# VPN on NEXYS2

## EDA385

*Design of Embedded Systems:*
*Advanced Course*


Mentor:

***Flavius Gruian***
(flavius.gruian@cs.lth.se)

Project Participants:

***Dan Kvelstad***
*(dt06dk5@student.lth.se)*

***Michael Gissing***
*(int11mg3@student.lth.se)*

***Leo Bärring***
*(et06lb2@student.lth.se)*

# System Description

## The Big Picture

This is a project about developing a proof-of-concept system for establishing an encrypted tunnel over an IP network. An encrypted tunnel is a way of safely connecting two Local Area Networks via an insecure and possibly very large IP network. This will result in an entity called Virtual Private Network (VPN) in which the clients believe that there are two sub-networks connected via a single router. This is visualised in picture 1 below.

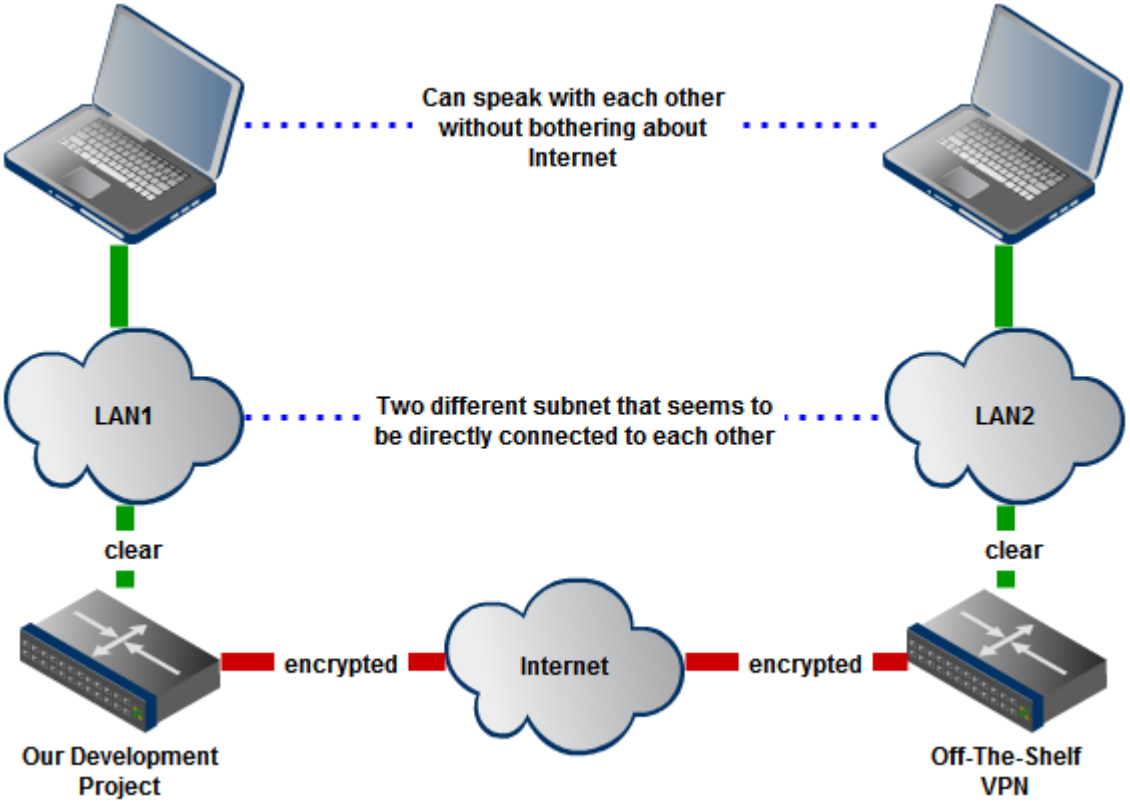


figure 1: System Description

The IP routing network, i.e. Internet, is not trusted and some protection of the tunnelled traffic between LAN1 and LAN2 is desired. This will be done according to the IPsec standard and should be compatible with of-the-shelf systems using the same standard. This process is not known by, nor cared about, the clients belonging to the separate LAN.

# Tunneling

The IP packet transmitted from a LAN transports a higher level protocol such as TCP. The actual protocol above IP is irrelevant for this project since they will all be treated as payload.

When our system receives a clear, non-encrypted, data packet it will create a new packet with a new Ethernet, IP, ESP information and authentication data. It will then encrypt the received packet and attach this encrypted packet as payload to the new packet. This principle is called encapsulation and is presented in figure 2.

