3. [Trusted computing]

The following protocol, described in lectures, is designed to enable an appropriately equipped computer to provide unforgeable guarantees about the application software it is running. The protocol relies on the computer hardware having a public/private key pair, PKH and SKH, and firmware to run the protocol. (In trusted computing, the hardware keys are called the endorsement key, and the device that stores them is called the TPM.)

Attestation protocol
- When an application A is started, it first generates a new public/private key pair PKA and SKA, called the attestation identity key (AIK). The application requests the hardware to certify its public key. The certificate CA = SignSKH(PKA,#A) returned by the hardware includes a hash #A of the executable A.
- When the application wants to attest its validity to a remote server, it sends the certificate (PKH,CA) to the server. The server checks:
  - The signatures are valid, and PKH has not been revoked.
  - The application hash embedded in CA is on the server’s list of applications it trusts.
- The application now authenticates itself by proving knowledge of SKA. For example, the application and the server can run a key exchange to generate a session key.

(a) Suppose a television broadcasting company wants to implement a "view again" service on its website which allows users to play video content, but prevents them from recording it or copying it. Can this protocol help them? Explain your answer. [11%]

Yes, trusted computing and attestation are suited to digital rights management applications. The application A would be the application for playing the video content. It would prevent recording or copying. The company can deliver the content only if the client attests to be running the correct application.

(b) Suppose Microsoft wants to write a new generation of web browser, called “Internet Explorer Ultimate”, which is immune from attacks from viruses and buffer overflows. Can this protocol help them? Explain your answer. [11%]

No, this protocol is not useful for that purpose. Trusted computing can’t help to avoid vulnerabilities in software.
This protocol (which is not part of the Trusted Computing Group specification) has been criticised because it could compromise the privacy of users. Explain why this is so. Briefly explain how protocols which have been adopted by the Trusted Computing Group have addressed this privacy concern.

In the protocol, the client computer reveals its long-term unique public key PKH to the server. That means that the server can link transactions from that client. For example, the server could build a profile associated with PKH of the kinds of video content viewed, the frequency, and other habits. This compromises the privacy of users. The Trusted Computing Group has developed two protocols to address this. The first one uses a Privacy CA trusted party to certify a temporary key, which the user can change. The other one, called Direct Anonymous Attestation, does a similar job but avoids the necessity of a trusted party.

4. [Secure hash functions]

(a) What is a secure hash function? [11%]

A secure hash function is a function taking an arbitrary bitstring and returning a fixed-size bitstring, such that

* It is easy to compute the hash value for any given input message,
* Given a hash value, it is infeasible to find an input that produces that value
* Given an input, it is infeasible to find a different message that has the same hash,
* It is infeasible to find two different messages with the same hash.

(b) SHA1 is a 160-bit (or 20-byte) secure hash function. Assume that it is perfectly secure. What is the probability that, if I take a random message M and compute its hash value, I will get the result

2e ae 70 47 5c 2d 90 ce 07 d0 84 53 ce 27 bc a8 fb 35 3d f1

Since the output hash values are uniformly distributed in the set of bitstrings of length 160, the probability is $1/2^{160}$.

(c) A software company uses the following method to distribute software to its clients.

- It puts the software on an insecure web server
- The client downloads the software and computes an SHA1 hash of the downloaded file. She then telephones the software company, and reads out the first two bytes of the SHA1 20-byte hash (that would be 2e ae in the example above). Again, we assume that SHA1 is perfectly secure.

-2-
If the bytes read out agree with the company's record of the hash value, the client accepts the software as securely transmitted.

Estimate the difficulty an attacker would have to create malware which, when downloaded in this way, would result in a hash value in which the first two bytes are correct. Since SHA1 is considered secure, the attacker can only use brute force. Your answer should be an estimate of the computation time required by the attacker. You can make any reasonable assumptions about the computation power of the attacker and the time to compute the hash of a file, but you should state those assumptions. [12%]

The attacker might take the following approach. First, he creates the malware, leaving a block of data at the end that can be manipulated in order to get the right first two bytes of hash value. Next, he repeatedly tries different values for the data block, and computes the hash value of the whole. If the first two bytes of the hash agree with the first two bytes of the hash of the real program, he is done. Otherwise, he tries different data in the block, and repeats.

If the block of data is reasonably long, then each hash value computed in this way has approximately $1/2^{16}$ chance of being correct. That means the attacker has to perform about 45500 hashes to have a 50/50 chance of getting it right. (The probability of being wrong a given time is $1-1/2^{16}$, so the probability of being wrong 45500 times is $(1-1/2^{16})^{45500}$, which is about 0.5.) Suppose a reasonable computer can perform 10 hashes per second (we assume the program is quite big). Then it will take about 4550 seconds, which is a bit more than an hour.