PISA UNIVERSITY

CYBERSECURITY PROJECT

REPORT

ACADEMIC YEAR 2019-2020

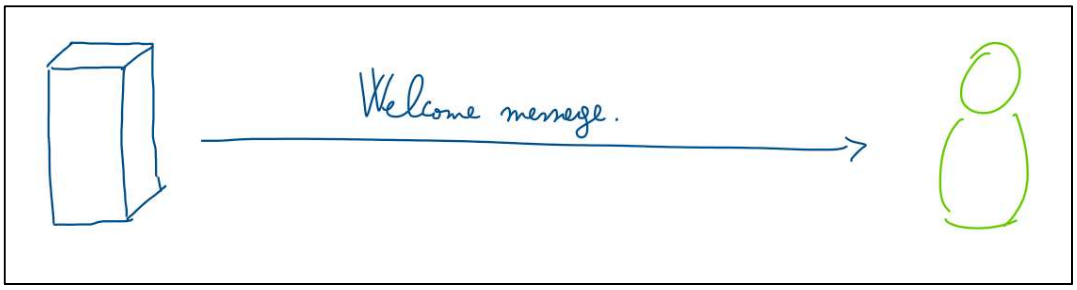
STEFANO PETROCCHI, ANDREA TUBAK

SUMMARY

MESSAGES FORMAT

START

When a new client starts the application, it tries to connect to the server. The server reacts to the new incoming connection by sending a welcome message:

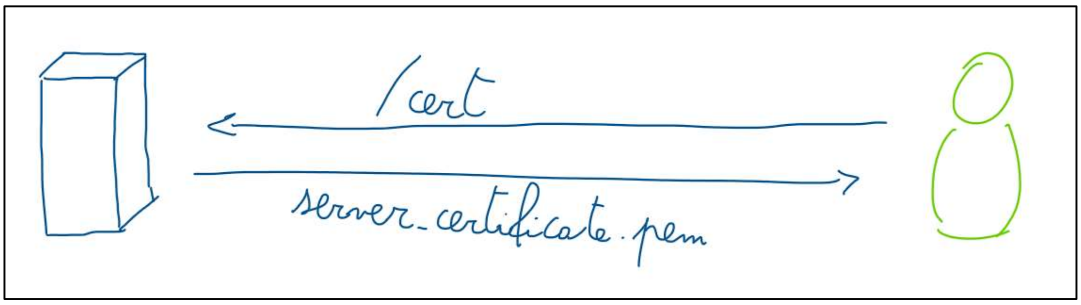


The message contains: 'Welcome to Four in a Row! \nTo receive the certificate type: \"/cert\"'

A non-logged user can type only two commands which will be sent in clear:

- /cert → to ask for the certificate of the public key of the server
- /login:username → to log in as the user called username

CERTIFICATE REQUEST



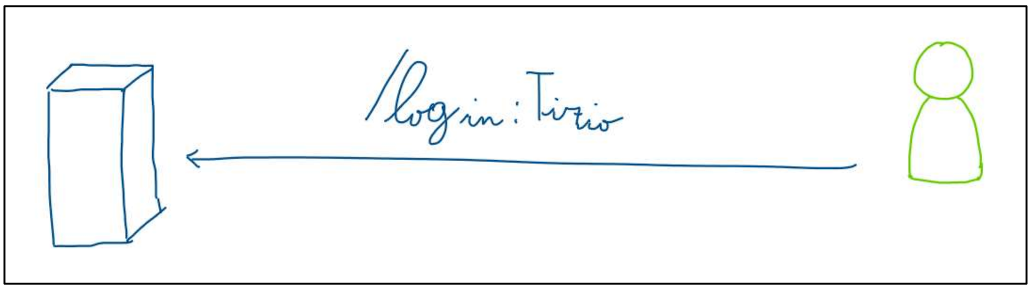
The server sends back the pem file of its certificate.

