Q1. Explain secret and public key cryptography schemes. Use small examples to illustrate your claims. State relative advantages of each scheme.

Secret Key Cryptography

Overview
Alice wants to send a message to Bob. Both Alice and Bob share the same secret key. To encrypt the message Alice XORs her message with the shared secret key. To decrypt the message Bob also XORs the message with his (the same) secret key.

Ex.

Key = 0011
Alice's message = 0101
Alice's message XORed with the key: 0011 XOR 0101 = 0110
What Bob Recieves = 0110
Bob applies the secret key again to get the original message: 0110 XOR key = 0101

This works for three reasons:
- XOR is associative and
- Any binary value XORed with itself is 0
- Any binary value XORed with 0 is itself

The advantages of secret key cryptography are that
1. Performing XOR is very fast.
2. It has been well tested.

The disadvantages are that
1. The key must remain secret.
2. Exchanging keys with someone must be done in secret.
3. Each communicating pair of people need to share a key.

Public Key Cryptography

Overview
In public key cryptography there are two parts to the key: a secret part and a public part. In order for Alice to send Bob a message she first needs to obtain his public key. Because Bob likes to be contacted (albeit only via encrypted messages) he has published his public key on his homepage for anyone to download. Alice obtains his public key, encrypts a message using this key and then sends it to Bob. Bob is then able to decrypt the message using the secret part of his own key.

Advantages
1. Only one part must be kept secret
2. There is no need to change your public/private key pair (unless someone finds your public key)
3. For N people to communicate there need only be N public/private key pairs.
4. There is no need for initial key exchange
5. It can serve as a digital signature
    a. Alice would like to post a message on a web forum and she would also like everyone reading the forum to be able to authenticate that the message was really posted by her (and not someone else claiming to be her). Alice takes her message and uses her private key to encrypt it. If the encrypted message on the forum was really posted by Alice then it should be decryptable using her public key. Since everyone has access to her public key this can be verified.

Disadvantages

1. Slow do to the enormous amount of computation involved.
2. Keys must be long (at least 1024 bits these days).
3. There is no proof for that any public key scheme is secure.
4. It has not been around long enough to be tested as much.

Diffie-Hellman Public Key Cryptography

Y = Public Key
X = Secret Key

User i derives Y in the following way.
$Y_i = a\hat{}(x_i) \bmod g$

User j derives Y in the same manner.
$Y_j = a\hat{}(x_j) \bmod g$

Users i and j exchange their Y's and then perform the following computation

$Z_{ij} = Y_j\hat{}(x_i) = (a\hat{}(x_j))\hat{}(x_i) \bmod g = (a\hat{}(x_i))\hat{}(x_j) \bmod g = Z_{ji}$

This value of Z can then be used as the input to a random number generator. These random bits will be used to XOR the original message.

Q2. Explain the Diffie-Helman key distribution scheme. Show a small example. Explain (very briefly) other public key schemes and their relative advantages and limitations.

Explain the Diffie-Helman key distribution scheme:

- Public key cryptography scheme
- Uses a one-way function.
- Security due to the difficulty of calculating logarithms as compared to the ease of calculating exponentiations.
- How
  - Alice and Bob want to generate a secret key
  - Public Givens:
    Large prime number n
    g = primitive mod of n

  - Alice choses a random large number integer x and sends to Bob
    $X=g^x \bmod n$
  - Bob choses a random large number y and sends to Alice
    $Y = g^y \bmod n$
  - Alice computes
    $k=Y^x \bmod n$
  - Bob computes
    $k'=X^y \bmod n$

  - The secret key
    $k=k'=g^{xy} \bmod n$
  - No one listening can compute the value k since they only know n, g, X, Y, and it is too difficult to compute the log to obtain x and y.

Show a small example:

- g= 6 = 35 mod n
  n = 29
- Alice choses x = 101
  X = 6^{101}
- Bob choses y = 53
  Y = 6^{53}
- Alice computes
  k = 6^{53*101}
- Bob computes
  k' = 6^{101*53}

Explain other public key schemes and their relative advantages and limitations:

- RSA:

Difficulty of recovering plain text from cipher test is conjectured to be equivalent to factoring products of large prime numbers
Advantages:
Been around a long time, and heavily studied
Disadvantages:
Not proven only conjectured security

- McEliece:

Idea:
Based on Algebraic coding theory
Construct an error-correcting code, specifically a Goppa code
Disguise the Goppa code as a general linear code
Finding a code word of a given weight in a linear binary code is NP complete but fast algorithm exists for decoding Goppa code
Advantages:
Fast
Disdvantages:
Key is enormous
Data expansion is large

- Eliptic Curves:

Eliptical curves are utilized to implement algorithms such as Diffie-Hellman
Advantages:
Most efficient with regard to key

- Quadratic residuity

Quadratic Residue is based on number theory idea of the same name

- Largest Clique

Acknowledgement: Applied Crytpography by Bruce Schneier was heavily referenced

---

Q3. Explain the Shamir's secret sharing scheme. Explain its application. Show a small example. Explain potential alternatives.

Objective: Secret Sharing scheme that is both perfect and fault tolerant perfect - stored key does not reveal any information fault tolerant - if one part of key is lost - the key must still be recoverable.

Using polynomial interpolation - and the idea that for any polynomial of degree k, for any k-points, there is only one line that passes through all of these points.

The shares of the key are represented by these points on the k-degree polynomial. By interpolating the line at x=0, the secret is obtained.

For example, for a 2-degree polynomial,
if the shares are represented by the points
<x1,y1>,<x2,y2>...<xn,yn> - all of these points lie on the same line. Pick any two points - draw a line through them, and the intersection at x=0 is the secret.

For a k-degree polynomial, at least k points must be picked to determine the intersection at x=0.

The scheme is fault-tolerant because not all shares are needed to determine the secret.

In addition the scheme is perfect because the secret can be encrypted. Even complete knowledge of k-1 shares (for k-degree poly) will not reveal the secret.

Shortcomings:
The scheme does not provide:
key revocation -
  if A gives B a share, then decides it wants to take it back,
  it can not do this
traitor resilience:
  if A lies to B about the share, then B cannot detect this.

