**TPM Network Gateway Workshop:**

In our project we create a secured TPM authenticated connection between clients and a network gateway. This project can be easily integrated into the university network and grant or deny access to the internet.

**OpenVPN:**

The connection between the client and the server will be created by OpenSSL. The OpenSSL will create a secured tunnel between two endpoints using an IPSec like protocol. During the SSL connection establishment we make a key exchange process similar to IKE process. Afterwards we use the IPSec ESP protocol for tunnel packet security (but on transport layer instead of network layer). OpenSSL is used in our project to make it easily extended and maintained, changes can be performed by editing configuration file only without having to modified and recompile the project and more important OpenVPN is a user space applications (and not kernel mode), which leverage better system security for variety of operating system (not kernel dependent anymore).

The OpenVPN tunnel consists of two channels:

a. the control channel – used for key exchange – fully encrypted by TLS.

b. the data channel – used for data transfer – signed by HMAC and may be encrypted by TLS using the control channel keys.
We extended the OpenVPN protocol so it will not only use an ordinary user password for authentication but will also send another challenge response can be satisfied only by a previously registered TPM device.

**The Challenge - Response protocol:**

1. The client initiates a connection and sends its ID.
2. The server creates a challenge and sends it back to the client.
3. The client receives the message and does the following:
   a. Hash the challenge using SHA1.
   b. Signs the hashed challenge using the AIK private key with the tspi_tpm_quote API.
   c. Change the binary code to base64.
   d. Sends the result back to the server.
4. The server receives the response and does the following:
   a. Return the response to binary from base64.
   b. Hash the original challenge using SHA1.
   c. Check the signature using the registered public key that matches the client ID.
   d. Deny/Grant connection to the client.
Request to connect
Including ID
Randomize challenge
Send challenge
Sign challenge
SHA1
Hash the challenge to 20bit
Receive challenge
Sign challenge
Binary to Base64
Send Response
Receive Response
Base64 to Binary
Search ID and retrieve matching public AIK
Check Signature
Decide to Allow/Deny
SERVER
CLIENT
Search ID and retrieve matching public AIK
Randomize challenge
Send challenge
Sign challenge
SHA1
Hash the challenge to 20bit
Receive challenge
Sign challenge
Binary to Base64
Send Response
Receive Response
Base64 to Binary
Search ID and retrieve matching public AIK
Check Signature
Decide to Allow/Deny
**TPM Keys role:**

The most important feature in this solution is the TPM authentication. In order to achieve this goal the TPM device will create on the client will create a public and private AIK keys. The private key will be used in the challenge – response earlier described at the client side to sign the hashed challenge. The public key will be used on the server side to validate that signature.

**TPM Keys creation and exchange:**

In order to get the AIK keys we planned to use a live-cd (will be introduced below) to create a clean environment, without any unknown programs or kernel modules, and run a script that access the TPM and creates the AIK keys. In this way the public key will be saved raw, while the private key will be wrapped (encrypted by the TPMs SRK, and can be decrypted and used only by it). Both of the keys will be saved on a USB stick. When the machine will reboot without the live-cd we will copy the private key to the machine and use that key as discussed above. The public key will be copied to the server data base and create an ID for this client.

After creating the live-cd and script we discovered that the TPM device has a protection against foreign Operating system and therefore does not respond to key creation commands. Instead of using the live-cd a client can either run the script on his computer operating system (in this way we are expose to the threats listed above) or use the privacy CA as described below.

**Client ID:**

After the server has obtained an ID for the client, the ID can be sent to the client through any media available, like email, DOK etc.

**Root of trust:**

When working in a Trusted Computer Group the platform level of trustworthiness and platform characteristics can be described in three different Roots of Trusts:

1. RTM: Root of Trust for Measurement
2. RTS: Root of Trust for Storage
3. RTR: Root of Trust for Reporting

Concerning RTR, this is a piece of code capable of vouching for the authenticity of PCR values (based on trusted platform identity, using AIK). The integrity measurements are digitally signed to authenticate PCR values.
In our solution each time a client connects to the network gateway we are adding a random challenge to the PCR and signing them together with the AIK private key we previously created.

**AIK (Authentication Identity Key):**

The AIK is an asymmetric key pair that can be created by the TPM. The TPM can create an unlimited number of AIKs. The AIK can be used only to sign information that was generated internally by the TPM. AIK must never sign arbitrary external data so attackers could not take advantage and create fake PCR values.

In our solution we will use the AIK capability of signing PCR values together with a randomized challenge in the authentication process.

**AIK attestation process:**

On our project we implemented one of few available approaches for attestation:

1. The implemented approach - Based on certificate authority which stated by the TCG as Privacy CA which issue the AIK credentials. The TPM create a pair of AIK asymmetric keys and send the AIK public key and the EK public key. Some TPM manufactures embed inside the TPM chip EK certificates which helps the Privacy CA validate the authenticity of the TPM which created the AIK. If the TPM has certificates the Privacy CA validate that the public EK is valid TPM key using the TPM manufacturer published certificates. If the key is valid the Privacy CA signs the AIK and send encrypting it using the public EK and send it back to the TPM client. Now only the TPM which has the valid private key can decrypt the CA signed AIK and publish the key to the server. Now the server can validate that the AIK key is genuine. The reason we make this complex process is that the EK cannot sign due to privacy concern, hence that is the way stated by the TCG to create keys without exposing the TPM identity. This approach allows us to create credential without physical presence.

2. The third approach is using direct attestation presented on privacyca.com, which do not keep the user privacy, but it require EK certificate as well. Therefore, we decided not to implement this approach.

3. The last approach is DAA (Direct Anonymous Attestation) using blind signatures, presented by IBM, which was not fully investigated by us due to limited resources.

Identity Server
Privacy CA
TPM Client
Create AIK asymmetric pair
Validate EK certificate
Sign AIK public key
Encrypt Signed AIK using public EK
AIK public, EK public + certificates
Encrypt Signed AIK using private EK
Decrypt Signed AIK using private EK
Verify PrivacyCA signature
PCA signed AIK
VPN configuration files
Configure VPN environment
**Username Password registration:**

In addition to our TPM solution, a client can register also by username and password.

The IT admin can add registries of username and password on the server, where the password will be saved hashed and moved to base64 using a script called sha1_base64.

The client will run a script called openvpn_user_pass followed by username password (example: openvpn_user_pass avicohen4 Okj4cnj#fd).

**LIVE-CD:**

Live-CD is a CD or a DVD containing bootable computer operating system.

The term "live" derives from the fact that these CDs each contain a complete, functioning and operational operating system on the distribution medium.

When running live-cd with default options, it allows the user to return the computer to its previous state when the live-cd is ejected and the computer is rebooted.

In our solution we created a live-cd that will be used when new user wants to register to the TPM service. The IT admin will reboot the client laptop from the live-cd, run our TPM script and save the AIK keys on a USB stick.

Using a live-cd will ensure a clean environment and therefore makes the TPM script safe and secure for the user laptop and to our TPM code and results.

The AIK private key is wrapped and can be opened only by the TPM so there is neither safety nor privacy problem there.

**TPM Prerequisites:**

- TPM EK and SRK keys should be protected by the well known password.
- TPM should be installed and enabled.
- TPM supports TSS Spec 1.2
Libraries we used:

- Privacy CA – remote attestation - http://www.privacyca.com
- OpenSSL – cryptographic SLL functionality - http://www.openssl.org