Homework 2 – Xavier Gitiaux

March 2019

1 Exercise 1

1.1 Question 1

The customer cannot compute x since it can only hash m_i and not m_j for $i \neq j$. Therefore, to verify whether m_i was signed, she needs a string $s_{1i} = h(m_1)||...||h(m_{i-1})$ and $s_{2i} = h(m_{i+1})||...||h(m_{10})$. Then she would be able to compute $h(m_i)$, insert it between s_{1i} and s_{2i} hash $H(s_{1i}||h(m_i)||s_{2i})$ as $x^{'}$ and check whether $Sign_{pk}(x^{'}) = s$.

1.2 Question 2

To trick the customer, the company would have to forge a $x^{'} = H(h^{'})$ $(h \equiv H(m_1^{'})||...||H(m_{10}^{'}))$ such that $x = x^{'}$ but either with $H(m_i) = H(m_i^{'})$ and $h = h^{'}$ or with $H(m_i) \neq H(m_i^{'})$ and $h \neq h^{'}$. In the former case, m_i and $m_i^{'}$ will be a collision of H; in the latter case, h and $h^{'}$ will be a collision of H. Both cases violate the collision-resistance assumption of H.

1.3 Question 3

The company signs 10 messages at a time so in a second, it proceeds 100 messages since each signature takes $0.1s$. Without its new scheme, the company will only proceed 10 messages per second.

1.4 Question 4

To each customer i, the company sends x and the 9 hashes of m_j for $j \neq i$. Therefore, it sends 320 bytes of additional information.

1.5 Question 5

We could use a Merkle binary hash tree with messages $m_1, ..., m_K$ as leaves (where K will be computed below). To each customer i, the company sends m_i, the hash of m_i's sibling and of sibling of m_i's parent and so on to the root. For a tree with K leaves, the number of hashes that have to be included to what is sent to the customer is equal to the depth of the tree $log(K)$, so the total number of bytes is $32 * log(K)$. If the company needs $0.1s$ to sign and if they want to proceed 10^7 messages per second, it means that in a second they can sign at most 10 hashes with 10^6 messages each. Therefore $K = 10^6$ and the number of bytes transferred to each customer is $32 * log(10^6) \sim 638 < 1000$.

2 Exercise 2

2.1 Question 1

Step 3

- confirms to B that he is talking to A since only A was able to decrypt c_2;
- and it prevents a man-in-the-middle attack with an adversary intercepting c_2, replacing it with $c_2^{'} = Enc_a(N^{'})$ and forcing A to use a wrong key $N_a \oplus N^{'}$. B would know if this type of attack happens since c_3 would not decrypt to N_b.

2.2 Question 2

- Upon receiving $c_1 = Enc_{pk_P}(A, N_a)$, P decrypts with his private key sk_P and gets (A, N_a), which he encrypts using B's public key: $c_1^{'} = Enc_{pk_b}(A, N_a)$.
- P sends $c_1^{'}$ to B who decrypts it and thinks he is talking to A.
- B encrypts N_b, N_a with A's public key pk_A and send it to A: $c_2 = Enc_{pk_A}(N_b, N_a)$.
- P intercepts c_2 and sends it to A, who thinks it is coming from P. Therefore, A will encrypt N_b with P's public key and sends $c_3 = Enc_{pk_P}(N_b)$ to P.
- P decrypts c_3 with his private key sk_P and re-encrypt it with B's public key: $c_3^{'} = Enc_{pk_B}(N_b)$.
- B receives $c_3^{'}$ and decrypts. Since he gets the nonce he sends, he thinks that he is talking to A and encrypts his message with the key $N_a \oplus N_b$.
- P has both nonces N_a and N_b. Therefore, he can communicate with B as if he were A.

2.3 Question 3

In Step 2, B should encrypt (B, N_a, N_b) with A's public key. P cannot alter this message since he does not have A's private key. But if he sends it to A as a response to the initial message c_1 A sends to P, A will know that N_b was not produced by P but by B and should abort the protocol (A is expecting $Enc_{pk_A}(P, N_a, N_b)$).