LOGIN + IDENTIFICATION

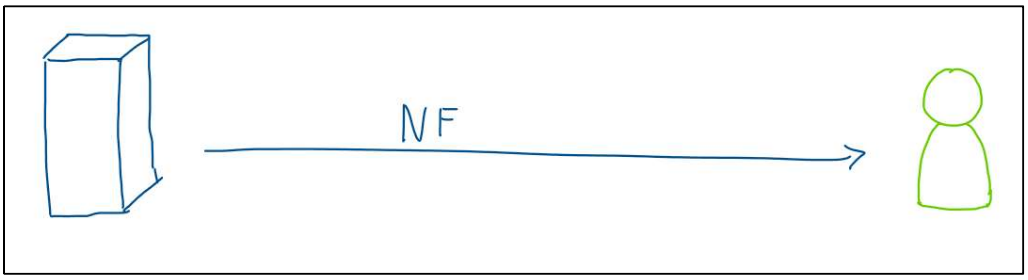
The server owns all the public keys of the users subscribed, while all users have the certificate of the public key of the server.

This is the sequence of messages sent between user and server for the log in operation. The user sends the log in command together with the username.

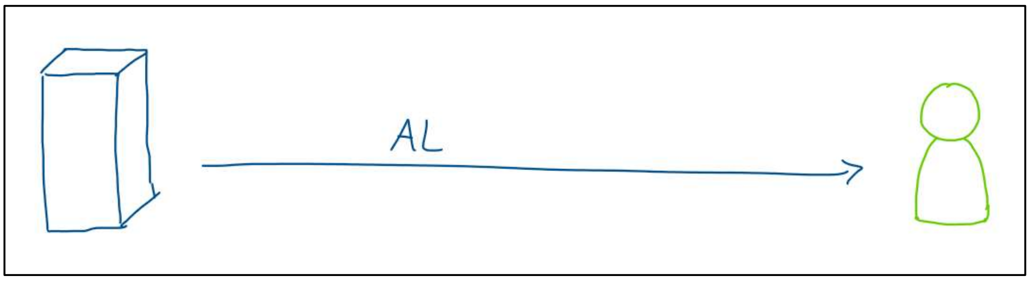
/login:username



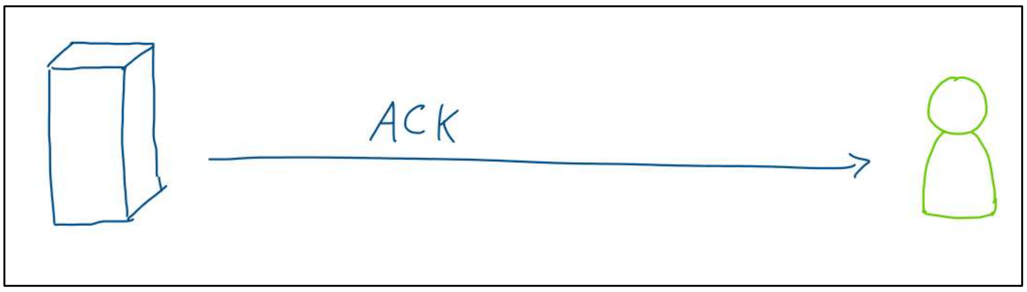
In case of user not found:



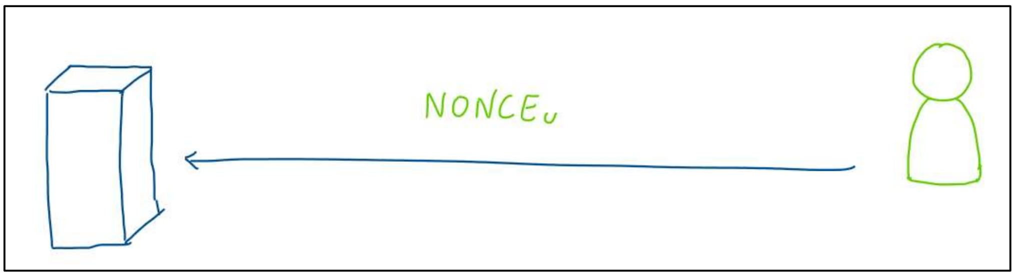
In case of user already logged:



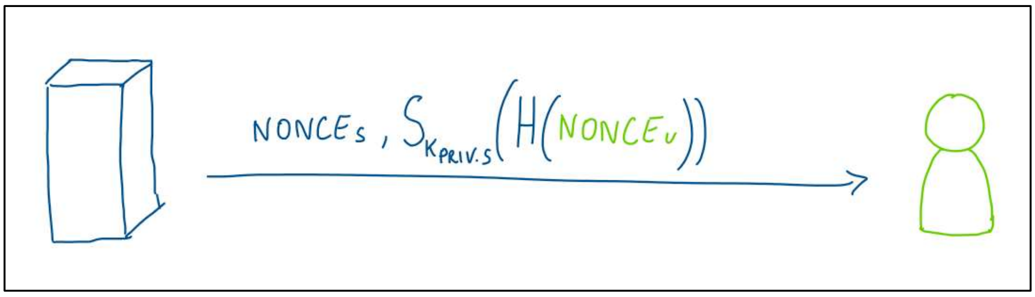
In case everything is ok:



