COMP6451 Summer 2018-19
Assignment 1
Total Marks: 20
(all questions worth equal marks)
Due: 12:00 Wed Jan 30

Question 1: (Q1, CH1 from the textbook) You are designing SecureBox, an authenticated online file storage system. For simplicity, there is only a single folder. Users must be able to add, edit, delete, and retrieve files, and to list the folder contents. When a user retrieves a file, SecureBox must provide a proof that the file hasn't been tampered with since its last update. If a file with the given name doesn't exist, the server must report that – again with a proof.

We want to minimize the size of these proofs, the time complexity of verifying them, and the size of the digest that the user must store between operations. (Naturally, to be able to verify proofs, users must at all times store some nonzero amount of state derived from the folder contents. Other than this digest the user has no memory of the contents of the files she added.)

Here's a naive approach. The user's digest is a hash of the entire folder contents, and proofs are copies of the entire folder contents. This results in a small digest but large proofs and long verification times. Besides, before executing add/delete/edit operations, the user must retrieve the entire folder so that she can recompute the digest. Alternatively, the digest could consist of a separate hash for each file, and each file would be its own proof. The downside of this approach is that it requires digest space that is linear in the number of files in the system.

Can you devise a protocol where proof size, verification time, and digest

size are all sublinear? You might need a sub-protocol that involves some amount of two-way communication for the user to be able to update her digest when she executes and add, delete, or edit. Hint: use the Merkle tree idea from Section 1.2.

Question 2: Consider paper wallets, generated by the following process (from the user's point of view):

- the user points their browser to a `https` secured webpage, hosted by a reputable Bitcoin exchange, that provides paper wallet production functionality. (In order to attract potential customers to their site, the exchange offers this functionality openly, and does not require users to have an account with the exchange or to authenticate themselves in order to access the page.)
- the user disconnects their machine from the internet
- the user runs a Javascript function on the webpage (still showing on their browser) that asks the user to move the mouse as a source of randomness
- the function produces a page showing a Bitcoin private key and address
- the user prints this page and puts it into their home safe
- the user copies the address, closes the webpage and reconnects to the internet
- the user publishes the address on their `https` secured webpage with the message "send me money here".

Explain some risks involved in this process, by describing at least 3 distinct attacks (*not* including a brute force attack to find the private key associated to the user's address) that an attacker might try to use to remotely (i.e., without breaking into the users' houses) steal the money sent to addresses generated by this process. For each, describe what the user can do to mininize the probability that the attack is successful.

Question 3: We noted in class that the simple secret sharing scheme has the property that compromise of one share decreases the cost of a brute force attack, because the attacker now knows some of the bits of the key.

Consider the following alternative. Let k be the randomly generated private key, represented as a number mod n for some appropriate n. Generate p random numbers $x_1, .., x_p \in [0, n-1]$ uniformly at random and let $x_{p+1} = -(x_1 + ... + x_p)\ mod\ n$

Define the $p+1$ shares to be

$$k + x_1\ mod\ n, \quad k + x_2\ mod\ n, \ ....\ , \quad k + x_{p+1}\ mod\ n$$

Prove the following:

1. Recombining the $p+1$ shares allows k to be reconstructed (explain how).

2. An attacker who compromises any p of the $p+1$ shares still gets no information about k. That is, any key k' is consistent with the p shares, with every possible key k' having the same probability, from the attacker's point of view, of being the secret k.

Question 4: Suppose that h is a hash function taking long messages as input and producing 256 bit outputs and that M is a long message. Consider the following idea for constructing a signature scheme:

- A private signature key Ks is a pair of randomly generated sequences of of 256 bit messages $x_1, ..., x_{256}$ and $y_1, ..., y_{256}$.

- The corresponding public verification key Kv is the pair of sequences $h(x_1), ..., h(x_{256})$ and $h(y_1), ..., h(y_{256})$

- To sign message M, where the hash $h(M)$ is the sequence of bits $b_1...b_{256}$, define $sign(M, Ks) = (M, z_1...z_{256})$ where $z_i = x_i$ if $b_i = 0$ and $z_i = y_i$ if $b_i = 1$.

1. Explain how you could verify this signature is correct: what does the verification function $V(Kv, (M, r_1 \dots r_{256}))$ do to check that $(M, r_1 \dots r_{256})$ is message M signed using the signature key corresponding to Kv?

2. Explain why this scheme is secure - why is it hard for an attacker to forge a signed document that passes the test? Clarify what assumptions on the hash function are needed for your argument.

3. Is it safe to use the private key to sign more than once? If not, explain what an attacker could do to attack a user who does this.

Question 5: By a large amount, the majority (reportedly, as much as 70%) of Bitcoin mining power is currently said to be based in China, and a smaller proportion in the rest of the world. In no more than a page, describe what would be the effect on the Bitcoin ecosystem (i.e., users, miners, and exchanges) both inside and outside of China, if the Chinese government were to block all internet traffic in and out of the country for a year, and then reconnect the Chinese internet to the rest of the world. If there are negative impacts, what could be done to defend against or recover from those negative impacts?