References:
- Lecture
- http://szabo.best.vwh.net/secret.html

---

4. What is perfect secret sharing scheme? Show at least one perfect and three non-perfect schemes.

In a secret sharing scheme, the sender divides the secret into n parts and gives each participant one part so that any m parts can be put together to recover the secret, but any m - 1 parts reveal no information about the secret. A secret sharing scheme is perfect if any group of at most m - 1 participants (insiders) has no advantage in guessing the secret over the outsiders.
Shamir's secret sharing scheme is an example of perfect secret sharing. It is an interpolating scheme based on polynomial interpolation. An (m - 1)-degree polynomial: F(x) = a0 + a1x + ... + am - 1 xm-1, is constructed such that the coefficient a0 is the secret and all other coefficients are random elements in the field. Each of the n shares is a point (xi, yi) on the curve defined by the polynomial, where xi not equal to 0. Given any m shares, the polynomial is uniquely determined and hence the secret a0 can be computed. However, given m - 1 or fewer shares, the secret can be any element in the field. Therefore, Shamir's scheme is a perfect secret sharing scheme.

Blakley's secret sharing schemes is an example of non-perfect secret sharing. The secret is a point in an m-dimensional space. n shares are constructed with each share defining a hyperplane in this space. By finding the intersection of any m of these planes, the secret (or point of intersection) can be obtained. This scheme is not perfect, as the person with a share of the secret knows that the secret is a point on his hyperplane.

Other non-perfect secret sharing schemes are, e.g. secret sharing schemes based on Latin Squares, or Room Squares.

---

5. Explain zero knowledge proof. Demonstrate how zero can be applied to graph coloring and Hamiltonian cycle problems.

A Zero Knowledge Proof is a proof in which a Prover (Peggy) wants to prove to a Verifier (Victor) that she knows a particular secret. In the process of the proof, however, Peggy doesn't want to reveal her secret to Victor. In fact, Peggy does not want to reveal even a tiny bit of information about her secret to Victor. In the end, Peggy wants Victor to have no more information about Peggy's secret than he could have learned on his own.

Zero Knowledge Proofs usually take the form of a probabilistic, interactive protocol. The protocol is said to be probabilistic because in the end, Victor will only be certain that Peggy knows the secret with high probability. Usually, the probability that Peggy is gets away with cheating shrinks like (1/2)^K where K is the number of rounds in the protocol.

Quisquater and Guillou illustrate the concept of Zero Knowledge Proofs with the following example:

Peggy and Victor know of a cave that forks into two tunnels. Each tunnel dead-ends at a door. Peggy claims that the door connects the two tunnels and that anyone who knows the password can pass through. Victor does not believe that Peggy knows the password and asks for proof. Peggy wants to convince Victor that she knows the password, but she does not want to reveal what it is. Peggy proposes the following:

1. Peggy and Victor start at the mouth of the cave. Peggy enters, leaving Victor behind. When she reaches the fork, she randomly chooses one tunnel.
2. Peggy calls to Victor to enter the cave. Victor comes to the fork. Peggy is now committed to the choice she made. She cannot change which tunnel she is in by walking back to the fork, or Victor will catch her.
3. Victor randomly selects one of the tunnels and calls to Peggy to emerge from that tunnel.
4. If Peggy truly knows the password and can pass by the door, then it is trivial for her to come out the correct tunnel. If she does not, then 50% of the time she will not be able to come out of the right path.
5. Peggy and Victor repeat this protocol many times until Victor is convinced that Peggy is not merely lucky, but that she can pass from one tunnel to the other through the door and therefore she must know the password. However, Victor knows nothing more about the password than when he started.

Now let's look at the mathematical underpinnings of such a scheme. Let's suppose that Peggy has a secret expressed in the form of the solution to a hard problem. Here's how Peggy can prove that she knows the solution to Victor without giving him any more information than he could come up with on his own.

1. Peggy selects a random number and uses it to transform her hard problem into a different hard problem that is equivalent to her initial problem.
2. Peggy then uses a cryptographic commitment scheme to commit to the solution of her new problem instance and sends that commitment to Victor. If she tries to switch the solution out later, she risks getting caught.
3. Peggy now sends the new problem instance to Victor.
4. Victor randomly chooses to ask Peggy to either:

   a) demonstrate that the new instance and the old instance are equivalent by providing a mapping between the two, or
   b) demonstrate that the solution she committed to in step 2 is really a solution to the new instance (and show that it matches the commitment).

5. Peggy complies as appropriate. Failure to do so will make Victor very suspicious and might be enough to convince him that Peggy is bluffing.
6. In order to reduce the odds of Peggy being able to get away with cheating, they repeat steps 1--4 until Victor is satisfied that Peggy really knows a solution to the original problem. However, no matter how many times the protocol is repeated, Victor learns nothing about Peggy's secret that he could not have learned on his own.

Below are schemes for Zero Knowledge Proofs using the Hamiltonian cycle problem and the three-coloring problem. For those who are a bit rusty on their Graph Theory, the Hamiltonian cycle problem is to find a cycle (if one exists) that visits every node in a graph exactly once and returns to the starting node. The three-coloring problem is to find an assignment (if one exists) of color from a set of three colors for each node such that no two adjacent nodes share the same color.

Zero Knowledge Proof Method for Hamiltonian Cycles

Suppose that Peggy has a graph G and that she knows a Hamiltonian cycle for it (probably because she generated G and in doing so, guaranteed that it had such a cycle). In general, finding a Hamiltonian Cycle in an arbitrary graph is NP-Complete. Peggy has engineered her graph so that it is no exception.

Let's now suppose that Peggy wants to prove to Victor that she knows a Hamiltonian cycle in G, but that she does not want to reveal the cycle to Victor. Peggy can prove to Victor that she knows the cycle using the following Zero Knowledge Proof:

1. Peggy selects a random number and uses it to permute the labeling of G, forming an isomorphic graph H. If Peggy knows a Hamiltonian cycle in G, she can easily find one in H by applying the labeling permutation.
2. Peggy then encrypts H to get H' and sends H' to Victor. (Peggy uses probabilistic encryption on each edge in H to ensure that Victor cannot attempt to discover H by encrypting various graphs and comparing them to H'.) This commits Peggy to her choice of H.
3. Victor randomly chooses to ask Peggy to either:

   a) demonstrate that H' is an encryption of H and that H is isomorphic to G, or
   b) reveal a Hamiltonian cycle in H by revealing the decryption of the necessary edges in H'.

4. Peggy complies by either:

   a) providing the decryption on H and the permutation of G so that Victor can see that H is isomorphic to G, but without demonstrating a Hamiltonian cycle in G or H, or
   b) providing the decryption for edges in H' which form a Hamiltonian cycle, but without demonstrating that G and H are isomorphic.