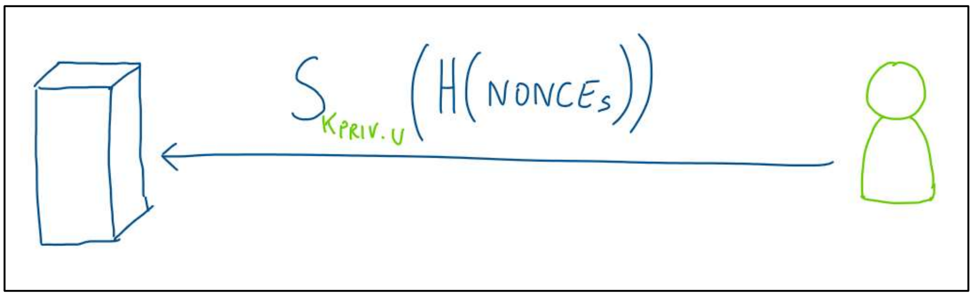
If the user receives an ACK message, he will send a randomly chosen number called NONCEu:



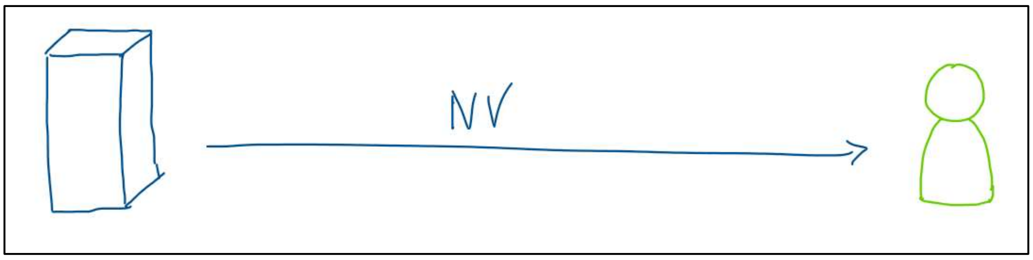
The server replies with its own nonce (called NONCEs) followed by the digital signature of NONCEu:



The user controls the validity of the digital signature and, if correct, proceeds to sign NONCEs and send it back to the server.



The server controls the signature of the user and if not correct it responds with a not valid (NV) message:

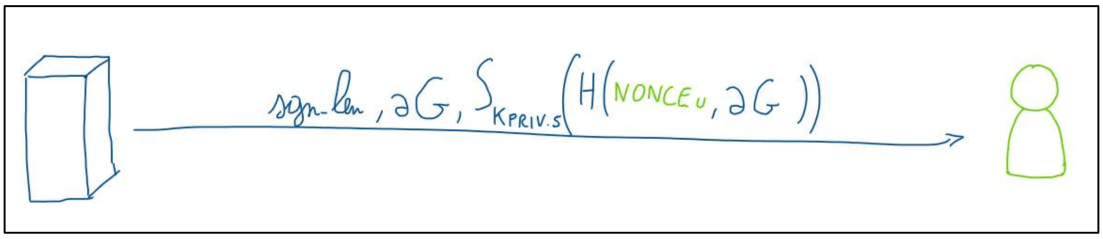


Otherwise it sends an ACK message and the authentication process is finished.

Both client and server will know that the messages sent between them are fresh, authentic and unmodified. We need also to guarantee perfect forward secrecy, so we need to establish a session key at every log in.

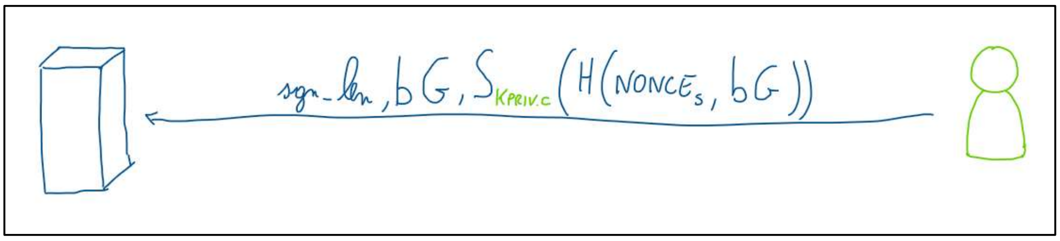
To achieve that we are going to use Elliptic curve Diffie-Hellman Key Exchange together with a nonce to guarantee the freshness.

