(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 288 863**

(51) Int. Cl.:
H04L 9/08 (2006.01)
H04L 9/32 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **00948590 .5**
(86) Fecha de presentación : **29.06.2000**
(87) Número de publicación de la solicitud: **1197032**
(87) Fecha de publicación de la solicitud: **17.04.2002**

(54) Título: **Regeneración asistida por servidor segura de un secreto fuerte a partir de un secreto débil.**

(30) Prioridad: **29.06.1999 US 141571 P**
23.11.1999 US 167453 P
10.03.2000 US 188834 P
17.05.2000 US 574687

(45) Fecha de publicación de la mención BOPI:
01.02.2008

(45) Fecha de la publicación del folleto de la patente:
01.02.2008

(73) Titular/es: **Verisign, Inc.**
487-E Middlefield Road
Mountain View, California 94043, US

(72) Inventor/es: **Ford, Warwick, S.**

(74) Agente: **Carpintero López, Francisco**

DESCRIPCIÓN

Regeneración asistida por servidor segura de un secreto fuerte a partir de un secreto débil.

1. **Campo técnico**

Esta invención se refiere en general a la regeneración segura de un secreto fuerte de usuario cuando el usuario suministra un secreto débil correspondiente, tal como una contraseña elegida por el usuario. Por ejemplo, en aplicaciones de redes de ordenadores, el secreto fuerte podría ser una clave de encriptado que se usa para proteger los datos privados altamente sensibles del usuario (tales como la clave privada del usuario usada en criptografía de clave pública). En este ejemplo, la invención se refiere a la regeneración segura de la clave de encriptado (y a la recuperación segura de la clave privada de usuario) cuando el usuario suministra su contraseña. Como otro ejemplo, el usuario podría usar el secreto fuerte para autenticarse en un servidor, demostrando al servidor la capacidad del usuario para regenerar el secreto fuerte, sin necesidad de que ese servidor almacene datos que permitan que el secreto débil sea atacado por medio de pruebas exhaustivas.

2. **Técnica antecedente**

Como resultado del desarrollo continuo de nuevas tecnologías, en particular en las áreas de redes de ordenadores y comunicaciones, el uso de grandes redes tales como Internet se está comenzando a extender ampliamente. Esto ha dado como resultado un aumento en el comercio electrónico y otras transacciones electrónicas dirigidas sobre estas redes. También ha dado como resultado una flexibilidad aumentada para los usuarios, ya que éstos pueden acceder en mayor medida a estas redes desde cualquier número de localizaciones y/o dispositivos. El aumento en las transacciones electrónicas ha dado como resultado una correspondiente necesidad aumentada de seguridad para estas transacciones; pero la flexibilidad aumentada impone requisitos adicionales sobre seguridad ya que cualquier medida de seguridad de manera preferible acomodaría a los usuarios incluso mientras navegan por la red.

En un escenario común, el usuario puede acceder a la red de ordenadores desde muchas localizaciones diferentes y puede desear usar su clave privada desde cada una de las localizaciones. Sin embargo, en cada una de las localizaciones, el usuario puede estar accediendo a la red desde un dispositivo (al que se hace referencia de aquí en adelante en este documento como "terminal de cliente") que no puede almacenar o no almacena datos para el usuario, distintos a datos durante períodos transitorios. Por ejemplo, un empleado podría acceder a una red de ordenadores central de empresa desde terminales diferentes que se encuentren en las instalaciones de la empresa, o un consumidor podría acceder a Internet desde cualquier navegador web o podría acceder a una red privada desde un kiosco de consumidor. De manera típica se puede confiar en el terminal de cliente para ejecutar su código de una manera fiable para mantener el secreto de datos sensibles (por ejemplo, la clave privada del usuario o un secreto compartido con un servidor de aplicaciones) durante el período en el que el usuario esté usando de manera activa ese terminal, y para destruir de manera segura los datos sensibles cuando el usuario haya acabado de usarlos. De esta manera, los datos privados del usuario se podrían usar de manera segura en el terminal de cliente si el terminal de cliente pudiese obtener de algún modo de manera segura una copia de los datos privados.

En un enfoque, los datos privados se almacenan en algún almacenamiento hardware seguro o tarjeta de procesamiento, tal como una tarjeta inteligente. La tarjeta de procesamiento está conectada físicamente al terminal de cliente y los datos privados se hacen accesibles al cliente. Este enfoque sufre de un alto coste y de la inconveniencia para el usuario ya que se requiere un hardware dedicado, haciéndola de esta manera inapropiada para muchas aplicaciones.

En otro enfoque, los datos privados se recuperan con la ayuda de otros dispositivos conectados a la red (denominados de aquí en adelante en este documento como "servidores". En otro ejemplo, la recuperación de los datos privados es parte del proceso de registro de entrada de sesión del usuario. El usuario se autentica presentando un nombre de cuenta de usuario y una contraseña sometida a controles modestos (pero no extremos) de capacidad de adivinación. En particular, cualquier parte que intente un ataque de adivinación de contraseña está limitado a un pequeño número de intentos, y dado este control, se permite a los usuarios elecciones de contraseña razonablemente amigables.

Una vez que el usuario está autenticado, el terminal de cliente recupera los datos privados del usuario con la ayuda de los servidores.