5. Peggy and Victor repeat steps 1--4 K times. If Peggy ever declines to complete her part of the protocol, or if she ever fails to send valid information in step 4, Victor has reason to suspect
Peggy is lying. Otherwise, he is certain with probability 1-(1/2)^K that Peggy knows a Hamiltonian cycle in G. In either case, he has no more information about G than when he started.

Zero Knowledge Proof Method for Graph Coloring:

Suppose Peggy knows a three-coloring for a graph G with N nodes and E edges (again, probably because she generated G, and in doing so guaranteed it had a three-coloring). In general, finding a three-coloring of a graph is NP-Complete. Suppose again that Peggy has engineered her graph to be no exception.

Peggy wants to prove to Victor that she knows a three-coloring of G, but she doesn't want to reveal this coloring in the process of the proof. Peggy can prove to Victor that she knows a three-coloring using the following Zero Knowledge Proof:

1. Peggy selects a random number and uses is to permute the color labeling of G to produce H. G and H are identical in all ways except that the colors used for coloring H have been globally permuted from the original set of colors. This restricts H to one of six possible results of permuting the coloration of G. The resulting three-coloring must consist of colors from a set, say {C, M, Y}, that Peggy and Victor have agreed upon.
2. Peggy then encrypts H to get H'. (She uses probabilistic encryption on each node to ensure that Victor cannot attempt to discover any colors in H by randomly encrypting various colorings and comparing them to H'.) This commits Peggy to her choice of H.
3. Victor randomly selects two adjacent nodes chooses to asks Peggy to decrypt them, revealing that they are colored differently (as they must be if the H is correctly three-colored).
4. Peggy checks that the two nodes are indeed adjacent (to make sure Victor isn't trying to scam her into revealing any information). Then, she provides the decryption for those two nodes, demonstrating that they are indeed colored differently. Victor is convinced provided that the colors are indeed different, the decryption is correct, and the colors are from the agreed set {C, M, Y}.
5. Again, Peggy and Victor repeat steps 1--4 multiple times. This time, however, computing the odds is not so simple.

In the Hamiltonian cycle proof, Peggy had a fifty-fifty shot of getting away with lying, since Victor made a 1 in 2 random selection. In this proof, Victor must assume that Peggy might be very close to having a three coloring, but have two nodes adjacent nodes that she must assign the same color. Victor must assume that Peggy may try to slip this one by him. This makes his odds at detecting her in any given round way worse than 1 in 2.

Let's assume the worst, i.e., that Peggy has a coloring that is wrong for only one pair of nodes. The probability that Victor will pick the correct two nodes to reveal Peggy as a cheater is 1/E for each round. The probability that Peggy gets away with cheating in each round is 1 - 1/E. Thus, the probability that Peggy gets away with cheating in E rounds is (1 - 1/E)^E, which conveniently is less than ½ for E > 0. Thus, for E * D rounds, the probability of Peggy getting caught is ((1 - 1/E)^E)^D or (1 - 1/E)^(E * D) which, in turn, is less than (1/2)^(E * D).

So if Victor insists on repeating for (E * D) = K rounds, he can be confident with probability at least 1 - (1/2)^K that Peggy hasn't cheated him.

References:

Bruce Schneier. Applied Cryptography, Second Edition. John Wiley & Sons, 1996.

Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a method of cryptographic protocol design. In Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS), pages 174-187. IEEE Computer Society Press, 1986.

---

6. Explain time-stamping protocol.

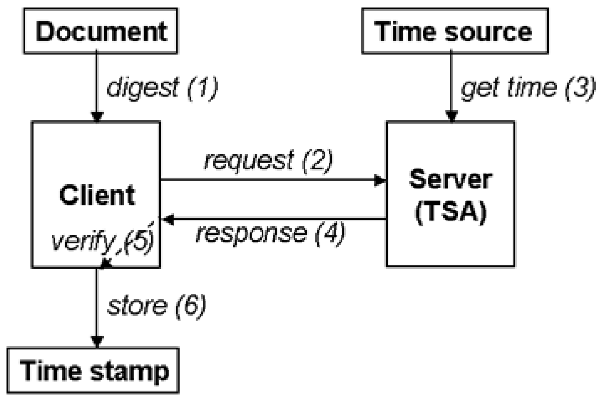
A time stamp logically is an electronic seal including a time indication applied over a document.

In practice it's a digital signature over a submitted digest, a time indication and other information.

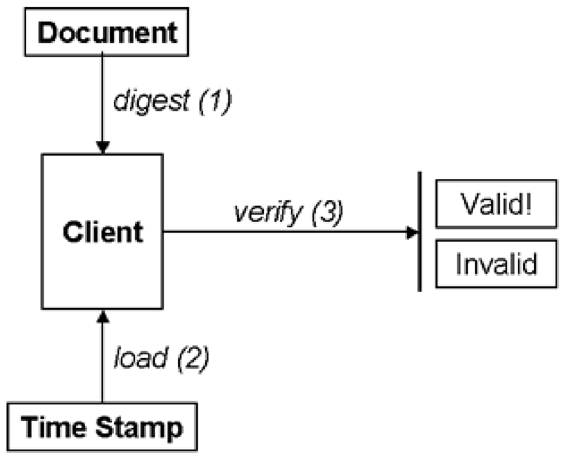
The Time Stamping Protocol defines the entities involved*, the message format and the transport protocol which permits the communication between the entities.

*The requestor (or client) and the Time Stamp Authority (or server)

It's possible to locate two distinct phases:



Phase I: request, issue, immediate verify



Phase II: late verify

7. Explain how poker can be played over phone. Explain the scheme using both physical analogy mechanisms as well as by using cryptographical protocols.

Suppose the two players are called Alice and Bob. In order to play the poker game, Alice puts her lock on each card of the deck.

Then, she suffles the deck and sends the whole deck to Bob. Bob randomly picks five cards for Alice and sends them back to her. Now Alice can unlock her 5 cards and see her hand. Moreover, Bob picks 5 more cards from the remaining cards of the (locked) deck. He also puts his lock on the picked cards and sends them to Alice. Now, Alice unlocks these cards. Since these cards are also locked by Bob, she can not see Bob's hand. Then she sends back Bob's hand and Bob can see his hand by unlocking his locks.

