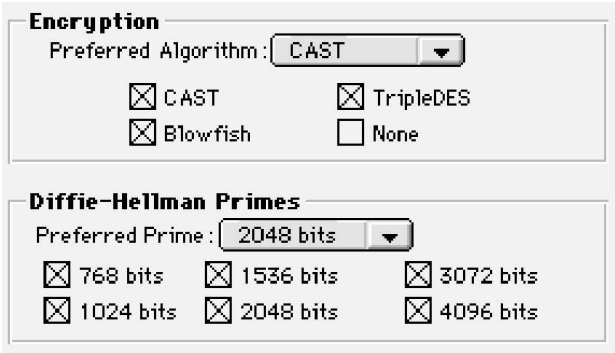
All cryptographic and speech compression protocols are negotiated dynamically and invisibly, providing a natural user interface similar to using a normal telephone. Public-key protocols are used to negotiate keys. ***Enough advertising!***

One of the peculiarities about PGPfone, is that it is available in two versions:

1. An international version available *outside* America, and a prohibited import *into* America.
2. An American version available *inside* America, and a prohibited import *out of* America.

These two versions are also exactly the same! This peculiar situation is a result of American restrictions on the import and export of munitions - strong cryptography is considered a munition.

When we look at the preferences dialog, we see familiar encryption and key exchange parameters:



When initially setting up a link, Diffie-Hellman key exchange is used to ensure safety in the choice of an encryption key.

12.4 Design principles

There are various principles related to the design of secure systems, outlined in a paper by Saltzer and Schroeder, and summarized below:

1. **Economy of mechanism:** Keep the design as simple and small as possible. (identd assumption)
2. **Fail-safe defaults:** Base access decisions on permission rather than exclusion. This is conservative design - the arguments are based on arguments as to why objects should be accessible, rather than why they should not. (mail server - mail only access)
3. **Complete mediation:** Every access to every object must be checked for authority. (DNS cache poisoning)

4. **Open design:** The design should not be secret. It is not realistic to attempt to maintain secrecy for any system which receives wide distribution. (DVDs, Microsoft SAM hashes...)

5. **Separation of privilege:** Two keys are better than one. Keys can be physically separated and distinct programs, organizations, or individuals made responsible for them. No single event can compromise the system. (su - password and *wheel* group)

6. **Least privilege:** Every program and every user of the system should operate using the least set of privileges necessary to complete the job. The military security rule of "need-to-know" is an example of this principle.

7. **Least common mechanism:** Minimize the amount of mechanism common to more than one user and depended on by all users. For example, given the choice of implementing a new function as a supervisor procedure shared by all users or as a library procedure that can be handled as though it were the user's own, choose the latter course.

8. **Psychological acceptability:** Human interface easy to use.

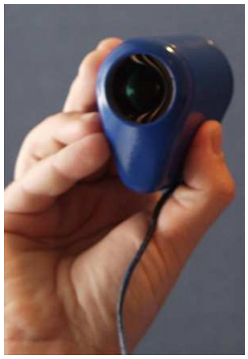
In the textbook there are examples of the use of each of these design principles.

12.5 Biometrics

Biometrics is the use of human physical characteristics to support authentication. Examples of this include:

- Fingerprint scanners are readily available, along with algorithms to perform reasonably efficient comparisons with small databases of fingerprints:

- Eyes - Iris and Retinal scanners are also readily available, again with algorithms to perform reasonably efficient comparisons with small databases of iris/retina patterns:



12.5.1 Minimal hardware biometrics

Some biometric identifiers can be captured with commonly available hardware. In particular, these days it is easy and inexpensive to capture sound and video images, giving the following biometric identifiers:

- Voices - Record and process voice leading to either speaker verification or recognition. Recognition may involve analysis of components of the voice that should not be duplicatable by anyone else.
- Faces - Capture either a static or moving image of a face. I had difficulty once with facial recognition :)
- Keystrokes - capture a sequence of keystrokes, recording timing.

Combinations of characteristics may be used, but in general biometric techniques are not reliable on their own. However, they do make a good second key for *separation of privilege*.

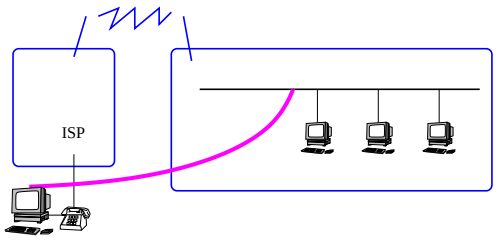
12.6 IPSec

The Internet is a notoriously unregulated network, with no guarantees of the safety or security of any traffic that traverses it. To address this concern, IPSec has been (and still is being) developed. IPSec is a set of standards intended to support communication security between networked computers, particularly in the newer IPv6 (IP Next-Generation) network. IPSec software is available in Windows2000, Linux, and on routers on the Internet.