The server generates its private and public key and it sends the following message:



Where sgn_len is the length of the digital signature, NONCEu is the same nonce generated by the user during the login process, aG is the ECDH public key and then the digital signature of NONCEu and the public key.

The user controls the validity of the digital signature, saves the ECDH public key and generates his own pair of private and public key. Then, the user sends the following message:



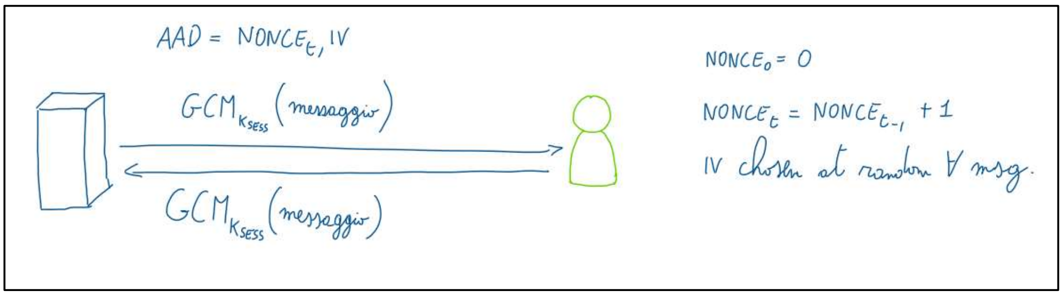
The format of this message is essentially the same as the one saw above, with the only difference being that bG is the public key of the user and the NONCE used is the server's one.

At this point, if everything went smoothly a session key can be derived:

$$\boxed{K_{SESS} = H(abG)}$$

MESSAGE EXCHANGE DURING A SESSION:

The messages sent during a session between a user and the server are encrypted and authenticated via AES-GCM. For each message sent the AAD contains a random IV and a NONCE which is incremented before every send and receive to avoid a replay of the same old message. Initially the NONCE starts from 0.

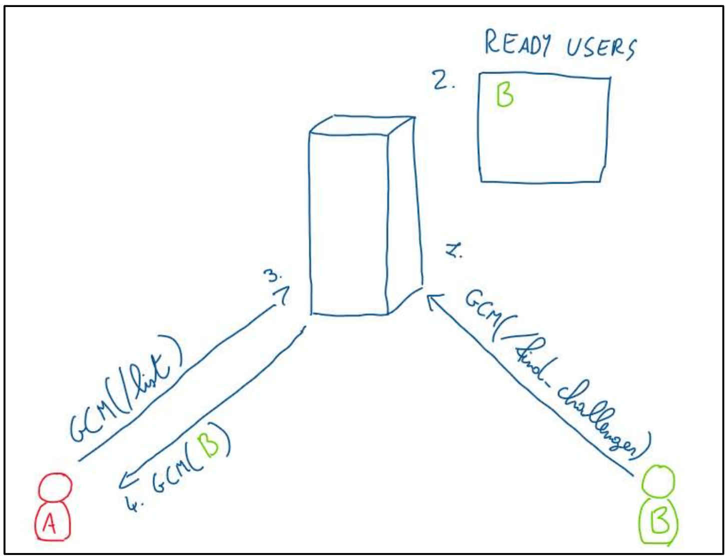

$AAD = NONCE_t, IV$
$GCM_{K_{SESS}}(messaggio)$
$GCM_{K_{SESS}}(messaggio)$
$NONCE_0 = 0$
$NONCE_t = NONCE_{t-1} + 1$
IV chosen at random ∀ msg.

All users will communicate with the server using GCM and their session key.

At this point a logged user can type other commands:

- /find_challenge → to ask the server to be inserted in the list of users READY to play;
- /list → to ask the server to return the list of users READY to be challenged;
- /challenge:user → to ask the server to send a challenge request the the user specified.

