NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

3010 Lecture Week 12

CRYPTO ACCEPTANCE TEST

Dr Tay Kian Boon

CRYPTO ACCEPTANCE TEST

Sometimes known as Crypto Validation Tests

Crypto Acceptance Test (CAT)

- Most of us will not develop or write our own encryption or hash source codes.
- Some codes such as AES REQUIRES much expertise for best performance!
- You also got to think of how you generate keys for users
- That’s why most companies buy encryption products or download some free ones on the web
- That’s where the danger comes

KEY QUESTIONS: Vendor *Integrity, Capability*

- How do you know they are using, say AES, and for hash, some good solid hash function?
- How do you know if keys used are randomly generated?
- They may have a beautiful brochure abt the encryption workflow, but how do you know the program works accordingly to their brochures?
- Are keys stored somewhere?
- Are parts of keys leaked out in the traffic? Malicious vendors abound

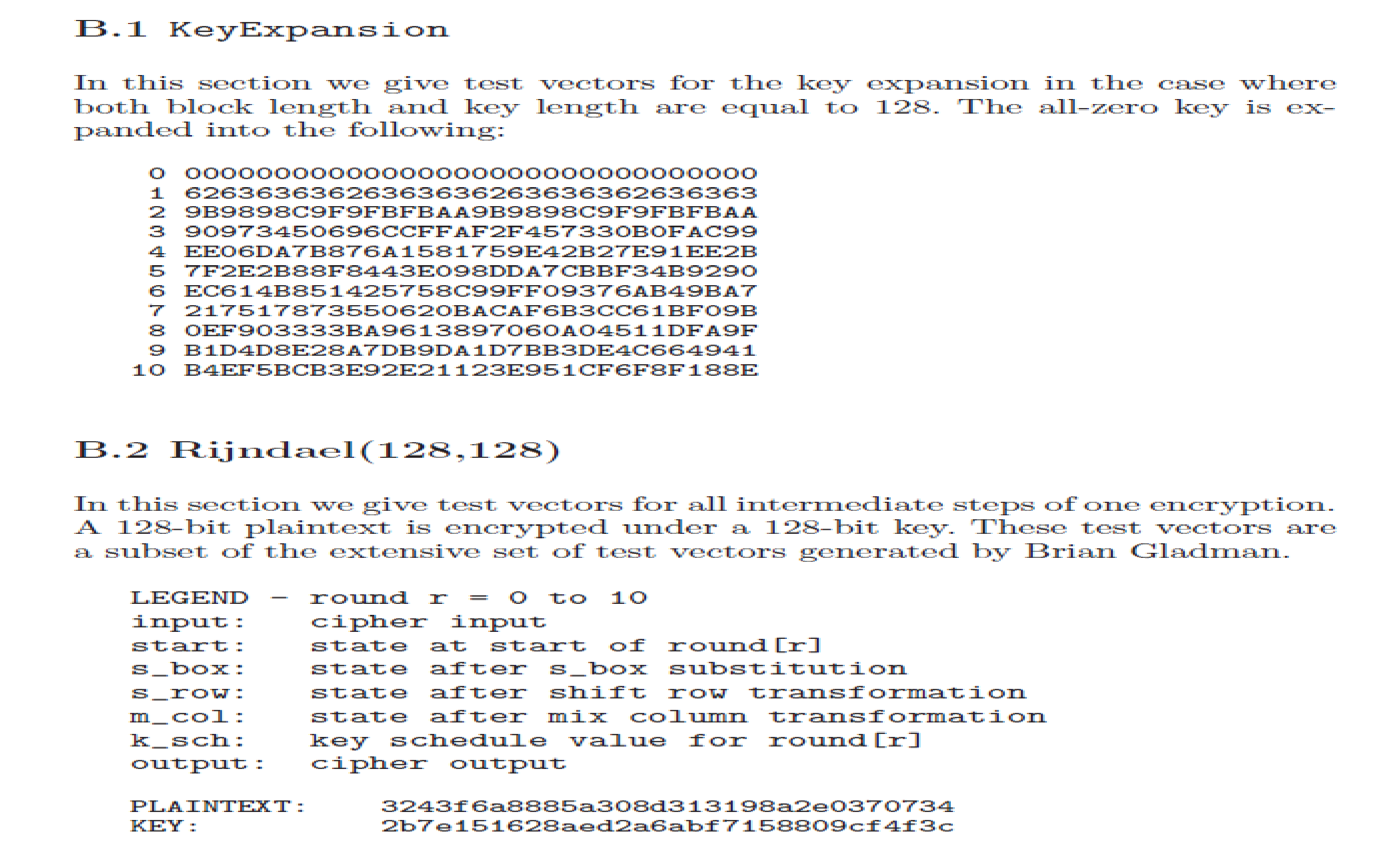
What Can Go Wrong

1. Mistakes in implementing algorithms
2. Does the key generation program reaches full entropy, e.g. can the program generates close to all possible 2^128 keys for AES?
3. Any weak implementations or practices?

Examples – some suggestions

- If program says they implement AES128, you got to verify it is true.
- If src available:
  - Check source code and compiled it and see if exe is the same for the ones you are sold and the ones you are testing
- If src not available:
  - Re EXE if you have expertise, else
  - Run the encryption and check for test vectors-plaintext (go to official AES page or book and see list of test vectors – given certain input, list will tell you what outputs are expected)
  - You might even want to check vectors not on the official list (why? Tutorial)

Test Vectors: from Design of Rijndael book



B.1 KeyExpansion

In this section we give test vectors for the key expansion in the case where both block length and key length are equal to 128. The all-zero key is expanded into the following:

```
 0 00000000000000000000000000000000
 1 62636363626363636263636362636363
 2 9B9898C9F9FBFBAA9B9898C9F9FBFBAA
