Side-channels

File `crypto.py` implements a simple Feistel two rounds symmetric cipher [https://en.wikipedia.org/wiki/Feistel_cipher], where each block consists of two bytes (16 bits). Here the round function `F` us implemented via a substitution box `T`, which maps bytes to bytes.
The code uses a simulated direct-cache, which is implemented via the class `Cache`. The cache has 32 lines and 32 bytes per line (i.e. 512 bytes in total). The whole system memory is 2048 bytes.
The class `Crypto` implements two methods:

- `load_table` which fills the memory with the substitution box
- `feistel_encrypt` which performs one encryption

The implementation has a side channel, since the final state of the cache depends on the key (and encrypted message). Write a different implementation of the cipher in `sec_cryptp`. The new implementation must use the cache, but must be free of side channels. You are free to organize the substitution table in a different way, perform additional memory accesses, etc.
To test your solution execute `./test.py` or `py.test test.py`.