Brandon Palonis

Dr. Garrison

Privacy in Electronic Society

2/21/2019

Project 1: Writeup

* W0: The Problem P that my program seeks to solve is that modular exponentiation. Given some byte arrays x, y, and m, it performs xy mod m and outputs some byte array z that is a result of the modular exponentiation of x,y and m.
* W1: Modular exponentiation is used in cryptography for key decryption and encryption. These keys are a way for a user of a system to confirm their identity by allowing them to sign digital signatures, as a user’s private and public keys should grant them and only them this ability.
* W4: To test Algorithm A I had it iterate over user submitted inputs of varying length and reported the average time that the modexp took during each loop. The time below is recorded in seconds. I believe the times varied so much because each large number that used modexp was randomly generated. As the function execution climbed in times, so did the bit length of the numbers. However, the fastest runtimes did not come from the smallest bit length. A 512-bit integer that was initialized to all 0’s ran the absolute fastest out of all the integers, with 256-bits not even in a close second, as the 256-bit numbers averaged ~0.01 seconds whereas the 512-bit with all 0’s had an average runtime of ~2.4E-7 seconds. This is because in my modular exponentiation algorithm, the loop that does the modular exponentiation is predicated on the fact that the private key value is not 0 (Line 94). The private key then shifts right, and attempts the same loop. Because of this, 0’s are done very quickly in my algorithm, which could leak key timing info, as 0’s are done rapidly, and when a 512-bit value initialized to be all 1’s was sent into the modexp function, it ran the second-slowest of all private keys, second to the 1024-bit key. This shows key information to an attacker, as 0’s cost almost no power to compute and 1’s cost significantly more. A simple power analysis could see the spikes generated by calculating 1’s.