To express this scheme with cryptographical protocols, the commutative cryptosystems must be used. Suppose that Ea and Da are Alice's encryption and decryption funstions, respectively. Similarly, Eb and Db are Bob's. Now if these functions are commutative, we have:
Ea(Db(x)) = Db(Ea(x))
Eb(Da(x)) = Da(Eb(x))
Therefore, the poker game can be played as the following. The deck card is the set {1, 2, ..., 52}. Alice encrypts each card separately, and randomly orders the resulting set {Ea(1), Ea(2), ..., Ea(52) }. She sends this set to Bob. Bob chooses five cards at random, say {Ea(18), Ea(24), Ea(27), Ea(31), Ea(39)}, and sends them back to Alice. Now Alice knows that her hand is : {18, 24, 27, 31, 39}.

Next, Bob chooses 5 different cards form the encrypted deck. Say {Ea(3), Ea(12), Ea(15), Ea(35), Ea(41) } . He encrypts them and sends the randomly ordered set {Eb(Ea(3)), Eb(Ea(12)), Eb(Ea(15)), Eb(Ea(35)), Eb(Ea(41)) } back to Alice. Noe Alice decrypts each element of the set and sends the resulting set
{Eb(3), Eb(12), Eb(15), Eb(35), Eb(41) } back to Bob. Then Bob decrypts the set to get his hand, {3, 12, 15, 35, 41}.

Once the hand has been played, Alice and Bob exchange their encryption keys and verify that each played fairly.

One of the commutative cryptosystems that was initailly proposed for this protocol is the cryptosystem proposed by Shamir et al. which is as the following.

Alice and Bob agree on a large odd prime number n, and separately choose secret keys k=A or k=B, where gcd(A, n-1) = gcd(b, n-1) = 1. Then Ek(x) = x^k (mod n) and Dk(x) = x^z (mod n), where kz = 1 (mod n-1).

Reference:

S. Fortune, M. Merritt, "POKER POROTOCOLS", CRYPTO'84, LNCS 196, pp454-464, 1985.

8. Explain at least four methods for watermarking text. Discuss advantages and limitation. Explain possible attacks.

Technique 1: Line Coding

- Spacing between adjacent lines of text are adjusted up or down to encode data.

Technique 2: Word Coding
- Spacing between adjacent words are adjusted in small amounts.

Technique 3: Character Coding
- Characters are adjusted in small amounts (such as the extending the t in unperceivable amounts)

Technique 4: Text Compression

- Words/Letters are compressed from the average according to small amounts to encode data.

Technique 5: Natural language watermarking.
- Embedding data in the wording, sentence structure, or use of specific words.

Attack Types:
- Retyping the document or passages from the document
- OCR to generate digital copy that can then be reprinted
- Image processing to adjust spacing (ie adding new watermark or in general throwing off existing watermark)

Advantages:
- Techniques 1-4 can be done after the text is written in a highly automated fashion.
- Techniques 1-4 may still be strong after portions of text are removed (highly redundant)
- Technique 5 will survive re-typing and OCR based attacks.

Disadvantages:
- Techniques 1-4 are easy to remove by OCRing the text and reprinting it.
- Technique 5 may have low redundancy depending on variants
- All techniques are can be overwritten with a new watermark, although Technique 5 is stronger than the others.

PLEASE NOTE - Techniques 1-3 were taken out of
http://www.acm.org/~hlb/publications/dig_wtr/dig_watr.html

,9. Explain at least four methods for watermarking audio. Discuss advantages and limitation. Explain possible attacks.

Explain at least four methods for watermarking audio. Discuss advantages and limitation. Explain possible attacks.

1) Put watermark in a frequency that is not perceptible to humans. This is easy to do and can be easily retrieved. However, this can also be easily defeated through filtering and compression.

2) A watermark can be inserted in the time domain be speeding up or slowing down the audio. This would be done in small degrees so that humans could not tell the difference. An attacker could use the same method and modify the time scale. This would destroy the embedded watermark.

3) Embed a watermark signal right after a loud signal. The human brain cannot perceive this. This could be easily detected and erased by an attacker. The attacker could also add noise or do filtering to distort or destroy the watermark.

4) Embed the watermark in the transform domain using FFT(Fast Fourier Transform), DCT(Discrete Cosine Transform), or DWT(Discrete Wavelet Transform). An attacker would have to transform the audio and then look for the watermark. This could be difficult if the attacker does not know which transform to use or where in the data to look for the watermark..

---

10. Explain at least four methods for watermarking image. Discuss advantages and limitation. Explain possible attacks.

1. One very simple method to watermarking an image is to alter each least significant bit (LSB) in each pixel. Ownership is proved by matching the LSB's of all the pixels to the key originally embedded in the image. This method is good because it does not reduce the quality of the image and it is fast and simple. However, it has no robustness: it is easy to remove the watermark just by removing the bits. It is also not immune to geometric distortions (scale/crop/rotate), signal processing (dither/recompression/contrast/color), and compression.

2. A second way to water mark images is to insert small geometric patterns into the image, creating the patterns by varying the brightness levels. Ownership is proved by identifying the geometric patters and matching them to the original patters. This method has a few severe limitations: the amount of information it is possible to encode with geometric patterns is not very large and this method is sensitive to signal processing and compression.

3. Another way to watermark an image is to add or subtract a small random amounts from each pixel. A binary mask of bits is compared with the LSB of each pixel, and if the mask bit and the LSB are equal the small random amount is added, otherwise the small random amount is subtracted (or vice-versa). Ownership is proved by

computing the difference between the original image and the watermarked image and comparing the sign of the all the resulting pixels to check if they match up with the original sequence of additions and subtractions. A drawback of this method is that very small amounts must be added and subtracted from each pixel so that there is no noticeable degradation of image quality. This also makes the the watermark more susceptible to signal processing and compression attacks. The watermark is also susceptible to collusion attacks.

4. Out of all of these, the best is the spread spectrum/DCT method. This method is performed by first applying a frequency domain transformation to the image; typically this is a discrete cosine transformation (DCT). Next, the perceptually significant regions of the image are determined. A key is created from independent, identically distributed samples from a Gaussian distribution. The watermark is then inserted and the inverse DCT is applied. This watermark does not change the image quality. It is also very robust: for most attacks to be successful, the image data would be severely distorted resulting in large degradation of image quality. The attacks that this method is resilient to include compression, dithering, scaling/cropping, and data conversion (rescanning or faxing the image). If the same method is used to add more watermarks, the original still remains. The use of Gaussian distributions to generate the key help make this method more resilient against collusion attacks.