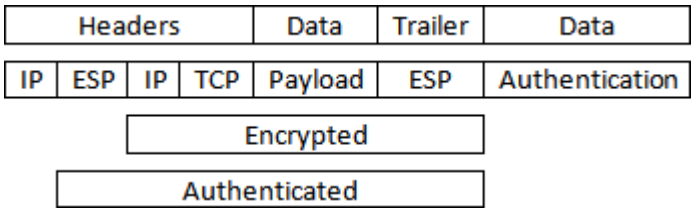


figure 2: Encapsulation

The ESP header contains information about what encryption algorithm was used to encrypt the payload and the authentication data is used to ensure that the packet has not been tampered with. The new packet is then sent over the insecure network to the other VPN server. Once the other end receives the packet it will strip away the wrapper packet and send the original IP packet into its LAN.

It is not the actual algorithm or keys that will be transferred in the ESP header. It only carries a reference to which was used. The actual means to decrypt the packet is stored locally at the receiver in a so-called a Security Association (SA) that corresponds to a SA at the senders. The receiver has a specific SA for every sender and store all of these in a local database called Security Association Database (SAD).

Security Associations store parameters like the cryptographic algorithm, mode and key. A SA is identified by the Security Parameter Index (SPI) that is part of each IPsec package. A two way communication requires at least two SA since each can only contain the parameters for one direction.

SA are designed to be configured via a protocol called Internet Key Exchange (IKE). This project will not implement this protocol due to time constraints and will setup the SA manually instead. Another restraint will be that the system only supports one bi-directional tunnel between two peers and by extension will only contain two SA. This will result in the concept of SAD being naively implemented.

The IPsec standard supports a multitude of encryption algorithms but this system will only support the encryption algorithm DES-CBC. To add other standards can be seen as trivial but time and space consuming.

Authentication will be implemented in software using the HMAC-SHA1-96 standard.

# Hardware Design

The project will be started up using a design as in the image below.

