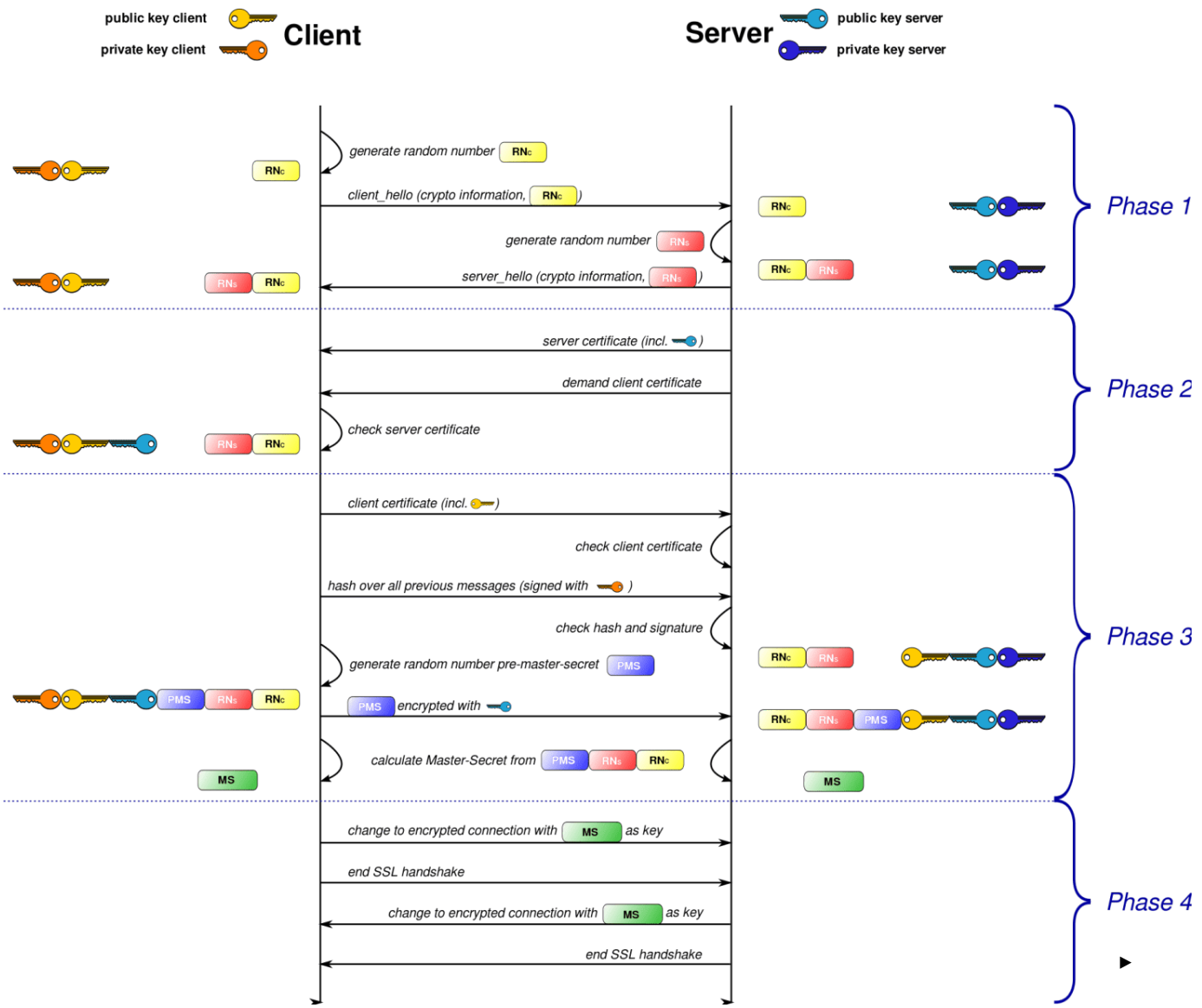
sockets seguros

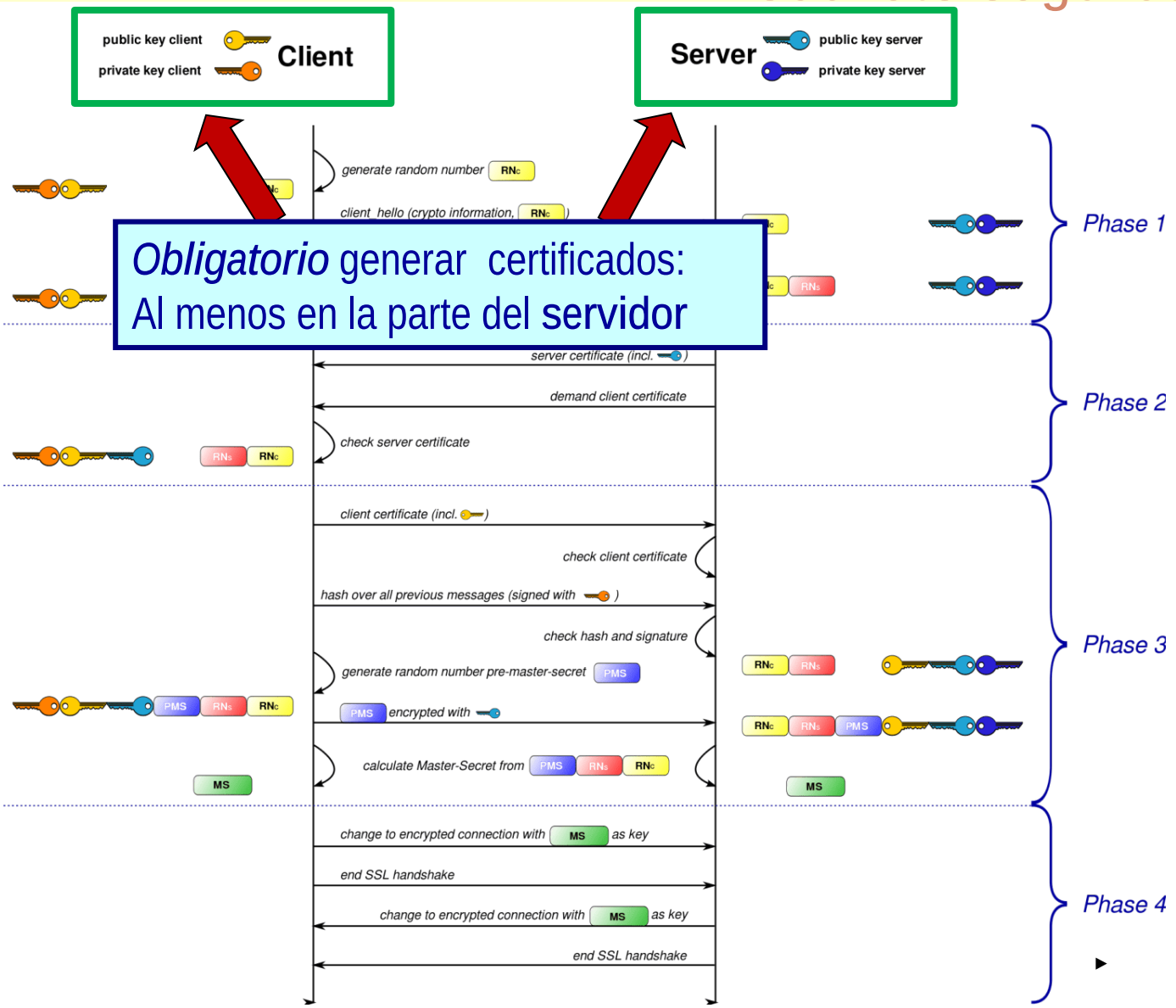
seguridad

sockets seguros



seguridad

