Constant Time Encryption as a Countermeasure Against Remote Cache Timing Attack

D. Jayasinghe, J. Fernando, R. Ragel and D. Elkaduwe

Department of Computer Engineering, Faculty of Engineering, University of Peradeniya

Rijndael was standardised in 2001 by the National Institute of Standards and Technology (NIST) as the Advanced Encryption Standard (AES). Therefore, AES is still being used as an encryption standard to store and transmit confidential data from financial, military and government institutions. Before its inception, AES was widely analysed for prevailing side channels and other vulnerabilities for a five year duration and was declared to be the encryption standard for the next 20 or more years. However, in year 2005, Daniel Bernstein illustrated a remote cache timing attack on AES using a client and a server. Therefore, he showed a side channel in the software implementation of the AES. He used client server architecture to demonstrate the attack. The server is fed with known and unknown keys and measures the timing details. Then by comparing timing differences for known and unknown secret keys, the secret key can be deduced and confidential data can be decrypted.

The vulnerability in AES exists due to the fact that there are T tables which are used to speed up the software encryption process. AES algorithm has four stages, 1. Sub Bytes, 2. Shift Rows, 3. Mix Columns, 4. Add Round Key. Since stages 1, 2 and 3 need huge computational power, those stages are pre-computed, and stored in tables called T tables. There are 4 T tables which have 256 entries, each of which takes 32 bit space per entry. During the encryption process, T tables may not fit into the cache with the encryption data, resulting in cache hits and misses. The attack is accomplished by performing several rounds of encryption and correlating the computation against the time taken to perform the encryption.

In this project, we propose a new countermeasure against Bernstein's remote cache timing attack. We have illustrated that our countermeasure is not vulnerable even to statistical analyses which perform the attack for several iterations and then uses statistics to deduce the key. In our countermeasure, the AES encryption program is carefully rescheduled such that it will take a constant time for execution, irrespective of cache hits and misses. To make timing details constant we used the pipeline depth of the targeted processor. Therefore, the limitation of our approach is that we need rescheduling before using it in another processor which has a different pipeline depth. Security measures always ingest resources or time or both. The proposed countermeasure is approximately 1.22 times slower than the unprotected AES code which is vulnerable to the remote cache timing attack. In conclusion, we demonstrate that remote cache timing attacks are not possible when our countermeasure is applied on AES implementation and the cost is approximately 22% performance degradation of the system.