



# Laboratorium 2 - Tworzenie podstawowych obiektów i konfiguracja FAPI



*To jest laboratorium ćwiczeniowe. Należy je wykonać w czasie trwania zajęć. Zadanie to nie powinno zająć więcej czasu niż czas trwania laboratorium. Jeśli zadanie zostanie zakończone wcześniej, to można kontynuować prace dotyczące poprzedniego laboratorium lub pracy semestralnej.*



Autorzy konspektu: Łukasz Cierocki, Mateusz Kłos, Jerzy Pejaś

## Metody zaliczenia:

Jako zaliczenie niniejszego laboratorium przewiduje się przygotowanie z niego sprawozdania. Terminem dostarczenia sprawozdania jest przesłanie go do kolejnych zajęć laboratoryjnych.

## Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z procedurowymi inicjowaniem metod FAPI, oraz tworzenia podstawowych obiektów przy wykorzystaniu TPM.



Wszystkie operacje niżej przedstawione muszą być wykonywane na koncie z prawami administratora - root.



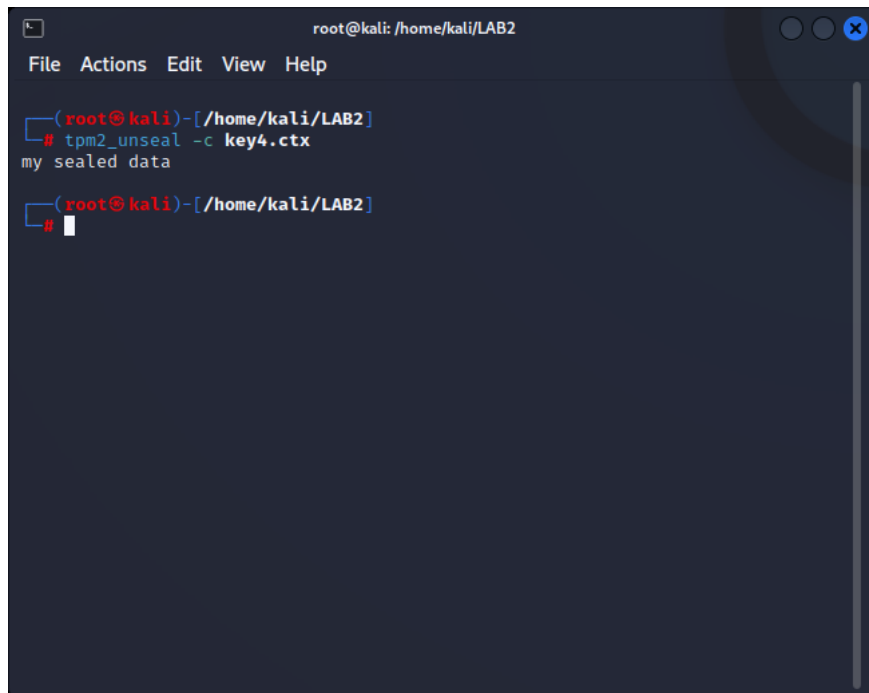
Pamiętać należy również o wcześniejszym uruchomieniu emulatora oraz narzędzia

## Tworzenie nowych obiektów

Przed wykonaniem poniższych operacji proszę zapoznać się z opisem każdego z poleceń z wykorzystaniem man oraz przygotować, krótki opis przydatny podczas tworzenia sprawozdania.

```
tpm2_createprimary -c primary.ctx
tpm2_create -C primary.ctx -G rsa2048 -u key.pub -r key.priv
tpm2_create -C primary.ctx -G rsa2048 -u key2.pub -r key2.priv -a "restricted|decrypt|fixedtpm|fixedparent|sensitive|dataorigin|userwit
tpm2_load -C primary.ctx -u key2.pub -r key2.priv -c key2.ctx
tpm2_create -C key2.ctx -G Gaes -u key3.pub -r key3.priv
echo "my sealed data" | tpm2_create -C key2.ctx -i- -u key4.pub -r key4.priv
tpm2_load -C key2.ctx -u key4.pub -r key4.priv -c key4.ctx
tpm2_unseal -c key4.ctx
```

Wynikiem ostatniej operacji, powinno być:

A terminal window titled 'root@kali: /home/kali/LAB2' with a menu bar (File, Actions, Edit, View, Help). The prompt is '(root@kali)-[/home/kali/LAB2]'. The command '# tpm2\_unseal -c key4.ctx' is entered, followed by the output 'my sealed data'. The prompt returns to '(root@kali)-[/home/kali/LAB2]' with a cursor on the next line.

```
root@kali: /home/kali/LAB2
File Actions Edit View Help
(root@kali)-[/home/kali/LAB2]
# tpm2_unseal -c key4.ctx
my sealed data
(root@kali)-[/home/kali/LAB2]
#
```