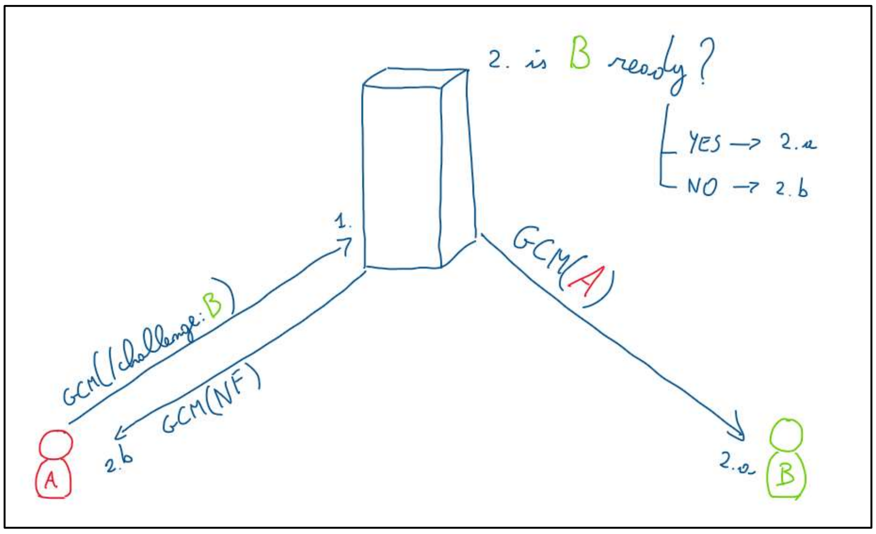
FIND CHALLENGE AND LIST COMMANDS



This image shows a possible scenario:

1) Bob wants to play and so he sends a find challenge message to the server.
2) The server inserts Bob in the list of READY users
3) Alice wants to know who is ready to be challenged and sends a list command
4) The server returns the list of READY users

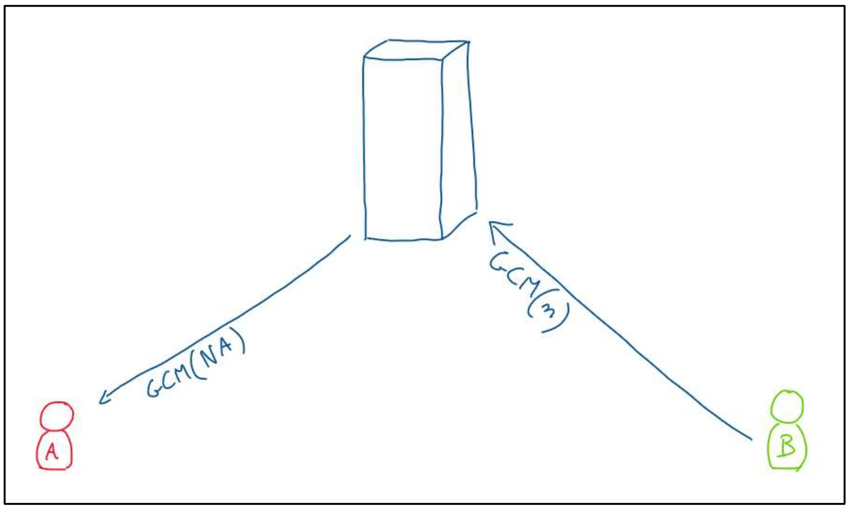
CHALLENGE USER



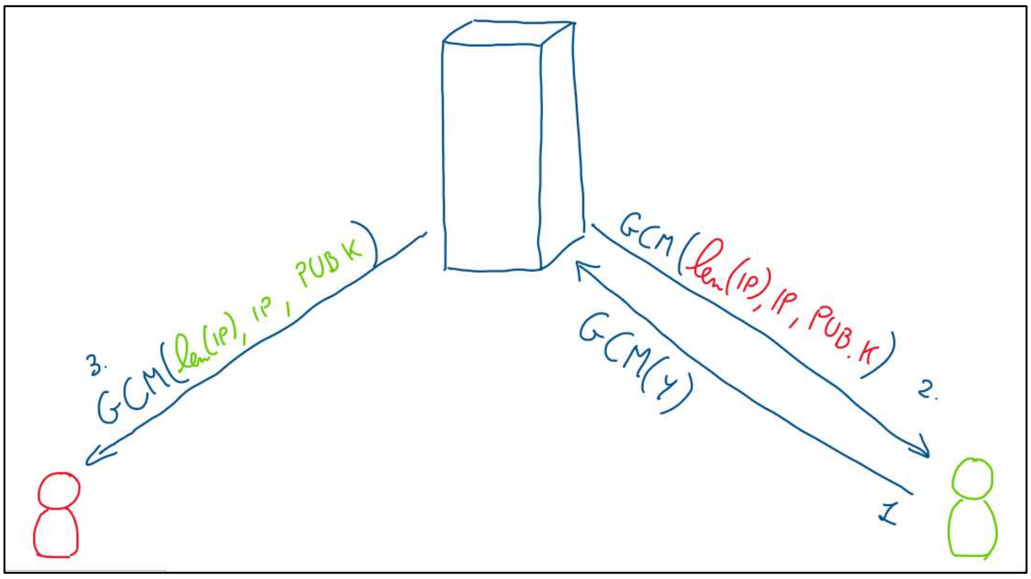
Let’s say Alice wants to challenge Bob:

