# Encriptación en Vortex – Modelo inicial

Este documento intenta bajar a texto algunas ideas de cómo manejar el encriptado de mensajes dentro de vortex contra el servidor http

## Vulnerabilidad del Modelo

El encriptado que se utilizará busca evitar que los mensajes sean interpretados fácilmente, brindando cierto grado de privacidad. Para ello se agregaría una URL nueva a la ya existente de manera que puedan convivir comunicaciones no encriptadas con otras que sí.

Esto significa que si un nodo NO encriptado se comunica con uno que sí, los mensajes que reciba este nodo PIERDEN el encriptado y todo tipo de privacidad. Es por ello que no deberían participar nodos no encriptados de comunicaciones con nodos encriptados. Sin embargo, por estar empezando con todo esto, y no tener la posibilidad de encriptar desde cualquier lado, dejamos esta vulnerabilidad abierta. Al menos hasta unificar y mejorar la seguridad (cuando tengamos más claro cómo).

## Alcance del encriptado

El encriptado planteado en este doc implica que el contenido de los requests HTTP serán encriptados. Se podría hacer que además viajen por HTTPS pero por temas de implementación (no todos los clientes pueden usar HTTPS) por ahora sólo utilizamos HTTP.

Esto significa que si alguien en el medio analiza los mensajes HTTP puede obtener una versión encriptada de los mensajes. Para poder recuperar el contenido debería conocer alguna de las claves usadas para desencriptar.

## Tipo de encriptado

Vamos a utilizar el modelo de clave pública/privada. Es decir, para poder enviar un mensaje a alguien tengo que conocer su clave pública y encriptar con ella. Para poder recibir el mensaje de alguien tiene que encriptar con mi clave pública y yo desencriptar con mi clave privada.

De esta manera intentamos eliminar la posibilidad de que terceros interpreten la comunicación sin conocer las claves privadas de cada nodo. Esto no impide que un tercero pueda generar mensajes en nombre de una de las partes en medio de la comunicación, pero al menos impide que pueda interpretar lo que se comunica.

## Esquema sin encriptado



En el modelo “tradicional” sin encriptado el mensaje HTTP contenía un WrapperVortex con los mensajes enviados en formato JSON.

Esta opción seguirá estando disponible pero cambiará la URL en que se publica “/controllers/naked” (para reflejar que si usas esta opción estás en pelotas).

Quizás este servicio lo demos de baja en un futuro, o puede quedar abierto para publicación de mensajes sin sesión, por ejemplo, proveniente de sensores (más simples de implementar).

## Esquema con encriptado

El encriptado tendría obligatoriamente 2 pasos. El intercambio de claves públicas por parte de los comunicantes, y el posterior intercambio de mensajes encriptados.  
Para esto se agrega una nueva URL con ese único objetivo: “/controllers/keys”

### Intercambio de claves públicas



El cliente HTTP que quiera comunicarse encriptado, deberá enviar un *PedidoDeSesionYClave*, en el que hace conocer al servidor cuál es la clave pública para recibir mensajes. Por otro lado, el servidor asocia a esa clave un ID de sesión, con el cuál valida los mensajes recibidos, y a su vez le hace conocer la clave pública del servidor con una respuesta *ConcesionDeSesionYClave*.

Vale aclarar que el ID de sesión que el servidor otorga al cliente, viene ya encriptada con la clave pública del cliente. El objetivo de esto, es evitar que un tercero conozca el ID de sesión otorgada al cliente. Lo que agrega un punto más para validar los mensajes del lado del servidor.

### Comunicación encriptada



La comunicación encriptada funciona muy similar a la “tradicional” o no encriptada, sólo que en último paso, antes de enviar un mensaje al servidor, el cliente debe ecriptar el texto JSON del VortexWrapper con la clave pública recibida, y al recibir una respuesta, desencriptar con la propia clave privada.

Este tipo de comunicaciones se realizará por una nueva URL “/controllers/crypted” que sólo espera este tipo de mensajes