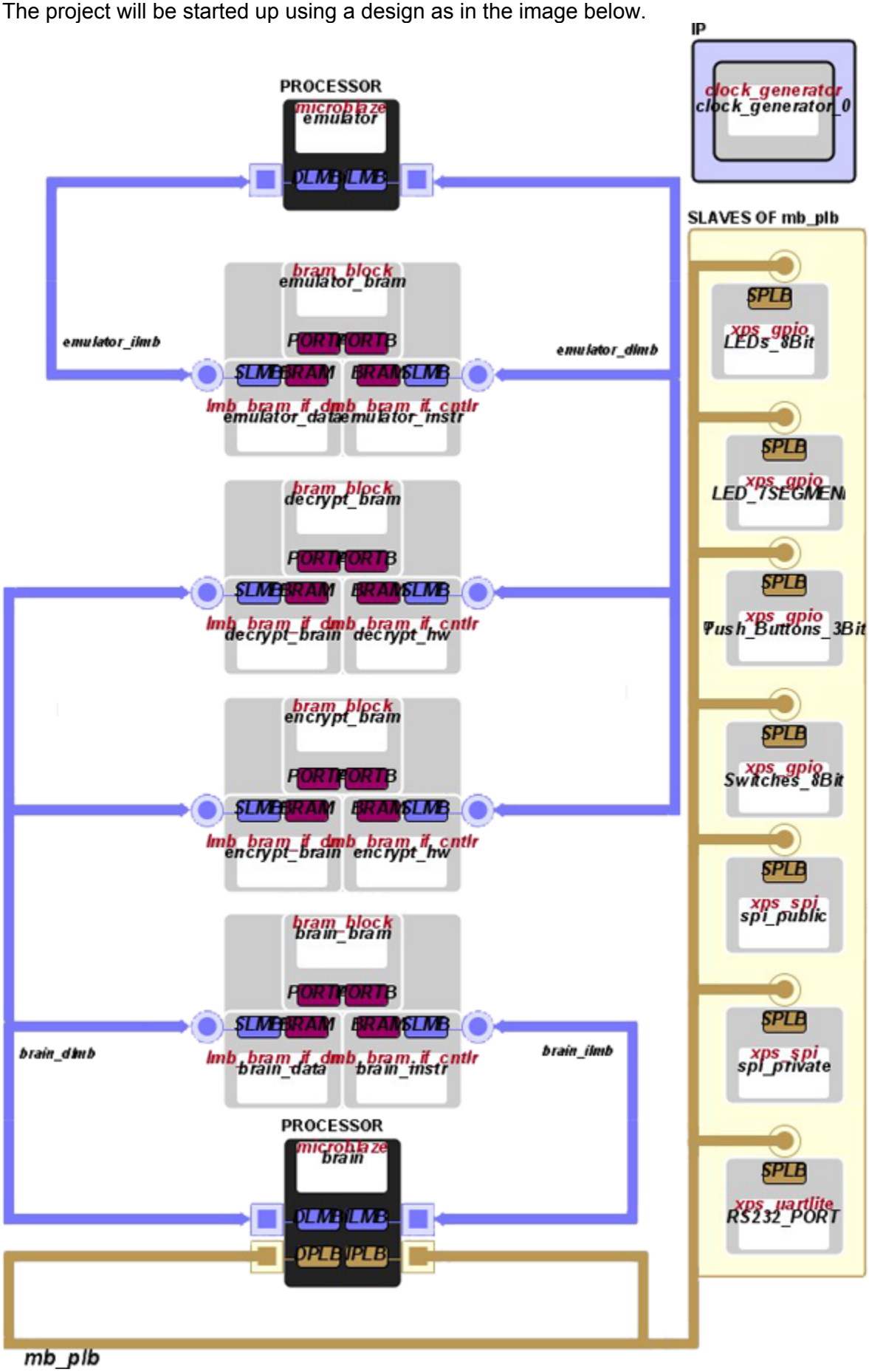


figure 3: System Design

The start-design consists of two processors. The most important one will be referred to

as *brain* and is responsible for flow control of the system. The second processing unit is entitled *emulator* and will emulate hardware accelerators until such a time that they can be implemented using VHDL. Brain should not be able to tell if the system uses hardware accelerators or the emulator core by other means than of measuring the time required for encrypting/decrypting (*crypting) packets.

The two BRAM blocks that are common between the brain and the *cryption system will be divided into three logical zones. The first being a control word that states if the memory is in use and how. This zone is the only mean of communication for the brain and the *cryption system. The second and third zone contains the incoming and the outgoing versions of a packet. The structure is the same for both BRAM. The following flowchart demonstrates this:

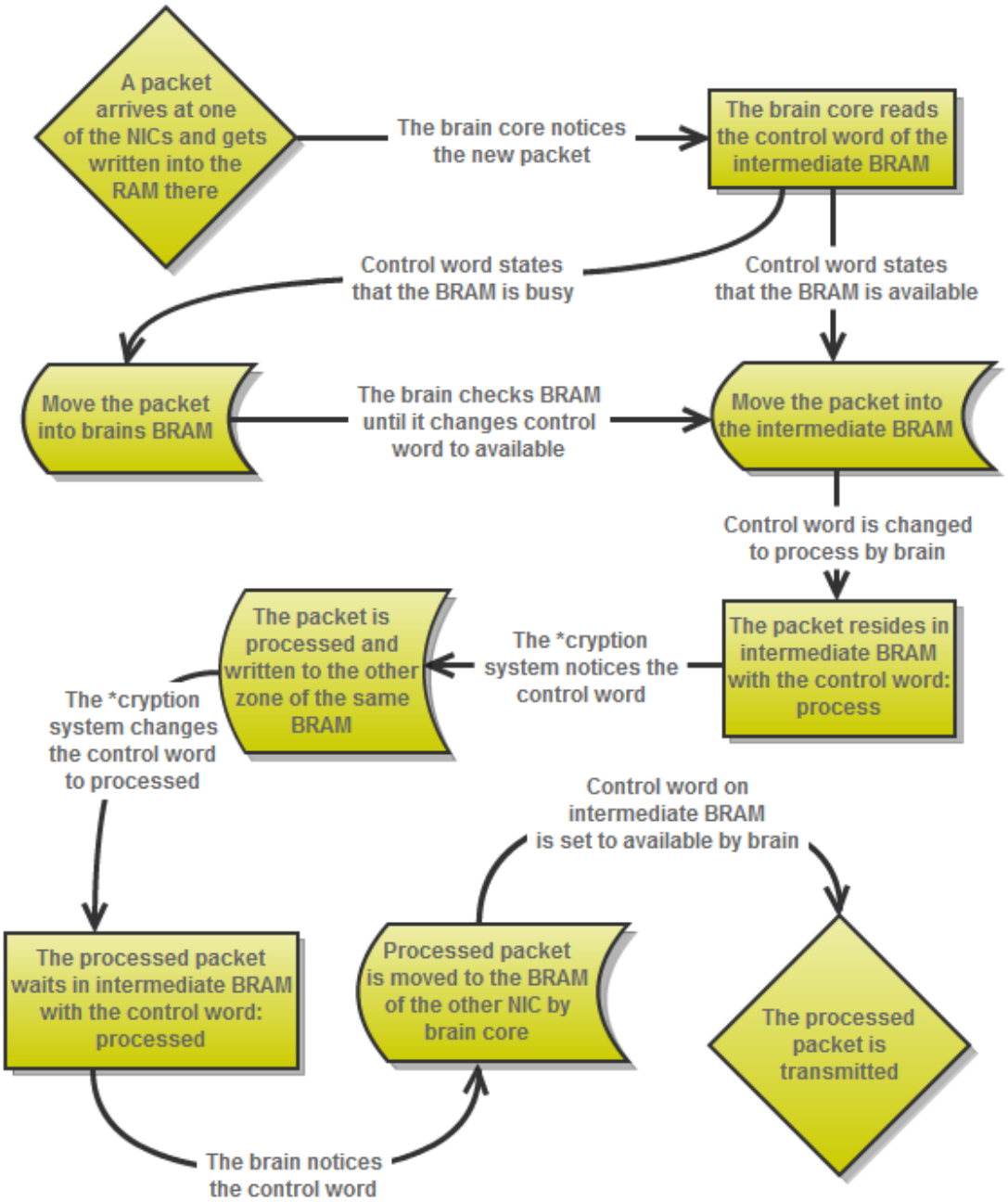


figure 4: Brain and *cryption system interaction

This design will lead to an incrementally implementable system that is also robust. It also enables an easy and fair comparison of time requirements for emulation and hardware.

# Possible Improvements

The system is a skeleton of the IPsec standard and has many areas of improvements. The most important addition would be to implement the authentication algorithm SHA1 in hardware instead of letting the brain calculating it. This is a feature that might actually be implemented, time allowing.

Another important improvement area would be the rudimentary implemented TCP/IP stack. Installation is now complicated due to static configuration and the ARP and DHCP protocols would significantly ease the installation process.

There are functional limitations to this implementation of IPsec as well. The encrypted packet might become fragmented in the large IP network. This causes problems since the whole packet is needed in order to decrypt it and send it on. IPsec contains recommended ways of dealing with this, using a sliding window, but it requires a considerable implementation effort. This system will simply drop fragmented IP packets.

The encryption could be improved by supporting more protocols such as 3DES, AES and Blowfish. More protocols and capability would cause more SA to be generated and the configuration of these should really be done via IKE. More SA also means that a system for storing them is needed and a SAD must be implemented. Further improvements could be to allow configuration by means of SD cards, USB thumbdrives or RS232 and credentials stored on smart-cards. These are all functionality that commercial systems usually contains.

This specific system could also benefit from completely removing the Brain microblaze from the IP packet data path. Allowing the system to do all *cryption autonomously. This is plausible here since the control path is simple. It should be noted that as the implemented features of IPsec increase, so does the control complexity. This increase would most likely render this optimization moot.

# Time Plan

1. **Oral Presentation** (1 day, Michael)
2. **Project Proposal** (2 days, Dan)
3. **FPGA hardware configuration** (3 days, Dan)
    a. VHDL modules are replace by emulator microblaze.
4. **Ethernet on both NIC's** (5 days, Leo)
    a. set Control Registers
        i. via SPI, i.e. Microblaze
    b. set PHY registers
        i. can only be accessed through MIIM implemented in the MAC.
    c. make sure that the Microblaze processor works with the RAM in PModNIC
        i. write data
        ii. read data
    d. working Ethernet functionality
        i. send frames
            - monitor via a network sniffer
        ii. receive frames
            - print to UART, when verified: remove this.
5. **Internet Protocol** (3 days, Dan)
    a. assign static IP address
    b. sending packets
        i. monitor via a network sniffer.
        ii. no encryption.
    c. receiving packets
        i. print to UART, when verified: remove this.
    d. on both NIC
6. **Tunnelling** (3 days, Dan)
    a. encapsulation, i.e. append second IP header.
    b. monitor connection status via UART, led or 7-segment display.
7. **Security Associations** (2 days, Michael)
    a. static setup located in software BRAM.
8. **Software Encryption** (5 days, Leo)
    a. NULL and DES encryption
    b. Encrypt the data using Microblaze
        i. Add ESP (IP protocol 50) header/trailer
    c. Decrypt the data using Microblaze
        i. Remove ESP (IP protocol 50) header/trailer
9. **Validation** (5 days, Leo)
    a. System should now be operational, but slow.
    b. Test towards of-the-shelf system.
10. **Hardware Encryption** (5 days, Michael)
    a. Implement VHDL module for encryption (NULL and DES).
    b. Implement VHDL module for decryption (NULL and DES).
11. **Validation** (5 days, Michael)
12. **Project Report** (2 days, Dan)
13. **Project Presentation** (1 day, Michael)