1) Alice sends a challenge message to the server specifying Bob as the recipient
2) The server checks if Bob is in the list of READY users
    a. If Bob is in the list the server sends the username of the challenger to Bob
    b. Otherwise it responds directly to Alice with a not found (NF) message

At this point Bob has to choose yes or no to the challenge (y/n)



If Bob rejects the challenge a message “n” is sent to the server which notifies Alice with a not accepted (NA) message. Both Alice and Bob return to the lobby (which means that Bob is no longer blocked waiting for a challenger).



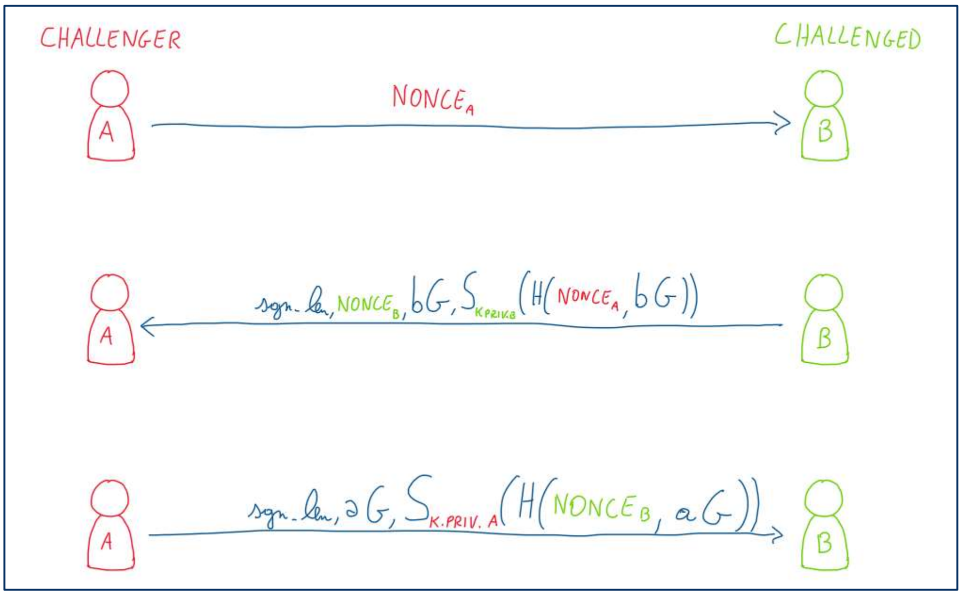
If Bob accepts the challenge a “y” message is sent to the server, then the server sends to both players a message containing the information of the adversary:

a. The length of the IP address of the adversary
b. The IP address of the adversary
c. The public key of the adversary

The players will now establish a P2P connection in which they will authenticate themselves and negotiate a session key.

P2P IDENTIFICATION + SESSION KEY GENERATION FOR A MATCH

The messages sent by the players pretty much follow the same structure as the client-server explained previously. The challenger starts:


CHALLENGER
CHALLENGED
A
B
NONCE_A
sgn_len, NONCE_B, bG, S_K.PRIV.B(H(NONCE_A, bG))
sgn_len, aG, S_K.PRIV.A(H(NONCE_B, aG))