---

11. Explain at least four methods for watermarking video. Discuss advantages and limitation. Explain possible attacks.

1. Watermark each individual frame separately using a well-known still-image watermarking technique.

This has the advantage that (ideally) only one frame is needed to recover the watermark from the video. However, it is limited in that compressing the video for broadcast or storage can damage or destroy the watermark. In practice, many frames may be needed to recover the watermark.

2. Manipulate the frame rate of the video. Studies have shown that the playback speed of a video stream can be varied up to 7% without being noticed by a human observer. Individual scenes or sections of the video can be sped up or slowed down to embed a signature in the video.

This has the advantage that it does not reduce the quality of the video itself and is resilient against compression and clipping. However, the watermark can be destroyed by the same process, if an adversary alters the frame rate of the watermarked video.

3. Manipulation of the low-pass temporal band of the video. The video is decomposed into a low-pass temporal band (the motionless components of the video sequence) and the high-pass temporal band (the actual motion). A watermark can then be embedded into the non-moving portion of the video.

The advantage of this scheme is that it does not affect the quality of compressed video since it only changes non-moving parts of the video, which are generally compressed well by current video compression schemes. The disadvantage is that the watermark is not spread throughout the entire video, but is restrained to the non-moving portion. An attack on the non-moving portion of the video could damage or destroy the watermark.

4. Watermarking during compression. A codec-specific technique can cooperate with the compression scheme and insert the watermark in the process of compressing the video. A scheme proposed by Hsu and Wu makes use of the MPEG standard to embed information in both still frames and groups of pictures that are defined in the standard.

The advantages of this scheme is that it may retain better video quality than performing the compression and watermarking steps separately by making use of the particular compression strategies used by the codec. However, it may not be resilient against attacks that decompress the video and the recompress it.

References:

"Video Watermarking." Retrieved Nov 21, 2003, from
http://www.cmlab.csie.ntu.edu.tw/~candy/watermark/video.html

Chiou-Ting Hsu and Ja-Ling Wu, "Digital Watermarking for Video," in
Proc. of DSP '97, July 1997, Greece.

---

12. Explain active watermarking schemes for audio, image and video.

Active watermarking consists of integrating the watermark as part of the design process. You can watermark a video by slightly shifting or tilting the camera according to a pattern. Further, you can alter the intensity or even number of lights in a controlled manner. Similarly microphone can be moved closer or further from a sound source. The watermark should, of course, be unnoticed in the end product. The Active watermarking techniques allow more freedom and create more complex side effects. By moving the microphone, you could possibly vary the volume, feedback, strength of frequencies. The number of effects created by the active watermarking technique makes the attacker's job more difficult.

---

13. Explain four schemes for watermarking graph coloring solutions.

The basic idea is to add extra constraints that make the coloring solution almost unique.

Solution 1: Add Edges
Input: Graph *G(V, E)* and the message in a binary string form $M = m_0m_1...$
Output: A new graph *G'* with *M* embedded
Algorithm:

```
        copy $G(V,E)$ to $G'(V', E')$
        foreach bit $m_i$
        {
                find the nearest two veritices $v_{i1}$, $v_{i2}$ that are not connected to vertex $v_i$;
                if($m_i = 0$)  add edge ($v_i$, $v_{i1}$) to E'
                else add edge ($v_i$, $v_{i2}$) to E'
}
        return $G'(V', E')$
```

Note: By the nearest two vertices v_{i1} and v_{i2} which are not connected to vertex vi, we mean that i2 > i1 > i (mod n), the edges (v_i , v_{i1}); $(v_i; v_{i2})$ are not in E and (v_i , v_j) is in E for all i < j < i1; i1 < j < i2 (mod n).

Solution 2: Selecting MIS
The idea is to select one or more MISes according to *M*, assign each MIS with one color and then color the rest of the graph.
The MIS containing *M* is constructed in the following way: choose v_i as the first vertex of the MIS, where the binary expression of *i* coincides the first $\lfloor \log_2 n \rfloor$ bits of M; then we cut *vi* and its neighbors from the graph since they cannot be in the same MIS as *vi*; we reorder the vertices and select the next vertex of the MIS based on *M*. When we get a MIS, we color it with one color, remove it from the original graph and start constructing the second MIS if M has not been completely embedded.

Solution 3: Adding Nodes and Edges
Given a random graph $G_{n,p}$ and a message *M* to be embedded. We order the vertices set $V = (v_0, v_1, ... , v_{n-1})$ and encrypt the message into a binary string $M = m_0m_1...$ which is then embedded into $G_{n,p}$ as follows: introduce a new node *v*, take the first $\lfloor \log_2 n \rfloor$ bits from *M*, find the corresponding vertex *v'* and connect it to *v*; take the next $\lfloor \log_2 n-1 \rfloor$ bits and locate the next vertex to which *v* is connected. Continue till we add np edges starting from v and get a new graph $G_{n+1,p}$; introduce another new node if *M* has not been completely embedded. We color the new graph, restrict the coloring scheme to the original graph $G_{n,p}$ and we have a solution with message *M* embedded.

Solution 4: Chopping M

The basic idea is to evenly divide *M* into n pieces. The graph *G(V, E)* is also divided into n subgraphs. Then for each piece M_i of *M*, we can apply the similar technique in solution 1 to embed M_i into subgraph G_i. Here the minimum ratio-cut could be used to divide the original graph into n subgraphs.

---

14. Explain three schemes for watermarking SAT solutions.

Basic notations for Satisfiability problem:

{xi : i=1,2,...n} is a set of boolean variables, and xi' is xi's complement A literal is a variable or its complement A clause is a disjunction (logic-OR, +) of one or more literals. It is true if one of its literals is assigned value 1 A formula is a conjunction(logic-AND, .) of one or more clauses. A formula is satisfiable if there is a truth assignment to the variables, such that all the clauses are true.

Three methods for watermarking SAT solutions:

1-Adding Clauses

Any clause in a formula is a constraint that restricts the solution space. The easiest way to embed a watermark is to add new clauses into the formula . This new clause is generated from the signature and can be used to prove the existence of the signature.

2-Deleting literals

The longer the clause is, the easier it will be satisfied. If we remove literals from each clause, the solution space for watermarked SAT will be a proper subset of the original solution space. The literals to remove are picked based on the watermark we want to embed.