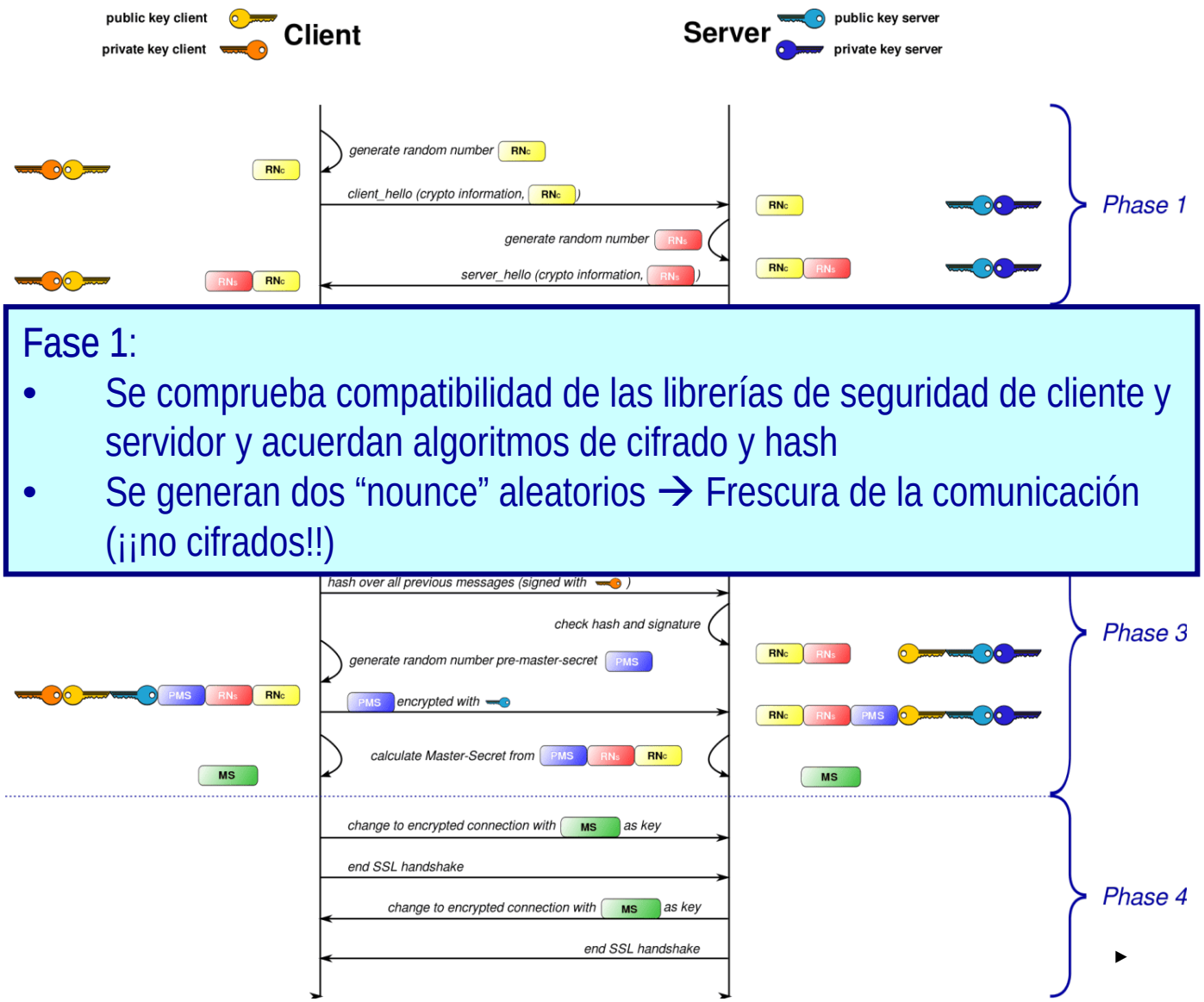
sockets seguros



Obligatorio generar certificados:
Al menos en la parte del servidor

seguridad

