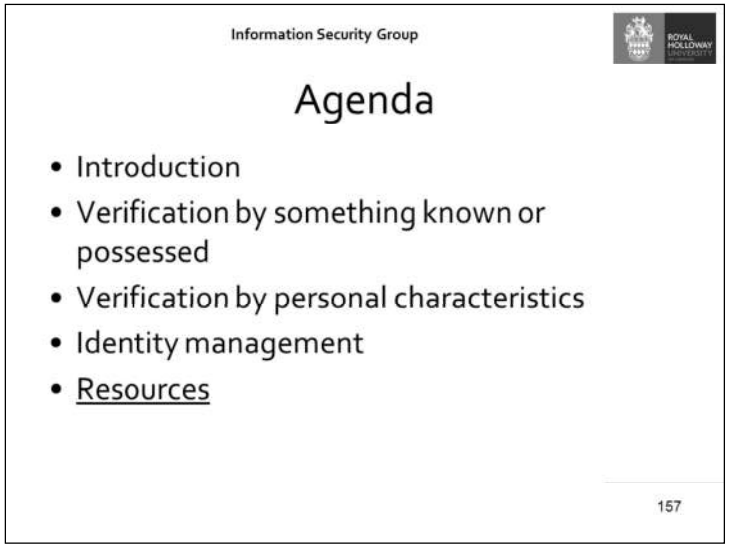
- *hybrid flow* [token sent via the UA, using an RP-provided JavaScript client running on UA];
- *client-side flow* [very similar to hybrid flow];
- *authorization code flow* [token sent directly from authorization server (IdP) to client (RP)];
- *pure server-side flow* [not supported by Google].



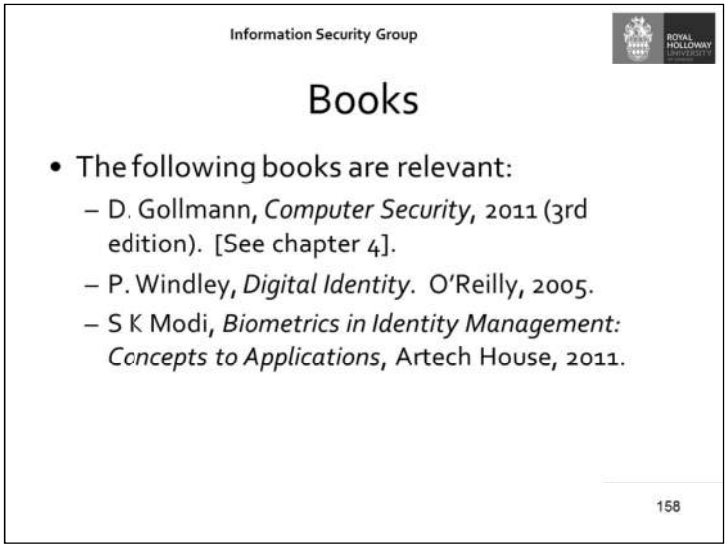
Unfortunately, just like with OAuth 2.0, RP implementations are often vulnerable.

A recent large-scale study found that many websites do not properly implement use of the *state* variable, critical to avoiding CSRF attacks.

Other sites do not use the *id token* properly.



We conclude by providing some pointers to relevant literature.



Of some relevance are the following books:

- chapter 4 of D. Gollmann, *Computer Security*, 2011 (3rd edition).
- P. Windley, *Digital Identity*. O'Reilly, 2005.
- S K Modi, *Biometrics in Identity Management: Concepts to Applications*, Artech House, 2011.

The following resources are relevant to this part of the course:

• `http://www.identityblog.com/` [Kim Cameron's identity weblog is here, and much else besides].

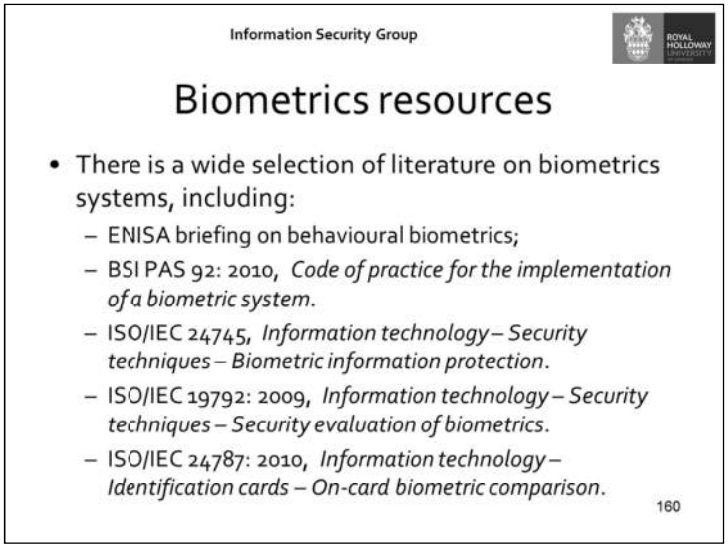
• http://www.kantarainitiative.org (superseding http://www.projectliberty.org).

• http://www.ietf.org [This is the IETF website from which RFCs and Internet drafts can be obtained].

• http://www.microsoft.com [Information about CardSpace is available here].

• `http://www.eclipse.org/higgins/` [Information about Higgins, the open source rival to CardSpace, is available here].

• http://oauth.net/ [This is the official OAuth site].



There is a wide selection of literature on biometrics systems Key documents include:

• ENISA briefing on behavioural biometrics: http://www.enisa.europa.eu/activities/risk-management/files/deliverables/behavioural-biometrics/at_download/fullReport

• BSI PAS 92: 2010, *Code of practice for the implementation of a biometric system*.

• ISO/IEC 24745: 2011, *Information technology – Security techniques – Biometric information protection*.

• ISO/IEC 19792: 2009, *Information technology – Security techniques – Security evaluation of biometrics*.

• ISO/IEC 24787: 2010, *Information technology – Identification cards – On-card biometric comparison*.

Two recent papers describing a range of real-world vulnerabilities are as follows:

- W. Li and C. J. Mitchell, 'Security issues in OAuth 2.0 SSO implementations', *in: Proceedings of the 17th Information Security Conference, Hong Kong, China, 12-14 October 2014 (ISC 2014)*, Springer-Verlag **LNCS 8783** (2014), pp. 529-541.
- W. Li and C. J. Mitchell, 'Analysing the security of Google's implementation of OpenID Connect', arXiv:1508.01707 [cs.CR], August 2015, 27 pages.