Gobierno, Gestión y Tecnologías de Ciberseguridad

CAI 1
(ST 2)

HOJA DE CONTROL DEL DOCUMENTO

Documento:	CAI1-15.05.2024
Denominación:	CAI 1
Edición:	V2
Fecha:	19/05/2024
Realizado por:	Equipo INSEGUS ST2
Aprobado por:	Equipo INSEGUS ST2

ÍNDICE

1. Resumen ejecutivo 3
2. Consultas 3
    1.1. Consulta 1 3
    1.2. Consulta 2 4
    1.3. Consulta 3 4
3. Bibliografía 6

EQUIPO INSEGUS

Nombre	Rol
Francisco Javier Cavero López	Consultor

CONTROL DE MODIFICACIONES

Nº de edición	Fecha	Comentarios	Autor
V1	15/05/2024	Se crea el documento	Francisco Javier Cavero López
V2	19/05/2024	Se mejora la redacción de los apartados y se incluye la tercera consulta y el resumen ejecutivo	Francisco Javier Cavero López

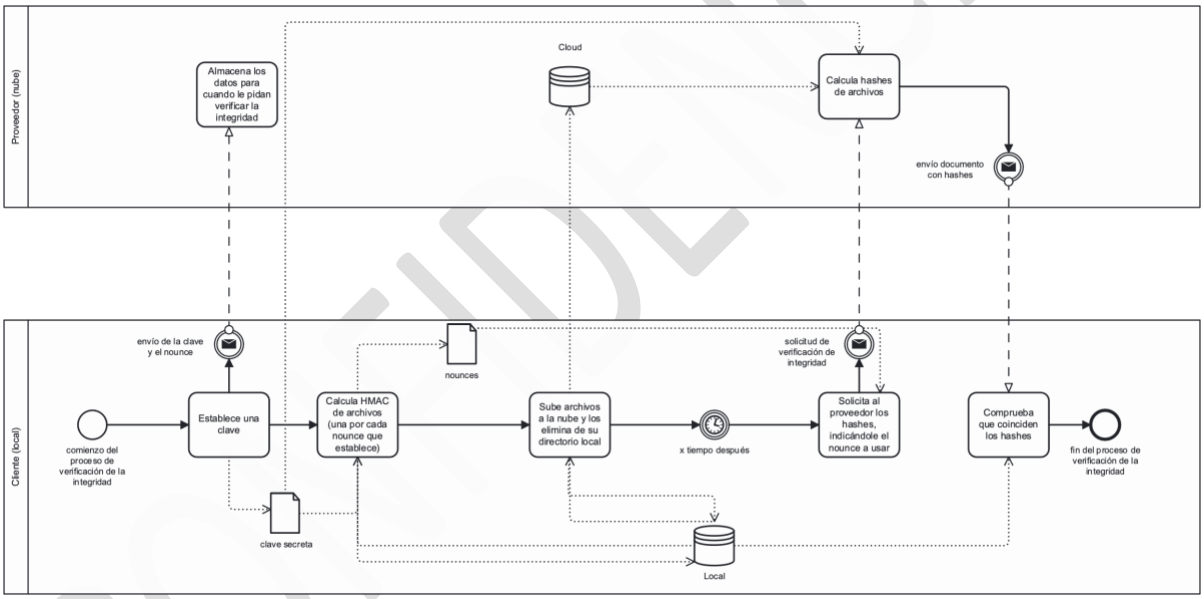
1. Resumen ejecutivo

Este documento presenta un análisis y una serie de recomendaciones para la implementación de mecanismos flexibles y económicos de verificación de integridad de datos y cifrado en tres contextos distintos, abordando problemas específicos relacionados con la seguridad de la información. Se han diseñado y probado varias soluciones utilizando protocolos y algoritmos criptográficos modernos, evaluando su eficiencia y seguridad. Cada solución propuesta ha sido implementada en prototipos de prueba utilizando el lenguaje de programación Python, y se proporcionan enlaces a los repositorios correspondientes para su revisión y aplicación práctica.

2. Consultas

1.1. Consulta 1

El modelo de negocio para poder llevar a cabo la verificación planteada en la consulta es el representado en el siguiente diagrama:



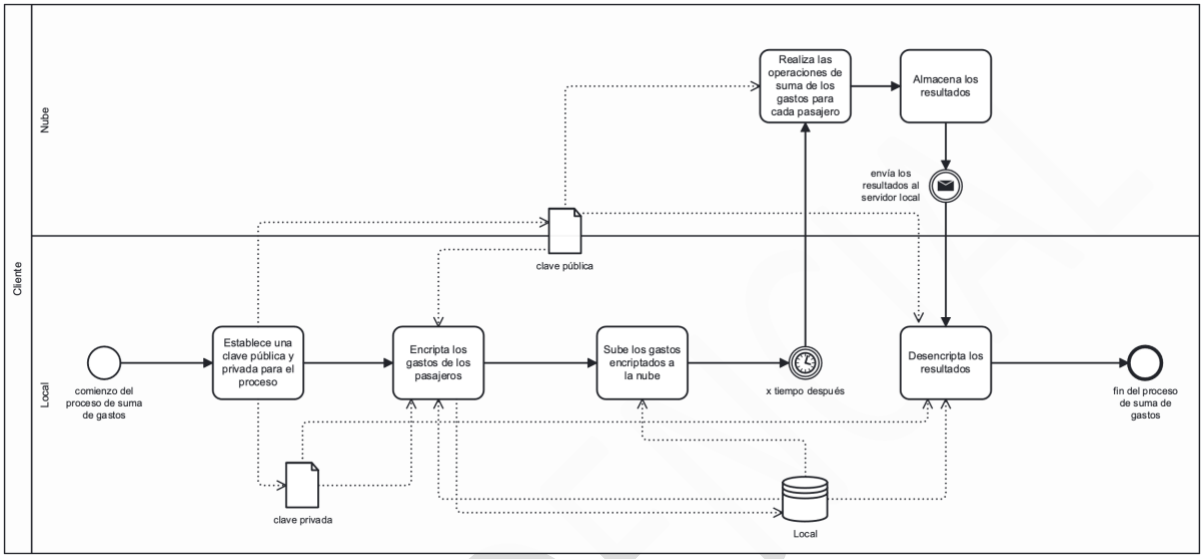
El protocolo HMAC que va a utilizarse, va a integrar una función de hash256 y el uso de nounce para garantizar la integridad de la información. Se ha implementado un pequeño ejemplo “de juguete” utilizando el lenguaje Python en el el que se realiza una pequeña aproximación de una variación (para simplificar la implementación) del protocolo propuesto. También, se ha implementado una suite de pruebas cuyos resultados indican que el nounce y la clave son esenciales para poder garantizar esta integridad. Además, los tests garantizan que el protocolo es veloz y eficiente (los tiempos de ejecución del protocolo son inferiores al segundo). El código implementado (incluyendo los tests) puede encontrarse en: https://github.com/javiercavlop/cibersecurity/tree/main/cai1/c1 y puede ejecutarse con cualquier ordenador con Python (>=3.8.13).

En cuanto a la recuperación de la información comprometida, lo mejor es recuperarla a través de una copia de seguridad. La idea sería que diariamente, tras verificar que los archivos están íntegros, se realice una copia de seguridad de ese archivo, de forma que en el momento que se detecte un problema, podamos recuperar la versión del día anterior. En caso de que los archivos no deban sufrir

nunca cambios (sean readonly) con realizar una copia de seguridad antes de subirlos a la nube valdría. La comprobación de la integridad debería hacerse diariamente y si salta algún error al utilizar el protocolo propuesto, se recupera inmediatamente el archivo de la copia de seguridad.

1.2. Consulta 2

El modelo de negocio para poder llevar a cabo la verificación planteada en la consulta es el representado en el siguiente diagrama:

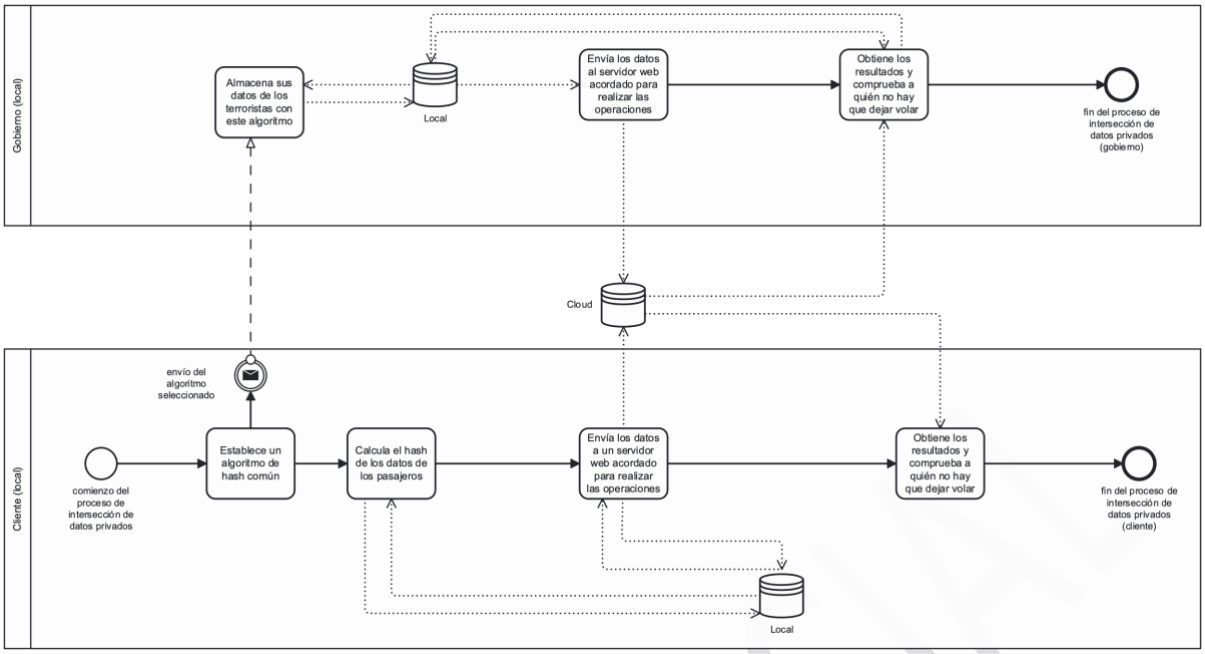


Para la resolución de esta consulta, proponemos el uso del criptosistema de Paillier[1] para la generación de las claves y la encriptación de los valores. Con este criptosistema homomórfico podemos sumar los valores encriptados con la clave pública en la nube y sólo con la clave privada y en nuestro servidor local, desencriptarlos.

Un ejemplo "de juguete" de la implementación en Python (>=3.8.13) y que utiliza la librería phe (>=1.5.0) puede encontrarse en: https://github.com/javiercavlop/cibersecurity/tree/main/cai1/c2. Se ha realizado un pequeño caso de prueba que arroja resultados preocupantes. El tamaño de las claves incrementa sustancialmente el tiempo de ejecución. Recomendamos usar un tamaño de 3072 bits aunque puede que en producción, si se busca velocidad, se prefiera sacrificar algo de seguridad, reduciendo el tamaño de la clave. Aún así, como mínimo debería ser de 2048 bits[2]. El testing indica que con una clave de 3072 bits las encriptaciones son más de 3 veces más lentas que con una de 2048 y el problema temporal del criptosistema viene de la demora producida en la fase de encriptación/desencriptación.

1.3. Consulta 3

El modelo de negocio para poder llevar a cabo la verificación planteada en la consulta es el representado en el siguiente diagrama:



El protocolo desarrollado consiste en una adaptación del protocolo Private Set Intersection[3]. Comenzaremos con una encriptación de los datos realizada por un algoritmo de hash. Como algoritmo proponemos el sha256, pero puede ser sustituido por cualquiera de los analizados en la fase de testing y que tengan tamaños superiores al sha256. Una vez encriptados los datos, se envían a un servidor de la nube donde se realizan una intersección con los datos del gobierno sobre las personas que no pueden volar. Los resultados se descargan y se comparan los hashs resultantes con los de la lista de entrada. Así cada organización puede detectar las coincidencias con los nombres de sus listas.

Para la fase de testing se ha realizado un pequeño test en el que se ha probado a medir el tiempo entre distintos tamaños de funciones de hash, obteniendo resultados similares en todas sus versiones. Se recomienda no usar un tamaño inferior al de sha256. Para los datos de las pruebas se ha utilizado un listado de 81000 strings que equivaldría al tamaño aproximado de la lista de “no flight” del FBI[4] y un listado de 1000 strings equivalente a la ocupación máxima de un avión comercial (<868 personas)[5].

Se ha implementado un pequeño caso práctico (“de juguete”) del algoritmo en el que se puede poner en práctica el protocolo y el funcionamiento del algoritmo. Se encuentra disponible en: https://github.com/javiercavlop/cibersecurity/tree/main/cai1/c3. Los tests también pueden ejecutarse utilizando Python (>=3.8.13).

Los resultados del testing arrojan una eficiencia óptima (la duración de las ejecuciones es inferior al segundo) de este algoritmo para asegurar la privacidad en el proceso dado que la cantidad de datos no es demasiado grande (en otro contexto, esta propuestas se volvería demasiada lenta cuando los datasets son enormes). Sin embargo, recomendamos analizar otras alternativas que garanticen más seguridad en la privacidad de la información y que aseguren la eficiencia necesaria para el contexto actual.

3. Bibliografía

[1] Paillier, P. (1999). *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Advances in Cryptology – EUROCRYPT '99 (pp. 223–238)*. Springer. doi:10.1007/3-540-48910-X_16
[2] Giry, D. (2024, May 24). *Criptographic Key Length Recommendation (v. 32.3)*. BlueKrypt. https://www.keylength.com/en/8
[3] Rosulek, M. (2002, February). *Security and Privacy Properties for Private Set Intersection*. National Institute of Standards and Technology (NIST). https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.2000-01.pdf
[4] Larenas, Nicolás (2021, August 31) *¿Cuál es el record máximo de pasajeros que ha llevado un avión?* https://www.nlarenas.com/2021/08/record-maximo-pasajeros-llevado-un-avion/
[5] Leonard, K. (2017, 30 de marzo). *No Fly Lists*. National High School Ethics Bowl.