3-Push-out and Pull-back

This is a variant of adding clauses, with the freedom of adding new variables. When adding clauses if we detect that there is a dangerous clause that can make the entire formula unsatisfiable, introduce a new variable. Introducing a new variable increases the solution space, because it serves as "don't care" in the formula. We can use this method to embed watermark into the same formula, but over a larger set of variables, then restrict the solution to the original variable set.

for more information on each technique, please read:

Optimization-Intensive Watermarking Techniques for Decision Problems.

Gang Qu, Jennifer Wong and Miodrag Potkonjak

---

15. Explain differences between horizontal and vertical active objects watermarking schemes. Show examples for both schemes.

In the horizontal watermarking, the design/software is structurally altered by a preprocessing or a post-processing step during its creation. In the vertical approach, the watermark is embedded during a synthesis or compilation step. The specification of the design at one process level is augmented with additional design constraints that correspond to a message. Therefore, after the synthesis step, the design has structural properties that correspond to the signature constraints.

Example 1 of horizontal watermarking) you have bitmaps. Using the least significant bit technique, you add a watermark to the original bitmap.
Example 2 of horizontal watermarking) another example is code obfuscation, which changes the syntax of the code, yet maintaining the semantics.

Example 1 of vertical watermarking) graph coloring (Graph coloring watermarks not the original graph the actual colored graph by adding edges or nodes in intelligent places

Example 2 of vertical watermarking) another example is a register assignment; derived of the original object, source code. Given the same source code to be compiled, different compilers can watermark their solution by watermarking the register

---

16. Palsberg's question

The tool built by Palsberg retrieves the Planted Plane Cubic Tree (PPCT) graph (representing the program's watermark) from the program's heap image by

i) extracting all classes in the heap image which are not standard Java runtime packages—these are the potential node classes

ii) extracting all potential node objects from the heap (instantiating the potential node classes)

iii) checking fields for all potential node objects to see if it can hold an outgoing edge—i.e. a field whose type declaration is a potential node class... determine each such potential edge (Now we have the part of the heap that could potentially have a watermark)

iv) do a graph search through the potential node objects/edges for the PPCT (a subgraph isomorphism problem which is made more tractable by knowing about the PPCT). First, find a potential leaf...then the origin...and then the root node of the subgraph.

---

17. Discuss Agrawal's proposal for preserving privacy in data-mining. Specifically describe how privacy is protected and why their scheme works (listing various steps of the proposal).

The objective in data-mining is to identify patterns in the data so you can classify future transactions. Essentially they want to build decision-tree classifiers without having to know the exact values in the database. The reason is one can do so is that one only needs to know the distribution of data to build decision-trees. That is what Agrawal's proposal does: random errors are added to the data to preserve privacy, and then original distribution is reconstructed from the perturbed data. Because random errors are added to the original data, it is not possible for anyone to reconstruct the original data. However, it is possible to reconstruct the original distribution from perturbed data because we know the distribution of the errors that were added to the original data. Bayes' rule can be applied iteratively to reconstruct the original distribution & the paper presents an algorithm to do so.

---

18. Explain at least two schemes for watermarking in biological computations.

Most forms of biological watermarking deal directly with DNA, since it can naturally contain a large amount of data. A great deal of DNA is, by it's very structure, never transcribed and turned into protiens. This gives us an opportunity to insert whatever data we want into the DNA, including an appropriate watermark.

As a concrete example, let's say we have a watermark as a series of bytes. We know that there are four base pairs, abbreviated A, C, T, and G. Since there are four possible bases, we can use assign a pair of bits to each of them:

A = 00, C = 01, T = 10, G = 11

Of course, this is completely arbitrary. Once we have done this, we can encode our watermark into a sequence of base pairs by converting each individual pair of bits to one of the base pairs. We can then insert this sequence into a segment of non-transcribed DNA, thus hiding a watermark without distrubing the operation of the DNA itself.

This sort of watermark is very easy to remove, since the processes for discovering sections of unused DNA are fairly simple, and an attacker could simply randomly change them.

Another possibility is to embed data into useful data. For some amino acids, there are multiple codons (i.e. groups of three DNA base pairs) which will map to it during the transcription phase of protien synthesis. Thus changing a codon to one which will emit the same amino acid would do nothing to the operation of the DNA segment. This gives us the opportunity to once again hide a watermark.

Assume for the moment our watermark is a real number of arbitrary precision such that 0 <= W < 1. We take the DNA sequence, and turn it into a sequence of amino acids, as it would be transcribed by the body. Each amino acid has a set of possible codons which map to it. For each codon, we create a subrange of the real numbers between 0 and 1.

As an example, let's take the amino acid abbreviated as Val. There are four codons which map to it: GUU, GUC, GUA, GUG (note that the codons are transcribed as RNA, and thus all T's are replaced with U's). If this amino acid appeared in a DNA sequence, we would create the following mapping:

GUA [0, 0.25)	GUC [0.25, 0.5)	GUG [0.5, 0.75)	GUU [0.75, 1.0)

After creating a sequence of these codon sets, and equivalent ranges, we start at the beginning with the Watermark W. For each of these sets, we figure out which range W belongs to. The codon which is attached to that range will be the one which is used in the final DNA sequence. Now, we rescale W to the range [0, 1). Given the L is the lower bound of the found range, and U is the uppper bound, we set W = (W - L) / (U - L). We then continue on to the next range set, and repeat until either we run out of sets, or we have enough data to reliably represent our watermark.

To retrieve the watermark, we effectively reverse the process, unscaling the values as we go back along the amino acid chain, until we retrieve the original watermark value. It is important that the ranges on the retrieval are the same as the ranges used to insert the watermark in the first place.

Since any DNA sequence will yield some number, we have to somehow prove that that number is one that we placed there. An easy way to use a one way function on the real watermark data, which yields the data which you actually insert into the DNA. Then you can prove that you intentionally encoded the data, since reversing the one way function is by definition difficult
This mechanism has a similar disadvantage to the first, where someone could simply perturb the different codons while preserving the transcribed amino acids. This version does have the advantage that such perturbing takes much more effort on the part of the attacker than before, since the DNA has to actually be read, analyzed, and recreated, as opposed to just spliced.

