Name: Dan Reynolds
User id: djreynol
Student Number: 20473104

# A2

question 1)

part a)

i)

Mallory can fool Alice into believing she is Bob by using a reflection attack. Mallory first opens a chat with Alice, and waits for Alice to initiate the protocol by sending $r_1$. Mallory then opens another chat with Alice, but this time initiates the protocol herself by sending the same $r_1$ to Alice.

In the second chat, Mallory will then receive $r_2$ and $y_1$ from Alice. Mallory forwards these values to the first chat with Alice, who then verifies $y_1$ by making $x_1$ and sends back $x_2$ to Mallory. Mallory does not even verify $x_2$ and now has finished the protocol with Alice believing that she is talking to Bob.

This was possible because the names were now missing from the protocol, and Mallory could use other chats with Alice to generate values from the function.

ii)

Mallory sends an $r_1$ to Bob, getting Bob's response of $r_2$, $y_1$ and then saves $r_1$, $y_1$, and $r_2$, and waits for Bob to go offline. Mallory then repeatedly opens chats with Alice, always waiting for Alice to initiate the protocol by sending $r_1$. Mallory continues doing this until Alice sends Mallory an $r_1$ that has the same lower 14-bits as the $r_1$ Mallory used earlier with Bob. Once a match is found, Mallory sends back her saved $r_2$ and $y_1$ from Bob to Alice. Alice will calculate $x_1$ and see that it matches Mallory's saved $y_1$, since $2^{50} \cdot r_1 + r2$ will give the same result as the earlier time with Bob, since we shift the top 50 bits off of $r_1$ and only need the bottom 14-bits to match, which is what Mallory was waiting for.

Alice will then send back $x_2$ to Mallory, who doesn't verify $x_2$ and now has finished the protocol, tricking Alice into believing Mallory is Bob. The number of times that Alice may have to wait before getting a matching bottom 14-bits is $2^{14} = 16384$. At 5 protocol initiations per second, this would take ~3280 seconds, which is less than 55 minutes.

iv)

Bonus:

Mallory monitors all of the protocol initiations between Alice and Bob, recording all of the values they pass back and forth. Mallory then keeps opening chats until one of the $r_1$ values matches one of her saved values. At some point, Alice will reboot her computer and the random number used for $r_1$ will be the same as one of the ones Mallory has saved. Mallory then can use the other values she saved from that particular protocol initiation to respond with an $r_2$ and the correct $y_1$ given the matched $r_1$ value. She completes the protocol using the values she saved from Bob and Alice ends up believing that she is talking to Bob.

Mallory may be required to have a lot of space to record all of the sets of values from each protocol initiation between Alice and Bob before Alice reboots.

part b)

The bank uses three authentication factors to identify Alice. The first is something she knows, which is her PIN. Rather than allowing her to change it over the phone, the bank also uses something she has, her bank card, by telling her she must change it at one of the bank's ATMs. Finally, the bank uses something related to context, her location, specifying that Alice must authenticate at one of their banks.

Alice initially identifies the bank by calling her bank's phone number. Alice then identifies the bank by going to one of the bank's machines to change her PIN.

The bank identifies Alice by her card number when she is instructed to go put her card into one of their ATMs to perform the change.

question 2)

part a) Alice can read D105 and write to D104.

part b)

i) write D101:

D101: (C, {D}), Alice: (S, {B,D,E})

ii) read D102:

D102: (S, {A, B, D}), Alice: (S, {B, D})

iii) read D103:

D103: (TS, {B, E}), Alice: (S, {B})

iv) write D104:

D104: (S, {B}), Alice: (S, {B})

v) read D105

D105: (C, {D, E}), Alice: (C, {})

question 3)

part a)

New FAR:

Using a binomial distribution, the new FAR can be calculated by determining the probability that a stranger could get 4, 5, or 6 correct swipes:

$$\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

$$\binom{6}{4} \cdot (0.05)^4 \cdot (0.95)^2 + \binom{6}{5} \cdot (0.05)^5 \cdot (0.95)^1 + \binom{6}{6} \cdot (0.05)^6 \cdot (0.95)^0 = 0.000086$$

Therefore there is a 0.0086% chance that a stranger gets a false acceptance under the new system.

New FRR:

Again, using a binomial distribution we can calculate the new FRR by calculating the chance Alice is recognized by her phone, getting 4, 5 or 6 swipes, and subtracting this from 1:

$$\binom{6}{4} \cdot (0.9)^4 \cdot (1-0.9)^2 + \binom{6}{5} \cdot (0.9)^5 \cdot (1-0.9)^1 + \binom{6}{6} \cdot (0.9)^6 \cdot (1-0.9)^0 = 0.98415$$

Therefore the FRR is 1 - 0.98415 = 0.016, and there is a 1.6% chance that Alice is rejected when using her own phone under the new system.

part b)

If the cellphone locks, then the probability that it was a stranger can be described by conditional probability:

$$\begin{aligned} P(stranger|locks) &= \frac{P(stranger) \cdot P(locks|stranger)}{P(locks)} \\ &= \frac{P(stranger) \cdot P(locks|stranger)}{P(Alice) \cdot (FRR) + P(stranger) \cdot (1-FAR)} \\ &= \frac{(0.08) \cdot (1-0.000086)}{(0.92) \cdot (0.016) + (0.08) \cdot (1-0.000086)} = 0.845 \end{aligned}$$

Therefore there is an 84.5% chance that when the phone locks, it is because there was a stranger who tried to access it.

## Programming Bonus

1. The attacker could send a packet spoofed to have a source IP of one of the other machines on the LAN and going to one of the sinkhole destinations. The sinkhole detector would then claim that the source of the DNS query was that machine on the LAN, framing the machine.

2. An insertion attack uses the fact that an IDS may accept certain things that the actual system will not allow. A web request could be invalid, for example, to the actual system, but in terms of what our IDS sees, it will just look at a HTTP request and analyze the URL for potentially dangerous unicode encodings.

A worm could avoid detection by our IDS by making a request for a URL that would be modified by the actual system because it is invalid, but not modified by the IDS. If the string contained an invalid character that the system would strip out, but that our IDS would not, then the IDS will include the character and not pattern match a dangerous unicode character, while the actual system will strip it out resulting in a dangerous use of `\` or `/` resulting in a directory traversal exploit.