Homework 7

Erik Brakke

November 5, 2015

Collaborators: Alison Kendler.

Answer 1

We have m, σ' and (N, e) we can do the following:
$\sigma'^e = h^{de} \bmod q$
And we also know that $\sigma'^e \neq h^{de} \bmod p$ (Because we found this number with q and c)
Thus, we know that $\sigma'^e = h^{de} + xq$
We also know that know that $H(m) = h^{de} \bmod N$
We can figure out $H(m)$ because we have m and H is public
We can also do the following $\sigma'^e - H(m)$ and we are left with xq (Because σ' is still a number mod N)
We also know that xq is a non trival multiple of q mod N because $\sigma'^e \neq h^{de} \bmod p$
Thus we can do $gcd(N, \sigma'^e - H(m))$ to factor N

Answer 2

Suppose we have a function f that given a message $m \in M$ s.t. $|M| = 20$ will output 3 distinct numbers in $\{0, 1, 2, 3, 4, 5\}$. This function is publicly known. We can now use this to extend Lamport's signing algorithm to all 20 messages. Because there are 6 keys in SK, and we are choosing 3 of them, we have 6 choose 3, or 20, possible combination that we can choose values from SK.

The signer generates SK and PK just as in Lamports using a OWF. The signer then makes PK publicly available (PK is index 0-5). The, given a message m, the signer runs $f(m) = (i_0, i_1, i_2)$
The signer then sends the message m along with SK_{i_0,i_1,i_2}
To verify, the recipient must run $f(m) = j_0, j_1, j_2$ and very that $OWF(SK_{i_0}) = PK_{j_0}$ and repeat this for the other two positions
If each succeeds, then the message is valid.
The signature cannot be forged because f only reveals the positions, not the actual SK values. And because this is one time, a forger cannot use prior knowledge of the SK to forge a new signature.
Finally, forgery depends on the one way function.
The only knowledge the forger has is f which gives positions, and PK. He cannot modify either of these are they are publicly known. Also, because each position is unique and each combination is unique, the forger cannot use the SK of a message to forge the SK of another message because he will always be missing one value.

Thus, he must reverse the OWF on the public key to forge a message

Answer 3

Given n values and a hash function H:
Start at x_1. Compute $H(x_i, x_{i+1})$ and store in the parent
If the sibling of the parent has no value, compute and store in the sibling
When the sibling has a hash value, take $H(self, sibling)$ and store it in the parent. Forget the hash of self and sibling and move to the parent
Continue doing the above steps until you have reached the root. At every parent, you only have to store the value of self + the max number of values stored at one time to compute the sibling
Thus, at each level, you store 1 + the max number of values stored at one time to compute the sibling
Thus, you only have to store hash values = the depth of the tree (If you can instantly forget two hash values once the parent is computed)
The depth of the tree is $logn$
You only have to store at most $\lceil logn \rceil$

References

None