Securing data and binding executable to platform using TPM chip

The TPM chip is attached to the LPC bus of the CPU. The Linux kernel drivers for TPM chips from known manufacturers like Atmel, Infenion and others. The TSS(Trusted Software Stack) can be used to write applications to use the TPM chip present in the platform.
This application has the following goals:

Encryption

1) Compress the build folder into a .zip file
2) Randomly generate a password for the blowfish encryption of the file
3) Encrypt the .zip file with the randomly generated blowfish password
4) Obtain a public key from the TPM module and encrypt the blowfish password with the key.

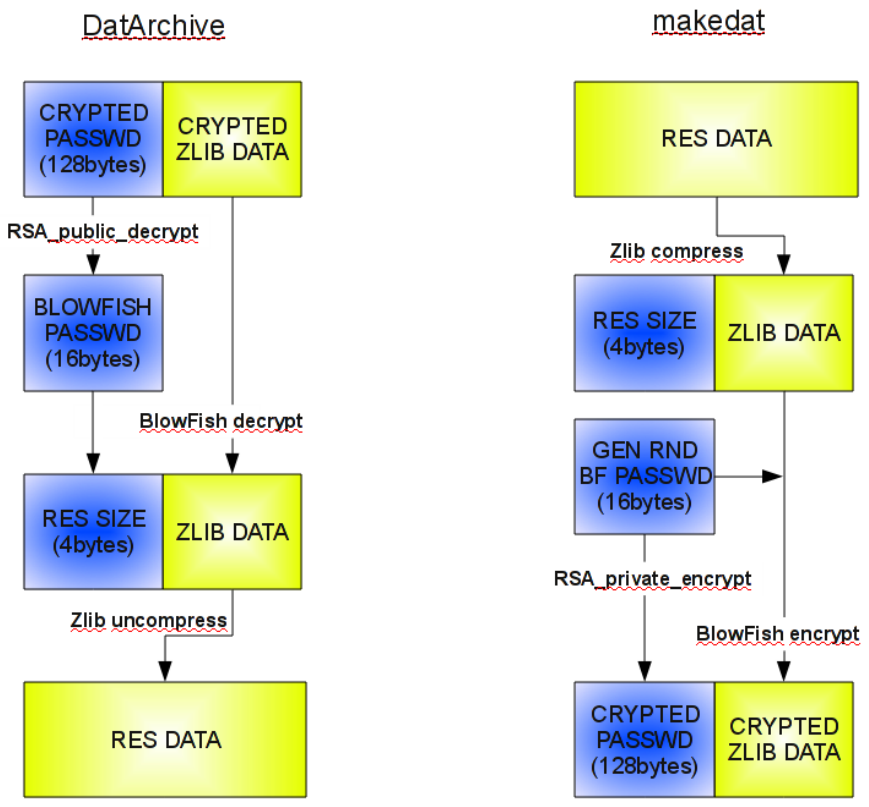
Decryption

1) Decrypt the blowfish password using the TPM module
2) Decrypt the .zip file using the decrypted blowfish password
3) Extract the .zip file to get the original build folder