---

19. Explain fairness and credibility issues in watermarking.

Simply, it means that it is just as easy, and the results are just as good, to add one watermark as it is to add a different one. More specifically, it should take approximately the same amount of computation time to add any particular watermark. For example, if there is a scheme to watermark a particular NP-hard problem, for instance SAT, and this scheme results in a situation where one particular watermark limits the underlying instance of SAT to exactly one solution (making it hard to find the solution), whereas a different watermark (of the same length, quality, etc) hardly affects the number of possible solutions (making it relatively easy to find a solution), that scheme is not fair.

Fairness further means that the results are just as good, e.g. the quality of the result is approximately the same. Quality of the result certainly depends on the problem. One example could be a watermarking scheme for images. If there is a scheme where one particular watermark does not effect the image very much (i.e. the signal to noise ratio is almost the same as before the watermark was added), but a different watermark (of the same length, quality, etc) drastically reduces the quality of the image (i.e. lowering the signal to noise ratio significantly), that scheme is also not fair.

Credibility is akin to 'how good is the watermark?' In other words, what are the chances that it's faked? First, if someone claims a watermark of a random sequence of bits, there's always the suspicion that they first found the set of bits in the supposedly watermarked object and then created their watermark to match it. To avoid this problem, the watermark should be a function of some unique information of the claimant. (i.e. Name, Birthday, Tax ID number, etc) It is even more convincing if the watermark is a secure hash (provably hard to reverse) of the unique information. Once convinced that the watermark actually belongs to the claimant, evaluation of the likelihood that the watermark appears in the object by chance is in order.

If the watermark is a single bit, then assuming that the particular bit could equally likely take either value, there's a fifty percent chance that it could appear as it is by chance. However if there are two bits, one is three times as likely to be a one as a zero, the other is equally likely to be either a one or a zero, the signature is zero one and matches the object, then the likely hood that it occurred by chance is one eighth. By extension: taking the product of the likelihood of being the value observed over each bit in the signature will give the probability that the signature occurred by chance. If the value is small, there is high credibility to the claim of the watermark.

---

20. Explain computational forensic engineering techniques. Discuss, in technical detail their application to intellectual property protection. Show at least one example.

Forensic analysis is used in scientific applications and various fields of art such as anthropology, science, literature and visual art, to gain a statistical measure of similarity between objects being compared. Its most common application is DNA identification.

IP being the principal source of revenue for the semiconductor and software industries, there is significant interest in IP protection techniques. At present, watermarking is the most popular form of establishing copyrights. It requires the owner to embed a signature into the design. Watermarking is susceptible to reverse engineering and adversely affects system performance and code modularity.

"Computational Forensic Engineering (CFE) aims to identify the entity that created a particular IP. … Rather than relying on watermarking designs, the CFE methodology analyzes the statistics of certain features of a given IP and quantizes the likelihood that a well known source has created it. … Formally, given a solution S_P to a particular optimization problem instance P, and a finite set of algorithms A applicable to P, the goal is to identify with a certain degree of confidence that algorithm A_i has been applied to P in order to obtain solution S_P." [1]

The CFE approach is to first identify a set of properties of S_P that will help to distinguish the algorithms in A and use the properties to compute a measure of similarity $z(A_i, A_j)$ between any two algorithms. The algorithms are then clustered in an n-dimensional space (where n is the total number of properties). Two algorithms remain in the same cluster if $z(A_i, A_j)$ is greater than some predetermined bound $\epsilon << 1$. If an algorithm A_x is clustered with algorithm A_y, and A_y is not clustered with any other algorithm, then A_y is a copy of A_x.

The generic CFE method works in four steps namely, i) data collection, ii) feature extraction, iii) algorithm clustering and iv) validation.

Data Collection:
Create perturbations of the problem instance P to eliminate algorithm dependency on naming or form of the input. Run each algorithm in A on all perturbed instances to obtain a set of solutions.

Feature Extraction:
(a) Identify relevant functional and structural properties π_k of S_P. Discard properties that don't help to distinguish algorithms.
(b) Compute feature quantifier, ω_k, for each property to help characterize solutions created by the algorithms.
(c) apply fast algorithms for extraction of selected properties from the set of solutions.

Clustering:
(a) Using the properties and quantifiers, compute the correlation $z(A_i, A_j)$ between any two algorithms. (refer to [1] for details of function z).
(b) Cluster the algorithms using the following method. (For pseudo-code refer to [1])
Initially, C = Null.
If Ai correlates to all algorithms in an existing cluster, add Ai to that cluster.
If Ai has high correlation with a subset of algorithms in a cluster, split the cluster into two clusters, {subset, Ai} and {original cluster - subset}.
If Ai does not correlate with any cluster, form a new cluster with Ai as the only element.

Validation: Apply non-parametric re-substitution software to establish validity of the ability to distinguish distinct algorithms.

Example Boolean Satisfiablility.
A: GSAT, WAlkSAT, NTAB, Rel_SAT_rand
Using 5 properties, CFE was shown to correlate solutions to the algorithm that produced them.

References
[1] Jennifer L Wong, Darko Kirovski, Miodrag Potkonjak, *Computational Forensic Techniques for Intellectual Property Protection.*
[2] Jenni's class presentation, *Statistical Forensic Engineering Techniques for IPP*

---

21. Explain differences between localized and distributed watermarking.

The localized watermarking carries the following attributes while the distributed watermarking does not address them:

Protection of design partitions: Distributed watermarking does not provide protection for design partitions. Namely, in many designs (cores), their parts may have substantial and independent value (for example, a discrete cosine transform filter in an MPEG codec).
Copied partition detection: Commonly, misappropriated designs or their parts are augmented into larger designs. This leaves no room for the existing protection techniques
to facilitate the existence of a part of the watermark in a design as a proof for authorship.

Effective signature detection: Since the encoding of a digital signature is dependent upon the structure of the entire design specification, detecting an embedded signature requires unique identification of each component of the design. Moreover, possible design alteration by the misappropriator may negligibly, but significantly alter the design in a way that restoring the identifiers of design components requires detection of a number of subgraph isomorphisms.