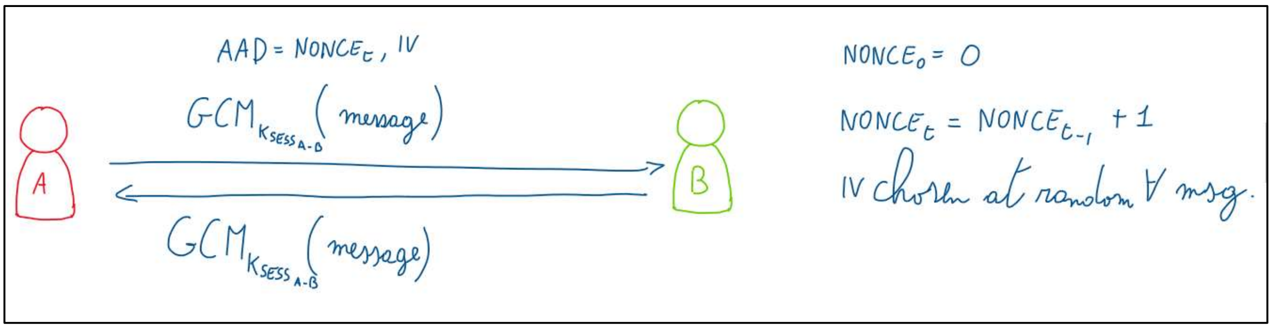
In this image we have:

- NONCEa which is the nonce of Alice
- aG which is the ECDH public key of Alice
- sgn_len is the length of the digital signature of Bob
- NONCEb which is the nonce of Bob
- bG which is the ECDH public key of Bob

The new session key for the game messages is:

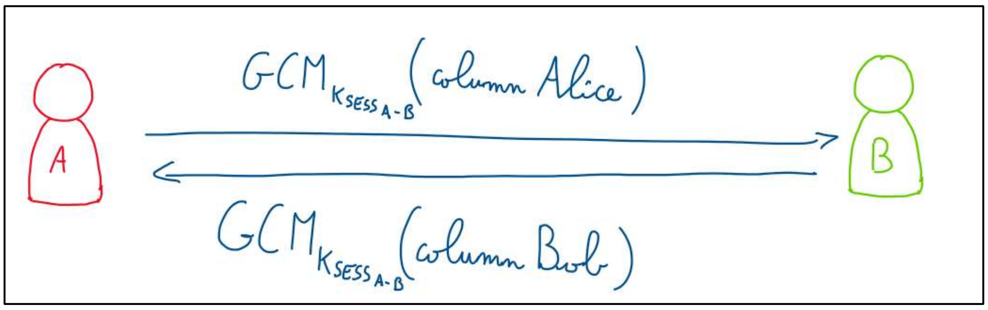
$$\boxed{K_{SESS.\,A-B} = H(abG)}$$

The following messages will be sent with an identical scheme as in the case of client-server:


$AAD = NONCE_t,\ IV$
$GCM_{K_{SESS_{A-B}}}(\text{message})$
A
B
$GCM_{K_{SESS_{A-B}}}(\text{message})$
$NONCE_0 = 0$
$NONCE_t = NONCE_{t-1} + 1$
IV chosen at random ∀ msg.

IN GAME MESSAGES

The only messages sent between the users contain just the number of the column in which the user sending the message is inserting a new peace [0-6]. The validity of the number sent is checked on both sides.


$GCM_{K_{SESS_{A-B}}}(\text{column Alice})$
A
B
$GCM_{K_{SESS_{A-B}}}(\text{column Bob})$

BAN-LOGIC PROOF

CLIENT-SERVER KEY EXCHANGE

ASSUMPTIONS

- $\xmapsto{k_u} U$
- $\xmapsto{k_s} S$
- $U \mid\equiv \xmapsto{k_s} S$
- $S \mid\equiv \xmapsto{k_u} U$
- $U \mid\equiv \#(N_u)$
- $S \mid\equiv \#(N_s)$

GOALS

Identification:

- $U \mid\equiv S \mid\equiv N_u$
- $S \mid\equiv U \mid\equiv N_s$

Shared key:

- $U \mid\equiv S \mid\equiv Y_s$
- $S \mid\equiv U \mid\equiv Y_u$

REAL PROTOCOL

- $M_1 \quad U \longrightarrow S \quad N_u$
- $M_2 \quad S \longrightarrow U \quad N_s, \{ N_u \}_{k_s^{-1}}$
- $M_3 \quad U \longrightarrow S \quad \{ N_s \}_{k_u^{-1}}$
- $M_4 \quad S \longrightarrow U \quad Y_s, \{ N_u, Y_s \}_{k_s^{-1}}$
- $M_5 \quad U \longrightarrow S \quad Y_u, \{ N_s, Y_u \}_{k_u^{-1}}$