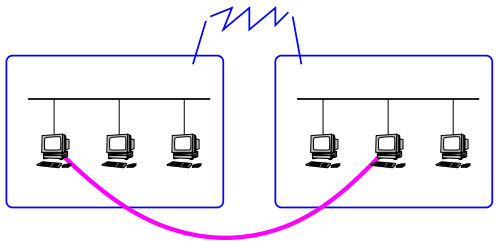
http://www.faqs.org/rfcs/rfc2401.html

IPSec may be used in a range of ways. For example:

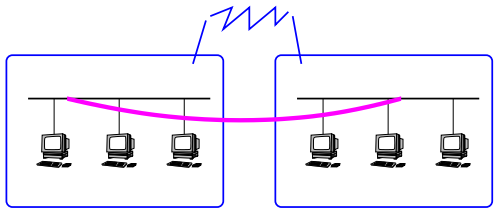
- to allow remote users to access safely a network (as in the NUS VPN)



- to allow two users to transfer data safely



- To interconnect two networks



In each case, IPSec provides mechanisms based on an open security architecture, with extendible encryption and authentication schemes. This means that it is possible to use any desired type of cryptography and key exchange schemes, or just to use standard ones.

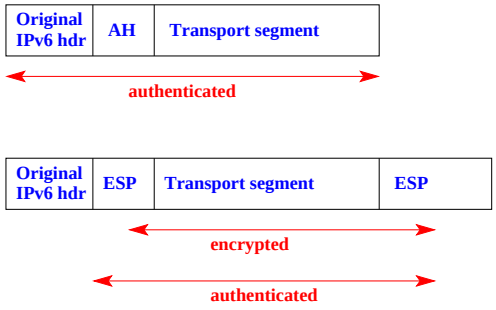
There are two types of header, one used for authentication, and the other used for encryption:

1. AH - the Authentication Header for data integrity, anti-replay and authentication
2. ESP - the Encapsulating Security Payload header, for confidentiality. ESP can also provide AH services.

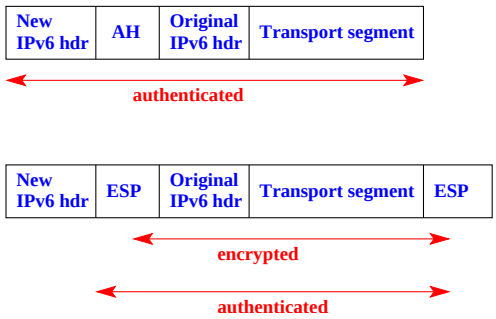
Communicating parties agree on a Security Association (SA), one SA for each direction, and one SA for each type of communication.

There are two modes of operation:

- An end-to-end SA - Transport mode



- An SA between security gateways - Tunnel mode



SAs are administered at each interface involved in the communication. The SAs form a kind of distributed database.

12.7 Formal methods

In general, formal methods encompasses a wide range of techniques. For example the model checking approach to verification of software may involve

- constructing formal *models*, with
- appropriate formal *specifications.*

An example of this process is found in **Promela** and **Spin**. The language Promela is 'C' like, with an initialization procedure. The *formal* basis for Promela is that it is a guarded command language, with extra statements which either make general assertions or test for reachability.

It can be used to model asynchronous or synchronous, deterministic or non-deterministic systems, and has a special data type called a *channel* which assists in modelling communication between processes.

Spin is the checker for Promela models, and can be used in three basic modes:

1. Simulator - for rapid prototyping with a random, guided, or interactive simulation.
2. State space analyzer - exhaustively proving the validity of user specified correctness requirements (using partial order reduction theory to optimize the search).
3. Bit-state space analyzer - validates large protocol systems with maximal coverage of the state space (a proof approximation technique).

We can pepper our code with assertions to test the correctness of our model:

```
assert(some_boolean_condition);
```

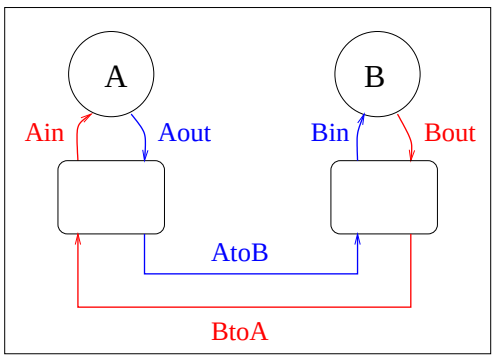
If the asserted condition is not TRUE then the simulation or verification fails, indicating the assertion that was violated.