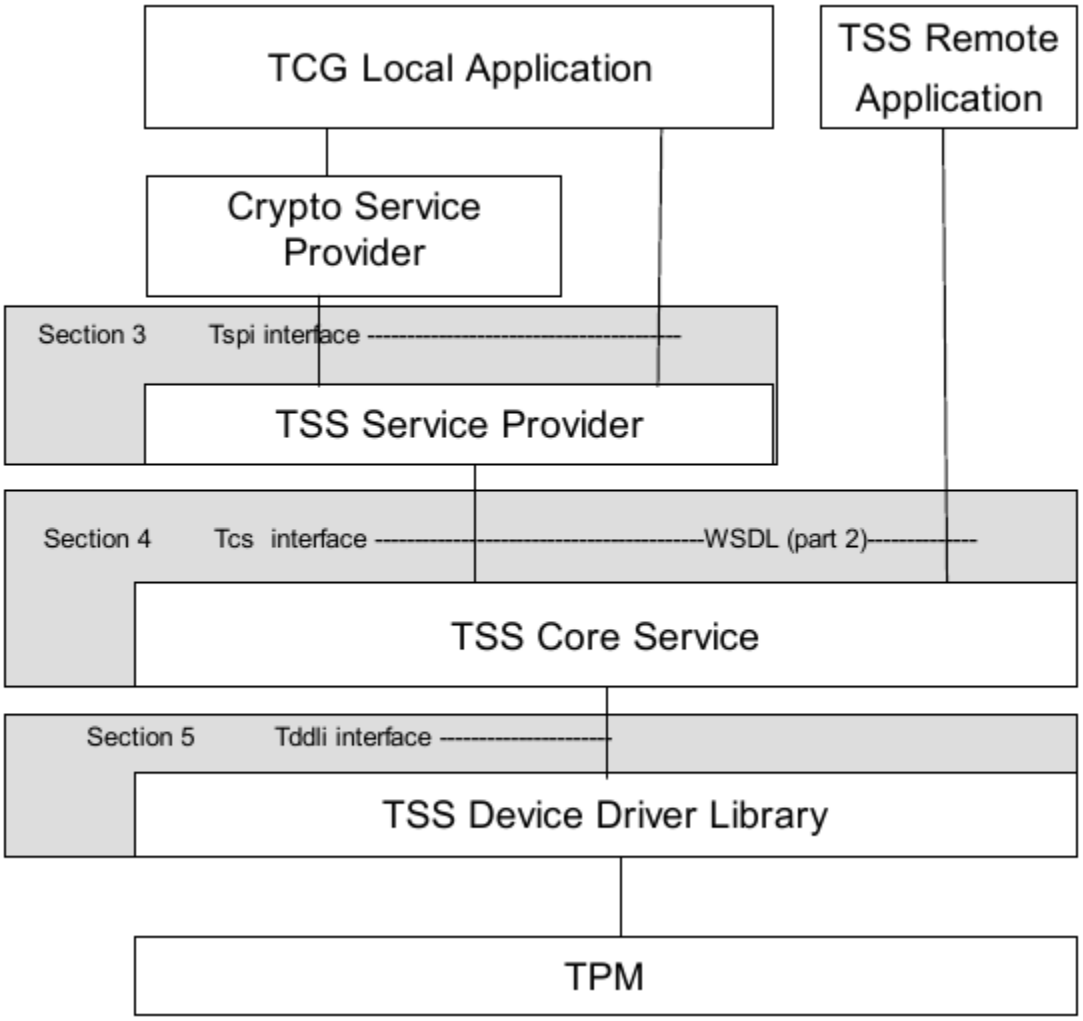
Points of security

1. The data encrypted with the public key of the TPM can only be decrypted using the private key of the TPM which is stored inside the TPM chip. This results in the data to be bound to that particular platform.
2. The Random number for the password for blowfish is generated in the TPM hardware giving a more secure password.

The program is written using the below diagram as the objective.



The TSS architecture is shown in the figure below. The application using TSS has to use TSPI(TSS service provider interface) which itself uses the TCS(TSS core service).



It is clear from the figure that writing an application using TSS would involve using the TSPI api to perform various functions using the on-board TPM chip. A detailed specification of the api can be obtained from the TSS specification.
http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification

The blowfish encryption is done using openssl encryption library.

How to use
[The compilation of the code requires the open ssl and trousers installed before hand]
For encryption program:
files requires: encrypt.c,blowfish.c, do_encryption.sh
compile command:
gcc encrypt.c -lcrypto -ltspi -o encrypt

run: Place the do_encryption.sh and the executable generated after compiling i.e. “encrypt” in the same directory. Put all the data to be encrypted in a folder named “out”. Run the do_encryption.sh

script.

This will generate a “encrypted_data.zip” this is the final encrypted file that contains the executable with the key to decrypt the executable. (Note that the key file alone is not sufficient to decrypt the executable file)

For the decryption program
files requires: decrypt.c,blowfish.c, do_decryption.sh, encrypted_data.zip
compile command:
gcc decrypt.c -lcrypto -ltspi -o decrypt
run: Place the do_decryption.sh and the executable generated after compiling i.e. “decrypt” in the same directory as the encrypted_data.zip file. Run the do_decryption.sh script.

This will generate the out folder that was initially encrypted by the encryption program.

References:
for TSS
http://www.trustedcomputinggroup.org/resources/tcg_software_stack_tss_specification
How to code your first TPM program
http://webcache.googleusercontent.com/search?q=cache:yph-iXgR02EJ:eip.epitech.eu/2011/boottruster/howto.php%3Fpart%3D2+error+Tspi_Key_CreateKey&cd=8&hl=en&ct=clnk&client=ubuntu&source=www.google.com

Tpmcreate key error
http://webcache.googleusercontent.com/search?q=cache:A8AneHHTPOoJ:comments.gmane.org/gmane.comp.encryption.trousers.user/1727+error+Tspi_Key_CreateKey&cd=4&hl=en&ct=clnk&client=ubuntu&source=www.google.com

For blowfish encryption using open ssl encryption library
http://www.faqs.org/docs/gazette/encryption.html
http://www.openssl.org/docs/crypto/blowfish.html
http://stackoverflow.com/questions/993780/assistance-with-openssl-blowfish-simple-example-inserting-garbage-characters