The LS-ARP proposal enables security of the ARP by making a verification process. The verification is carried out by the host (Alice) who wishes to communicate to another one (Bob). It consists of comparing two values and if the two values are the same then Bob is trustworthy. However, in order to make this comparison, Bob has to share certain information to Alice. The information is added within the ARP packet and transmitted through the ARP request and ARP reply. This leads not only to the modification of the ARP request and the ARP reply functions but also of the ARP packet.

At this point, it is crucial to know in detail the ARP packet structure format. According to the RFC 826, the structure of this packet is defined as follows:

16 bit: Hardware address space.

16 bit: Protocol address space.

8 bit: Byte length of each hardware address.

8 bit: Byte length of each hardware address.

16 bit: Opcode.

Variable: Hardware address of the sender of this packet.

Variable: Protocol address of the sender of this packet.

Variable: Hardware address of the target of this packet.

Variable: Protocol address of the target of this packet.

The sizes of the last four fields depend on the length of the hardware addresses in the message and the length of the protocol addresses. For Ethernet or other networks using IEEE 802 MAC addresses, the value is 6. For IPv4 addresses the value is four. As a result, the standard ARP packet requires a total of 28 bytes:

16 bit: Hardware address space.

16 bit: Protocol address space.

8 bit: Byte length of each hardware address.

8 bit: Byte length of each hardware address.

16 bit: Opcode.

48 bit: Hardware address of the sender of this packet.

32 bit: Protocol address of the sender of this packet.

48 bit: Hardware address of the target of this packet.

32 bit: Protocol address of the target of this packet.

![A description...](data:None;base64,)

Figure 1. ARP message format.

Once examined the structure of the packet we can proceed to the modification. According to the LS-ARP proposition, to make the verification process possible, Alice requires Bob´s expiration time and R-value. Therefore, the first step is the addition of two fields at the ARP packet.

Currently, for each the expiration time and the R-value a 32 bit field was added. Nevertheless, the size of these fields can be changed to a suitable size. As a result, the new ARP packet consists of 36 bytes. Naturally, this value has to be change in the header file (etharp.h) anytime the size of the any of the new fields is changed. Otherwise, because of the way ARP is implemented in Genode, the data structure (pbuf) allocated in RAM, which allows to store the ARP packet, would not allocate enough memory. Furthermore when and ARP request is received, the packet size is inspected and it has to be greater or equal than the size defined in the header file, otherwise the packet will be discarded automatically. This is implemented in the etharp\_arp\_input function of the etharp.c file.

After the modification of the ARP packet, the outgoing ARP request was modified. In Genode the function which prepares the data to be sent to the target host is called etharp\_raw.

![A description...](data:None;base64,)

Figure 2. ARP Transaction Process.

After the section where the standard fields of the APR packet are filled, the new two fields are filled. The expiration time is set to to zero and the R\_value is set to be the R-value of the source host. Right now only the R-value is used for the verification process in order to check that the transaction is completed. Nevertheless, this simple verification will have to be improved by means of a hash function which would take into account the expiration time, the ip address and the mac adddress.

Finally, there is a function called etharp\_arp\_input, which handles both the incoming ARP request (case of target host) and an incoming ARP reply (case for source host). Since this function defines both behaviours, it is suitable to make the verification process on both the target host and the source host. In other words, first Bob makes the verification process and later on Alice does the same, even though Alice was the one who started the ARP.

In the etharp\_arp\_input function, after the packet is received, first of all it is determined if the packet length is at least of the size of an ARP packet (defined in the header). If it is not the case the packet is discarded and a message is displayed stating this situation. Otherwise, the etharp\_update\_arp\_entry function is called which is in charge of inserting/updating and ARP entry in the the ARP cache table.

For this reason, it it only makes sense to do the verification process before the new ARP entry is stored in the ARP cache table. Hence, the new code regarding the verification process is defined at his point. As part of this verification process, it the R\_values are equal, the etharp\_arp\_input function continues the execution of the rest of the code. However if the two values do not match, the function terminates and the packet is dropped. Additionally a message is displayed referring to the fake identity of the host which is sent f the data with a state of severe (highest state defined in the Genode ARP implementation).

It is important to mention, that whenever a message is sent, the data is change from little-endian format (Host Byte Order) to big-endian format (Network Byte Order). So, when the r\_value from the source host is sent, it is convert to big-endian format through the macro PP\_HTONL, since our R\_values are defined of size 32 bit. And when the verification process takes place, the R\_value of the source host is changed to big\_endian, because the incoming packet is already convected to this format. Hence, we make sure that the values compared are in the same format, so the verification can be carried out correctly.