## Inicjalizacja FAPI

Aby przetestować inicjalizację FAPI, konieczne jest stworzenie programu w języku C. Na początku programu zostaje wywołana komenda:

```
Fapi_Initialize(&fapi_context, NULL);
```

Otrzymujemy po niej kontekst, którym będziemy się posługiwać w ramach tego uruchomienia. Po zakończeniu pracy z tym kontekstem należy go zakończyć za pomocą:

```
Fapi_Finalize(&fapi_context);
```

Po otrzymaniu kontekstu - przed skorzystaniem z klucza należy utworzyć instancję FAPI i powiązanego z nim TPM za pomocą polecenia:

```
Fapi_Provision(fapi_context, NULL, NULL, NULL);
```

Po poprawnym utworzeniu instancji można korzystać z potrzebnych operacji - czyli w naszym przypadku:

```
Fapi_GetRandom(fapi_context, bytesRequested, &randomBytes);
```

## Kod programu - fapi\_provision.c

```
#include <stdio.h>
#include <tss2/tss2_fapi.h>

int main()
{
    int ret;
    TSS2_RC rc;
    FAPI_CONTEXT* fapi_context = NULL;

    rc = Fapi_Initialize(&fapi_context, NULL);
```

```

if (TSS2_RC_SUCCESS != rc)
{
    printf("Fapi_Initialize FAILED! Response Code : 0x%x\n", rc);
    ret = 1;
}
else
{
    printf("Context: %p\n", fapi_context);

    printf("... fapi provision\n");
    rc = Fapi_Provision(
        fapi_context, // FAPI_CONTEXT *context,
        NULL,         // char const *authValueEh,
        NULL,         // char const *authValueSh,
        NULL          //char const *authValueLockout
    );

    if (TSS2_RC_SUCCESS != rc)
        printf("Fapi_Provision FAILED! Response Code : 0x%x\n", rc);

    Fapi_Finalize(&fapi_context);
    ret = 0;
}
return ret;
}

```

## Kod programu - fapi\_random.c

```

#include <stdio.h>
#include <tss2/tss2_fapi.h>

int main()
{
    int ret;
    TSS2_RC rc;
    FAPI_CONTEXT* fapi_context = NULL;
    size_t bytesRequested = 32;
    uint8_t* randomBytes = NULL;

    rc = Fapi_Initialize(&fapi_context, NULL);

    if (TSS2_RC_SUCCESS != rc)
    {
        printf("Fapi_Initialize FAILED! Response Code : 0x%x\n", rc);
        ret = 1;
    }
    else
    {
        printf("Context: %p\n", fapi_context);

        printf("... fapi get random\n");
        rc = Fapi_GetRandom(fapi_context, bytesRequested, &randomBytes);
        if (TSS2_RC_SUCCESS != rc)
        {
            if (TSS2_FAPI_RC_TRY_AGAIN == rc)
                printf("...\n");

            printf("Fapi_GetRandom FAILED! Response Code : 0x%x\n", rc);
            ret = 2;
        }
        else
        {
            printf("%p: ", randomBytes);
            for (int i = 0; i < bytesRequested; i++)
                printf("%#x ", (int)randomBytes[i]);
            printf("\n");

            Fapi_Free(randomBytes);
        }

        Fapi_Finalize(&fapi_context);
        ret = 0;
    }

    return ret;
}

```

W celu skompilowania powyższych programów konieczne jest przygotowanie pliku makefile, którego kod jest poniżej:



UWAGA! WCIĘCIA MUSZĄ BYĆ TABULATORAMI!

```
CC = gcc
FLAGS = -g -Wall -O3
DIRECTORIES = -L /usr/local/lib
LIBRARIES = -l tss2-fapi

OUTPUT = *.out

.PHONY: build clean

build: random.out provision.out

random.out: fapi_random.c
    $(CC) $^ $(FLAGS) $(DIRECTORIES) $(LIBRARIES) -o $@

provision.out: fapi_provision.c
    $(CC) $^ $(FLAGS) $(DIRECTORIES) $(LIBRARIES) -o $@

clean:
    rm -f $(OUTPUT)
```



PROSZĘ TYLKO SKOMPILOWAĆ! NIE URUCHAMIAĆ POWYŻSZYCH PROGRAMÓW - TO BĘDZIE ELEMENT KOLEJNEGO LABORATORIUM!