---

23. Explain key issues in reverse engineering.

Reverse engineering is taking apart an object to see how it works in order to duplicate or enhance the object. It's a practice taken from older industries that is now frequently used on computer hardware and software. (http://whatis.techtarget.com/definition/0,,sid9_gci507015,00.html)

In the automobile industry, for example, a manufacturer may purchase a competitor's vehicle, disassemble it, and examine the welds, seals, and other components of the vehicle for the purpose of enhancing their vehicles with similar components.
In conventional industries, reverse engineering involves producing 3-D images of manufactured parts when a blueprint is not available in order to remanufacture the part. To reverse engineer a part, the part is measured by a coordinate measuring machine (CMM). As it is measured, a 3-D wire frame image is generated and displayed on a monitor. After the measuring is complete, the wire frame image is dimensioned. Any part can be reverse engineered using these methods.
Reverse engineering is the process in which a manufactured part is measured without the benefit of a blueprint for dimensional reference. The results are then reviewed, refined, and the part re-manufactured.

Hardware reverse engineering involves taking apart a device to see how it works. For example, if a processor manufacturer wants to see how a competitor's processor works, they can purchase a competitor's processor, disassemble it, and then make a processor similar to it. However, this process is illegal in many countries. In general, hardware reverse engineering requires a great deal of expertise and is quite expensive. For example, Chipworks (www.chipworks.com) is a Semiconductor Reverse Engineering company that analyzes the circuitry and physical composition of semiconductors and electronic systems for a wide range of applications in patent licensing support and competitive study since 1992.

Software reverse engineering involves reversing a program's machine code (the string of 0s and 1s that are sent to the logic processor) back into the source code that it was written in, using program language statements. Software reverse engineering is done to retrieve the source code of a program because the source code was lost, to study how the program performs certain operations, to improve the performance of a program, to fix a bug (correct an error in the program when the source code is not available), to identify malicious content in a program such as a virus, or to adapt a program written for use with one microprocessor for use with a differently-designed microprocessor.

Someone doing reverse engineering on software may use several tools to disassemble a program.
Typical examples of reverse engineering tools are disassemblers and decompilers, which translate an object file produced by some compiler into an ASCII representation. A good source of information can be found at http://www.backerstreet.com/cg/work.htm.
For example, REC is a portable reverse engineering compiler, or decompiler.
It reads an executable file, and attempts to produce a C-like representation of the code and data used to build the executable file. Another common tool is the disassembler. The disassembler reads the binary code and then displays each executable instruction in text form. A disassembler cannot tell the difference between an executable instruction and the data used by the program so a debugger is used, which allows the disassembler to avoid disassembling the data portions of a program. These tools might be used by a cracker to modify code and gain entry to a computer system or cause other harm.
Another tool is a hexadecimal dumper, which prints or displays the binary numbers of a program in hexadecimal format (which is easier to read than a binary format). By knowing the bit patterns that represent the processor instructions as well as the instruction lengths, the reverse engineer can identify certain portions of a program to see how they work.

Software reverse engineering involves the following four steps (http://www.acm.uiuc.edu/sigmil/RevEng/)
(1) Observe and describe a phenomenon or group of phenomena
(2) Formulate a hypothesis to explain these phenomena.
(3) Either try to use your hypothesis to predict new events, or attempt to find events that demonstrate your hypothesis is incorrect or incomplete.
(4) Use your hypothesis to gain insight into the system, and perhaps even write some code.
The most important thing to remember is that this is an iterative process.
It converges on a solution through repetition of observation, guessing, testing, and predicting (coding).

Reverse engineering for the sole purpose of copying or duplicating programs constitutes a copyright violation and is illegal. In some cases, the licensed use of software specifically prohibits reverse engineering.
For example, The 1998 Digital Millennium Copyright Act (DMCA) made it a crime to circumvent technology used to protect copyrighted material, even though some may argue that DMCA gives too much protection to copyright holders at the expense of innovation and lower prices.
However, it is protected under DMCA that reverse-engineering a product for the purposes of making another product work with it -- the concept of interoperability is legal. A very recent article on this issue can be found at http://www.businessweek.com/technology/content/nov2003/tc2003114_5174_tc024.htm.

---

24. Explain Trojan horse and virus attack and defense techniques.
http://www.ee.ucla.edu/~bgrot/cs259/final_q24.html

Virus – Like a biological virus, uses a host (a program, operating system, disk's boot sector, etc) to attach itself to, replicate, and spread

- 3 key characteristics: replication mechanism, activation mechanism, and an objective
- Behavior (objective) can range from highly destructive to completely benign, such as a one-time message to raise awareness of AIDS.
    - Example of a highly destructive virus that ties in very well with the class discussions is presented in **Cryptovirology: Extortion-Based Security Threats and Countermeasures (1996)**, Adam Young, Moti Yung.

    - The main idea is a cryptovirus, which, using public-key encryption, would hijack the computer by encrypting user-data and decrypt it in return for a ransom.

Trojan horse – a malicious program disguised as something benign

- Typically, operates stealthily, unbeknownst to the user.
- Often used to capture information, such as username & password, from the system or create a backdoor into the system to give an attacker control
- Unlike a virus or worm, a Trojan does not replicate and propagate on its own – it must be downloaded or intentionally transferred by a person.

Worm (FYI – not on the exam) – replicates and infects other machines via a network connection

- Unlike a virus, a worm is self-contained (does not need a host program to propagate itself)

Defense:

- In order to successfully defend against viruses and Trojans, we must be able to detect them (the rest – identification and removal – is usually straightforward and easy)
- 5 classes of detection techniques:
    - Signature scanning and algorithm detection
        - Typical anti-virus program
        - Intrinsically limited to detection of known viruses (or at least, a known signature)
    - General Purpose Monitors
        - Protect the system from viruses and Trojans by actively monitoring for suspicious actions
        - Can detect unknown viruses, since no static information is required
    - Access Control Shells
        - Part of the OS
        - Enforce an access control policy at system level to identify unauthorized attempts to access/modify a file
    - Checksums for Change Detection
        - Based on the theory that executables are static objects; thus, any modification implies a possible virus infection
        - In reality, some exec's are self-modifying or may change over time (for instance, in a development environment)
    - Heuristic Binary Analysis
        - Relies on an analyzer to trace through the executable looking for suspicious behavior

In recent years, computer systems have been modeled as artificial immune systems which incorporate many properties of their biological counterparts, including dynamic learning, adaptation, and self-monitoring.

References:

1. Using Mobile Agent Results to Create Hard-To-Detect Computer Viruses. Wang, Yongge.
2. A guide to the selection of anti-virus techniques. W. Polk, L. Bassham.
3. An Overview of Computer Viruses and Antivirus Software. Karnish, Bob.
4. Wickipedia.org