sockets seguros

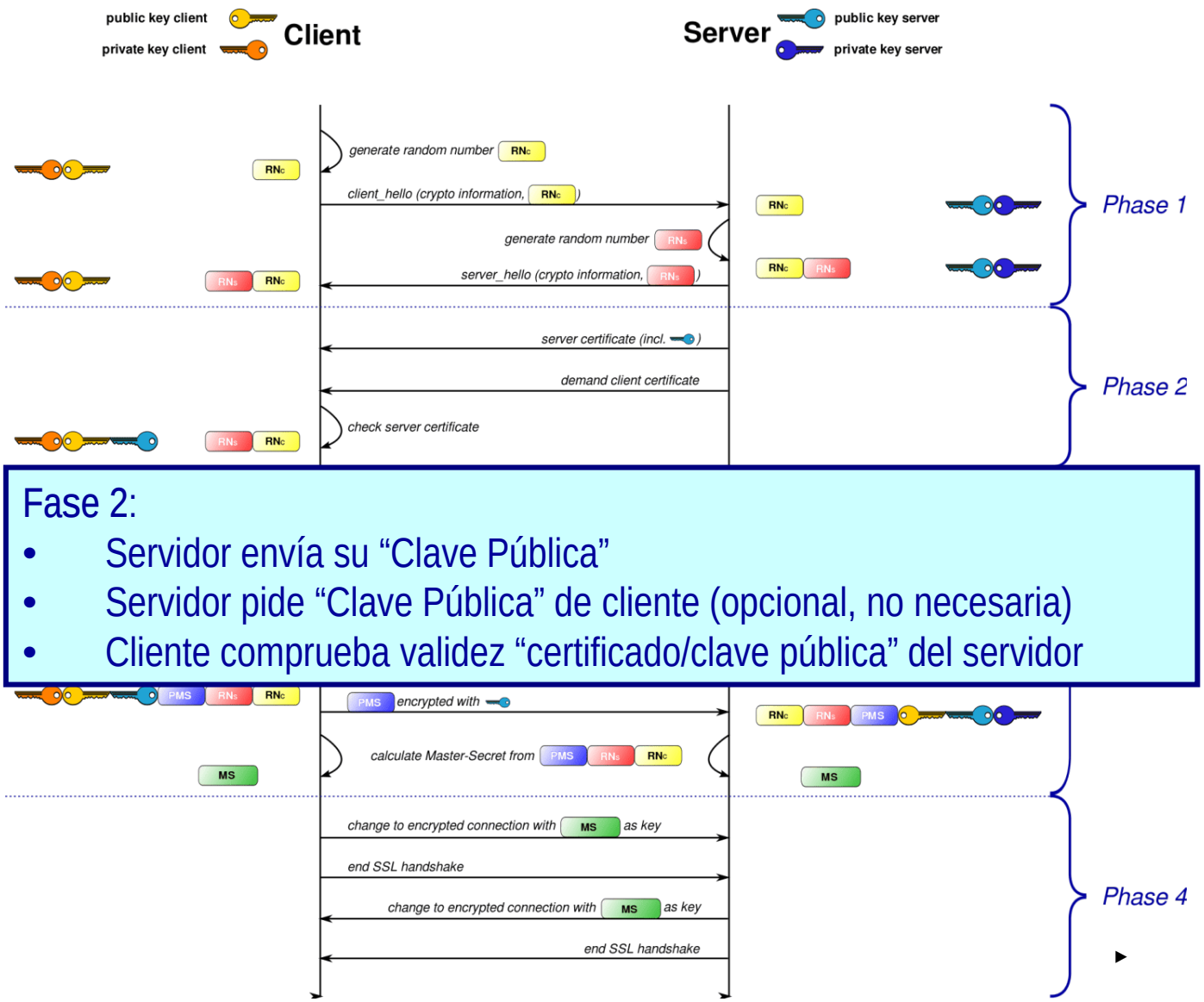


Fase 1:

- Se comprueba compatibilidad de las librerías de seguridad de cliente y servidor y acuerdan algoritmos de cifrado y hash
- Se generan dos “nounce” aleatorios → Frescura de la comunicación (¡¡no cifrados!!)