IDEALIZED PROTOCOL

- $M_2 \quad S \longrightarrow U \quad \{ N_u \}_{k_s^{-1}}$
- $M_3 \quad U \longrightarrow S \quad \{ N_s \}_{k_u^{-1}}$
- $M_4 \quad S \longrightarrow U \quad \{ N_u, Y_s \}_{k_s^{-1}}$
- $M_5 \quad U \longrightarrow S \quad \{ N_s, Y_u \}_{k_u^{-1}}$

PROOF

- M_2:

$$\frac{U \mid\equiv \xmapsto{k_s} S, \quad U \triangleleft \{ N_u \}_{k_s^{-1}}}{U \mid\equiv S \mid\sim N_u}$$

- $$\frac{U \mid\equiv \#(N_u)\,,\; U \mid\equiv S \mid\sim N_u}{U \mid\equiv S \mid\equiv N_u}$$

- M_3: *same as* M_2
  - $S \mid\equiv U \mid\equiv N_s$

- M_4:
  - $$\frac{U \mid\equiv \overset{k_s}{\longmapsto} S\,,\; U \triangleleft \{N_u, Y_s\}_{k_S^{-1}}}{U \mid\equiv S \mid\sim (N_u, Y_s)}$$
  - $$\frac{U \mid\equiv \#(N_u)}{U \mid\equiv \#(N_u, Y_s)}$$
  - $$\frac{U \mid\equiv \#(N_u, Y_s)\,,\; U \mid\equiv S \mid\sim (N_u, Y_s)}{U \mid\equiv S \mid\equiv (N_u, Y_s)}$$
  - $$\frac{U \mid\equiv S \mid\equiv (N_u, Y_s)}{U \mid\equiv S \mid\equiv Y_s}$$

- M_5: *same as* M_4
  - $S \mid\equiv U \mid\equiv Y_u$

PEERS KEY EXCHANGE

ASSUMPTIONS

- $\overset{k_a}{\longmapsto} A$
- $\overset{k_b}{\longmapsto} B$
- $A \mid\equiv \overset{k_b}{\longmapsto} B$
- $B \mid\equiv \overset{k_a}{\longmapsto} A$
- $A \mid\equiv \#(N_a)$
- $B \mid\equiv \#(N_b)$

GOALS

Identification:

- $A \mid\equiv B \mid\equiv N_a$
- $B \mid\equiv A \mid\equiv N_b$

Shared key:

- $A \mid\equiv B \mid\equiv Y_b$
- $B \mid\equiv A \mid\equiv Y_a$

REAL PROTOCOL

- M_1 $A \longrightarrow B$ N_a
- M_2 $B \longrightarrow A$ $N_b\ ,\ Y_b\ ,\{\ N_a\ ,Y_b\ \}_{k_b^{-1}}$
- M_3 $A \longrightarrow B$ $Y_a\ ,\ \{\ N_b\ ,\ Y_a\}_{k_a^{-1}}$

IDEALIZED PROTOCOL

- M_2 $B \longrightarrow A$ $\{\ N_a\ ,Y_b\ \}_{k_b^{-1}}$
- M_3 $A \longrightarrow B$ $\{\ N_b\ ,\ Y_a\}_{k_a^{-1}}$

PROOF

- M_2:
  - $$\frac{A \mid\equiv \overset{k_b}{\longmapsto} B,\quad A \lhd \{\ N_a\ ,Y_b\ \}_{k_b^{-1}}}{A \mid\equiv B \mid\sim (N_a, Y_b)}$$
  - $$\frac{A \mid\equiv \#(N_a)}{A \mid\equiv \#(N_a, Y_b)}$$
  - $$\frac{A \mid\equiv \#(N_a, Y_b),\quad A \mid\equiv B \mid\sim (N_a, Y_b)}{A \mid\equiv B \mid\equiv (N_a, Y_b)}$$
  - $$\frac{A \mid\equiv B \mid\equiv (N_a, Y_b)}{A \mid\equiv B \mid\equiv N_a}$$
  - $$\frac{A \mid\equiv B \mid\equiv (N_a, Y_b)}{A \mid\equiv B \mid\equiv Y_b}$$

- M_3: *same as* M_2
  - $B \mid\equiv A \mid\equiv N_b$
  - $B \mid\equiv A \mid\equiv Y_a$