 3 90973450696CCFFAF2F457330B0FAC99
 4 EE06DA7B876A1581759E42B27E91EE2B
 5 7F2E2B88F8443E098DDA7CBBF34B9290
 6 EC614B851425758C99FF09376AB49BA7
 7 217517873550620BACAF6B3CC61BF09B
 8 0EF903333BA9613897060A04511DFA9F
 9 B1D4D8E28A7DB9DA1D7BB3DE4C664941
10 B4EF5BCB3E92E21123E951CF6F8F188E
```

B.2 Rijndael(128,128)

In this section we give test vectors for all intermediate steps of one encryption. A 128-bit plaintext is encrypted under a 128-bit key. These test vectors are a subset of the extensive set of test vectors generated by Brian Gladman.

```
LEGEND - round r = 0 to 10
input:    cipher input
start:    state at start of round[r]
s_box:    state after s_box substitution
s_row:    state after shift row transformation
m_col:    state after mix column transformation
k_sch:    key schedule value for round[r]
output:   cipher output

PLAINTEXT:     3243f6a8885a308d313198a2e0370734
KEY:           2b7e151628aed2a6abf7158809cf4f3c
```

Test Vectors: from Design of Rijndael book

Examples – some suggestions

- If program says they hash with say KECCAK your password input to get the key, you must verify
  - KECCAK is really being used to hash password
  - Make sure your password is correctly hashed & not truncated (WHY?)
- If keys are generated by RNGs,
  - inquire which ones,
  - see the codes
  - Test the codes by outputting list of random numbers
  - Can verify if they pass NIST randomness test (good ones will pass)-pass does not mean 100% good unfortunately
  - Test if any key bits have been hardcoded (how to verify –Tutorial)
  - If in doubt, use your own

Examples – some suggestions

- Often times the easiest way out is to replace part of their encryption modules with your tested ones, such as crypto secure RNG
- Many many other scenarios…

Examine these uses of Hash

- Many applications such as pdf & Office use password encryption to protect files.
- Where is the key? Did user input hex?
- NO! User uses password, and application just hash them into key to be used in AES or other strong cipher.!
    - How long shud pswds be (95 printable chars) to achieve 2^128 complexity?
- Is this system good and secure?
- Many many other scenarios… (see tutorial)