We may also make temporal claims - for example a claim such as "*we got here again without making any progress*". The support for temporal claims takes the form of:

- Endstate labels - for determining valid endstates
- Progress labels - claim the absence of non-progress cycles
- Never claims - express impossible temporal assertions

12.7.1 Simple example

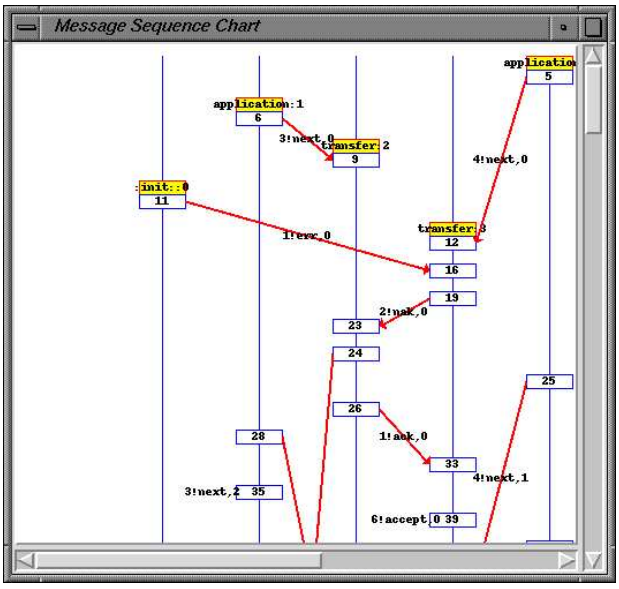
In this example, we model a simple system with two application processes (A and B), communicating with each other using a protocol implemented with two other *transfer* processes. We may visualize this as follows:



The following code segment specifies the simple protocol, and a simple application to 'exercise' the protocol.

```
#define MAX 10
mtype = { ack, nak, err, next, accept }
proctype transfer( chan in, out, chin, chout )
{
    byte o,i;
    in?next(o);
    do
      :: chin?nak(i) -> out!accept(i); chout!ack(o)
      :: chin?ack(i) -> out!accept(i); in?next(o); chout!ack(o)
      :: chin?err(i) -> chout!nak(o)
    od
}
proctype application( chan in, out )
{
    int i=0, j=0, last_i=0;
    do
      :: in?accept(i) ->
            assert( i==last_i );
            if
              :: (last_i!=MAX) -> last_i = last_i+1
              :: (last_i==MAX)
            fi
      :: out!next(j) ->
            if
              :: (j!=MAX) -> j=j+1
              :: (j==MAX)
            fi
    od
}
init
{
    chan AtoB = [1] of { mtype,byte };
    chan BtoA = [1] of { mtype,byte };
    chan Ain  = [2] of { mtype,byte };
    chan Bin  = [2] of { mtype,byte };
    chan Aout = [2] of { mtype,byte };
    chan Bout = [2] of { mtype,byte };
    atomic {
       run application( Ain,Aout );
       run transfer( Aout,Ain,BtoA,AtoB );
       run transfer( Bout,Bin,AtoB,BtoA );
       run application( Bin,Bout )
    };
    AtoB!err(0)
}
```

The spin tool may then be used to either exhaustively check the model, or to simulate and display the results:



12.8 Formal evaluation

TCSEC (The Orange book) was the first rating system for the security of products. It defined six different evaluation classes. The classes are:

- **C1** - For same-level security access. Not currently used.
- **C2** - Controlled access protection - users are individually accountable for their actions. Most OS manufacturers have C2 versions of the OS.
- **B1** - Mandatory BLP policies - for more secure systems handling classified data.
- **B2** - structured protection - mandatory access control for all objects in the system. Formal models.
- **B3** - security domains - more controls, minimal complexity, provable consistency of model.
- **A1** - Verified design - consistency proofs between model and specification.

A more international (and non-military) effort ITSEC has been developed from an amalgamation of Dutch, English, French and German national security evaluation criteria. The particular advantage of ITSEC is that it is adaptable to changing technology and new sets of security requirements.

ITSEC evaluation begins with the sponsor of the evaluation determining an assessment of the operational requirements and threats, and the security objectives. ITSEC then specifies the interactions and documents between the sponsor and the evaluator. Again there are various levels of evaluation: E0..E6, with E6 giving the highest level of assurance - it requires two independant formal verifications.

In [Woo98], elements of the first certification of a smart-card system under the European ITSEC level 6 certification are outlined. The smart-cards are electronic purses - that is they carry value, and the bank requirement was that forgery must be impossible. The certification encompassed the communication with the card, as well as the software within the card, and at the bank.

12.9 Summary of topics

In this section, we introduced the following topics:

- Systems for security
  - ssh
  - SSL
  - pgpfone
  - IPSec
- Design principles for secure systems
- Biometric identification
- Formal methods
- Formal evaluation

Further study

- Design Principles, textbook section 13.2, and the original paper at http://web.mit.edu/Saltzer/www/publications/protection/index.html
- Biometrics, textbook section 12.4
- IPSec, textbook section 11.4.3
- Formal methods, textbook section 20.1, 20.2
- Formal evaluation, textbook section 21.1, 21.2, 21.3
