Report 2

Improvements:

1. in place encrypt/decrypt
    - Encryption and decryption of data was performed with usage of two buffers (source & destination) and then copied from destination buffer back to source buffer
    - As it is written in documentation, *javacardx.crypto.Cipher.doFinal(...)* takes output buffer that can be the same as the input buffer
    - We changed this on multiple places and it improves the performance (there is no useless copy) and also makes the code readable and shorter while preserving the same functionality
2. fill transient memory with random data instead of 0s
    - Whenever the card used transient memory, at the end of instruction it filled used memory with 0s
    - We changed this behaviour to fill with randomly generated data as it is considered better practice because it does not separate used and not used transient memory blocks (blocks of 0s could help with memory analysis). Also if there was a bug using parts of not properly initialized memory, it would be definitely better to have values (e.g. keys) initialized with random bytes than with 0s.
3. double checks on critical condition
    - fault induction protection
    - attacker might be able to change the value of variable in condition evaluation so we added second check to make fault induction atack harder for the attacker
4. execution time randomization on crpyto-related parts of the code
    - we added function that creates random delay
    - this function is called before any sensitive operation to prevent time analysis and related fault induction analysis (attacker can not be sure when exactly he should fault-induce the chip)
    - our implementation is naive (empty cycle from 0 up to random number) and mostly serves as a demonstration of functionality, however it is easy to change only the body of this function (we have determined the sensitive places in the code)

5. tests
    - there were no tests in the original project
    - we added card-simulator support (from seminars) and implemented tests that will prove that future changes are not breaking current functionality
    - tests we have implemented:
        - verify pin
        - setting and changing of admin and user PIN
        - put data (set AES key)
        - encrypt/decrypt using AES
        - set RSA attributes and generate keys
6. POSSIBLE improvement - ALG_TRNG instead of ALG_SECURE_RANDOM
    - currently the project uses JC 3.0.4 kit and it supports also version 3.0.5
    - since version 3.0.5 algorithm for generating random data (ALG_SECURE_RANDOM) is considered deprecated and new algorithm (ALG_TRNG) is introduced as its successor
    - we could change this algorithm and related functions to newer version but it would make the project not compatible with 3.0.4 version and we would like to avoid that

Card compatibility problems and limits

- card does not support 3.0.4 version
- older version (3.0.3) runs on card but source code is not compatible (it uses 3.0.4's function *signPreComputedHash(...)* that is part of sign process
- as a result we were not able to measure signing performance

Performance testing

Instruction	Time [ms]
0xCA - Get data	16,5
0xCC - Get next data	11,7
0x20 - Verify	66,7
0x24 - Change reference data	75,4
0xDA - Put data da	66,2
0x47 - Generate asymmetric key pair (2048 b)	19649
0x2A - Perform security operation	
- Encrypt	16,7
- Decrypt	16,9
- Sign	NA
0x88 - Internal authenticate	
0x84 - Get challenge	19,1
0xE6 - Terminate df	19,8
0x44 - Activate file	189