seguridad

sockets seguros

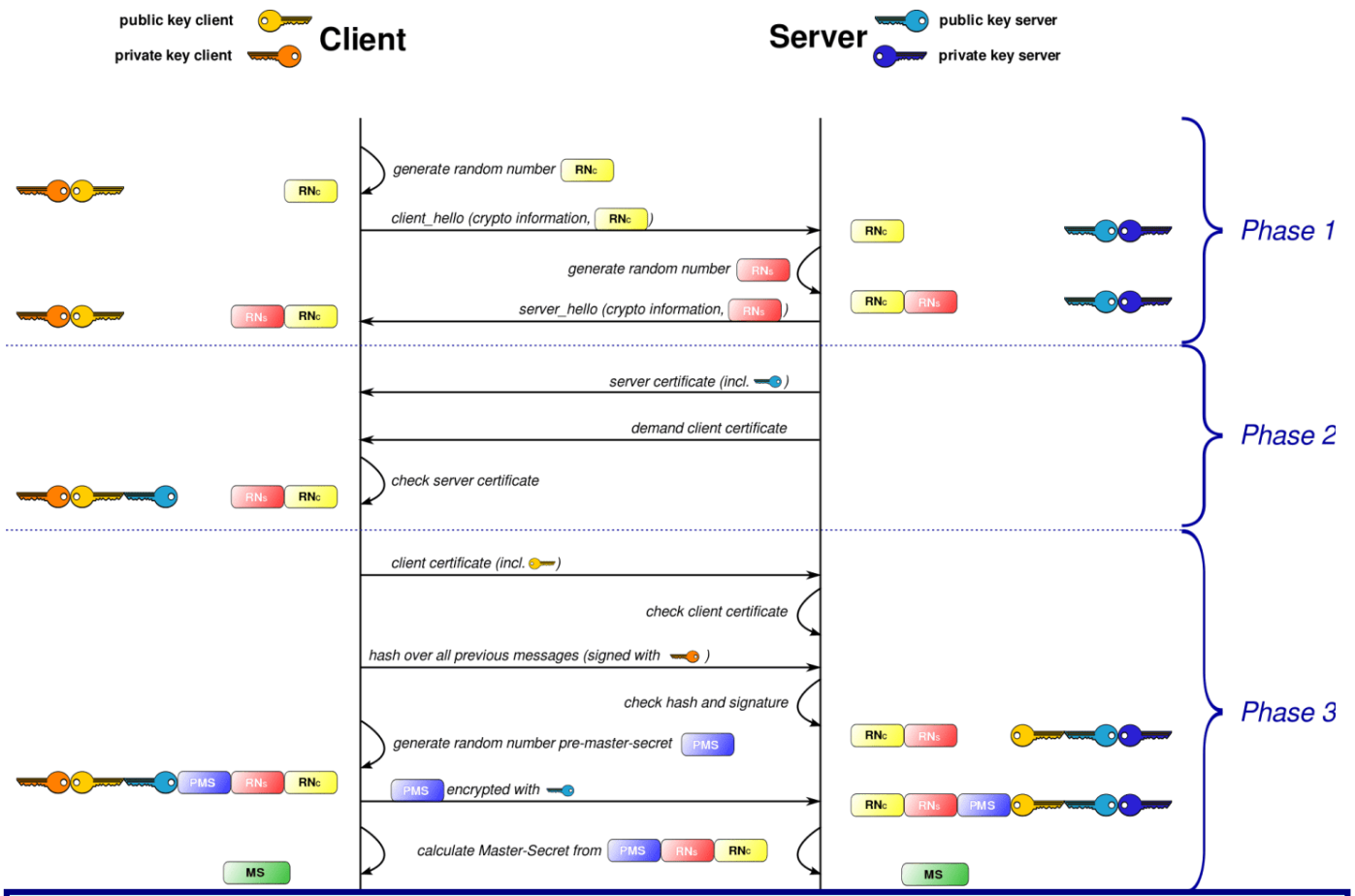


Fase 2:

- Servidor envía su “Clave Pública”
- Servidor pide “Clave Pública” de cliente (opcional, no necesaria)
- Cliente comprueba validez “certificado/clave pública” del servidor

seguridad

sockets seguros

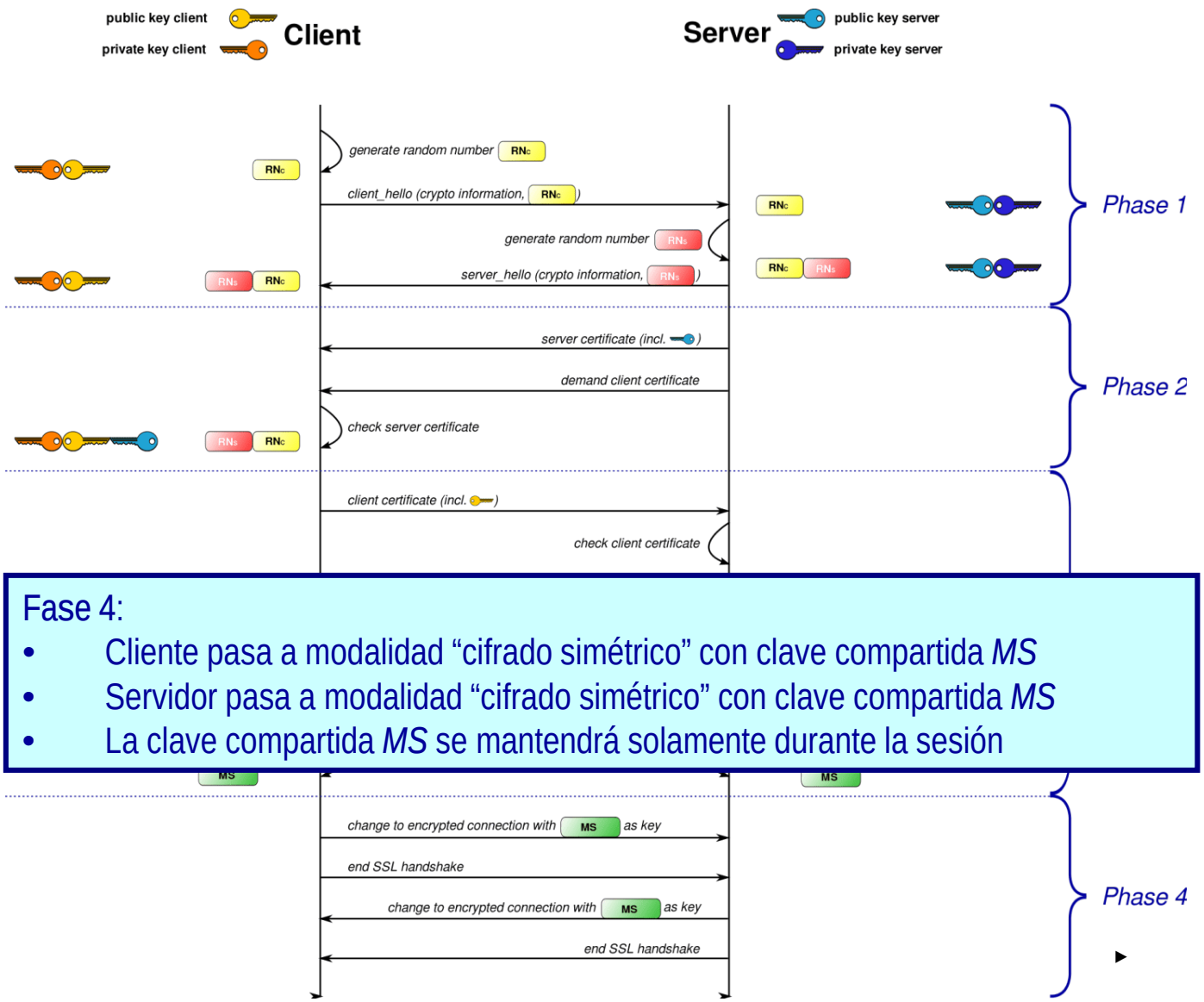


Fase 3:

- Cliente envía su “Clave Pública” (opcional, solo si tiene)
- Servidor comprueba validez “certificado/clave pública” del cliente (opc.)
- Cliente genera “Clave simétrica” *MS*, la cifra con “clave pública” de servidor y se la envía al servidor (generada a partir de *Aleatorio PMS + RN Cli + RN Serv*)

seguridad

sockets seguros



Fase 4:

- Cliente pasa a modalidad “cifrado simétrico” con clave compartida *MS*
- Servidor pasa a modalidad “cifrado simétrico” con clave compartida *MS*
- La clave compartida *MS* se mantendrá solamente durante la sesión

seguridad

aspectos técnicos (ubuntu)

- Instalación de las librerías OpenSSL:

  sudo apt-get update

  sudo apt-get install libssl–dev

  o mediante el gestor de paquetes de software (synaptic, …)

- Generación de los certificados (al menos de servidor):

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout certServ.pem -out certServ.pem

Opciones:

-nodes → Significa que no va a usar el algoritmo DES para cifrar la clave privada (¡ojo porque es un ejemplo sencillo¡ la clave privada aparece en el certificado sin cifrar)

-days 365 → El número de días de vigencia del certificado (y de las claves pública y privada), tras 1 año caducarán y ya no servirán

-rsa:2048 → Longitud de la clave generada y algoritmo de clave pública utilizado

Esta instrucción también solicita datos de dominio, sobre el propietario, email, “Common Name”, Country, etc. para que la CA (Certification Authority) pueda comprobar la autenticidad. En este caso se obvia, ya que crean certificados auto-firmados (self-signed).

seguridad
cliente – servidor C (ubuntu)

- Interpretación del código de **Servidor** (extracto):

```
SSL_library_init();
 portnum = Argc[1];
 ctx = InitServerCTX();        /* inicializa SSL */
 LoadCertificates(ctx, "mycert.pem", "mycert.pem"); /* carga de certificados */
 server = OpenListener(atoi(portnum));    /* creación del socket servidor */
 while (1)
 {
    struct sockaddr_in addr;
    socklen_t len = sizeof(addr);
    SSL *ssl;
    int client = accept(server, (struct sockaddr*)&addr, &len);  /* acepta conexiones entrantes */
    printf("Connection: %s:%d\n",inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
    ssl = SSL_new(ctx);              /* obtención de contexto de conexión seguro */
    SSL_set_fd(ssl, client);      /* conversión del socket en socket seguro SSL */
    Servlet(ssl);         /* se sirve la petición al servidor */
 }
 close(server);          /* cierre del socket servidor */
 SSL_CTX_free(ctx);         /* liberación de contexto seguro */
```

seguridad
cliente – servidor C (ubuntu)

Interpretación del código de **Cliente** (extracto):

```
SSL_library_init();
ctx = InitCTX();
server = OpenConnection(hostname, atoi(portnum)); /* Conexión no-cifrada con servidor */
 ssl = SSL_new(ctx);      /* crea un nueva conexión SSL */
 SSL_set_fd(ssl, server);    /* asocia el descriptor no-cifrado a la conexión cifrada*/
 if ( SSL_connect(ssl) == FAIL )   /* conecta con el servidor de forma cifrada */
    ERR_print_errors_fp(stderr);
 else
 {
   const char *cpRequestMessage = "Usuario: %s   Contraseña: %s";
        …
   sprintf(acClientRequest, cpRequestMessage, acUsername,acPassword);   /* crea contestación */
   SSL_write(ssl,acClientRequest, strlen(acClientRequest));   /* encripta y envía el mensaje */
   bytes = SSL_read(ssl, buf, sizeof(buf)); /* obtiene la respuesta y la desencripta */
   buf[bytes] = 0;
   printf("\nRecibido: \"%s\"\n", buf);
   SSL_free(ssl);        /* libera la conexión cifrada */
 }
 close(server);         /* cierra el socket */
 SSL_CTX_free(ctx);        /* libera el contexto: algoritmos de cifrado, etc. */
```

seguridad

cliente – servidor C (ubuntu)

- Compilación de los programas:

  gcc -Wall -o servidor servidor.c -L/usr/lib -lssl -lcrypto

  gcc -Wall -o cliente cliente.c -L/usr/lib -lssl -lcrypto

- Ejemplo de ejecución de los programas:

  sudo ./servidor 8090 (se requiere ejecutarlo como root)

  ./cliente 127.0.0.1 8090 (la dirección del servidor es localhost)

- Resultado:

  Solamente Usuario SD y Contraseña: 12345678 confirman el acceso por parte del servidor