El problema de recuperar datos privados, tales como una clave privada, en un terminal de cliente que no conserva un estado constante entre transacciones se ha tratado en trabajos anteriores mediante el uso de un servidor que almacena datos secretos para el cliente y facilita el proceso de recuperación. Se examinaron varios protocolos para el uso de dichos servidores con diferentes características de seguridad y de funcionamiento en el documento de R. Perlman y C. Kaufman, titulado "Protocolo Seguro basado en Contraseña para la descarga de una Clave Privada", *Proc. 1999 Network and Distributed System Security Symposium*, Internet Society, enero de 1999. Los protocolos descritos en ese trabajo son principalmente variantes o derivados del protocolo EKE de Bellovin y Merritt (por ejemplo, véase el documento de S. Bellovin y M. Merritt, titulado "Intercambio de clave encriptada: Protocolos basados en contraseña seguros contra ataques de diccionario", *Proc. IEEE, Symposium on Research in Security and Privacy*, mayo de 1992; y en el documento de S. Bellovin y M. Merritt, "Intercambio aumentado de clave encriptada: un protocolo basado en contraseña seguro contra ataques de diccionario y compromiso de fichero de contraseña", *ATT Labs Technical Report*, 1994) y el protocolo SPEKE de Jablon (por ejemplo, véase el documento de D. Jablon titulado "Intercambio de clave

autenticada sólo con contraseña fuerte", *ACM Computer Communications Review*, octubre de 1996; y el documento de D. Jablon titulado "Protocolos de contraseña ampliados inmunes a ataques de diccionario", *Proc. del WETICE '97 Enterprise Security Workshop*, junio de 1997. Patentes relacionadas incluyen la patente de los Estados Unidos número 5.241.599 ("Protocolo criptográfico para comunicaciones seguras" de Bellovin y Merritt) y la patente de los Estados Unidos número 5.440.635 ("Protocolo criptográfico para autenticación remota", de Bellovin y Merritt). Otros protocolos relacionados secretos de recuperación asistidos por un servidor han sido propuestos por Gong y colaboradores (por ejemplo, L. Gong, T. M. A. Lomas, R. M. Needham y J. H. Salzer, "Protección de secretos pobremente elegidos de ataques de adivinación", *IEEE Journal on Selected Areas in Communications*, volumen 11, número 5, junio de 1993, páginas 648 a 656; L. Gong, "Protocolos Óptimos de Autenticación resistentes a Ataques de Adivinación de Contraseña", *Proc. 8° IEEE Computer Security Foundations Workshop*, Irlanda, 13 de junio de 1995, páginas 24 a 29; y L. Gong, "Aumento de la disponibilidad y de la seguridad de un servicio de autenticación", IEEEE Journal on Selected Areas in Communications, volumen 11, número 5, junio de 1993, páginas 657 a 662); por Wu (por ejemplo, T. Wu, "El Protocolo de Contraseña Remota Segura", *Proc. 1998 Network and Distributed System Security Symposium*, Internet Society, enero de 1998, páginas 97 a 111) y por Halevi y Krawcyzk (por ejemplo, S. Halevi y H Krawczyk, "Criptografía de clave pública y protocolos de contraseña". "Criptografía de clave pública y protocolos de contraseña", *Proceedings of the Fifth ACM Conference on Computer and Communications Security, 1998)*.

Sin embargo, todos los procedimientos anteriores sufren un defecto significativo. El servidor representa una vulnerabilidad principal. Si un operador de servidor o alguien que comprometa a un servidor desee determinar una contraseña de usuario o datos privados (tanto la contraseña de usuario como los datos privados habilitando al atacante a hacerse pasar como el usuario), son posibles ataques viables, a pesar de aspectos de algunas aproximaciones que minimizan la sensibilidad de los datos almacenados en el servidor. Por ejemplo, en cierto del trabajo anteriormente mencionado, el servidor no almacena la contraseña del usuario sino que en lugar de esto almacena un valor calculado como una función unidireccional de la contraseña. Cualquiera que aprenda que el valor podría ser capaz de determinar la contraseña por medio de la aplicación exhaustiva de la función unidireccional para averiguar contraseñas y comparar el resultado con el valor almacenado. En términos generales, las aproximaciones anteriormente mencionadas sufren de debilidad en que cualquiera que pueda acceder a la base de datos del servidor o pueda inhabilitar cualquier mecanismo de estrangulación o bloqueo en el servidor puede probar contraseñas de manera exhaustiva hasta que entre en la cuenta del usuario en el servidor.

En algunos escenarios de aplicaciones la debilidad anteriormente mencionada puede socavar de manera significativa el atractivo de la aproximación asistida por servidor para recuperar datos privados. Por ejemplo, el escenario de ataque anterior restringe de manera significativa las propiedades de no repudiación en cualquier otro caso inherentes en tecnología de firma digital. Si un usuario itinerante firma una transacción usando la clave privada de usuario desde un terminal de cliente y posteriormente desea denegar la firma digital de usuario, el usuario puede reclamar de manera convincente que el operador de servidor o alguien que comprometa al servidor obtuvo la clave privada de usuario como se ha descrito anteriormente y que firmó digitalmente la transacción haciéndose pasar por el usuario. Los riesgos y la fiabilidad a las que hace frente un operador de servidor se reducen si puede de manera justificada contrarrestar las peticiones de usuarios que tanto ellos como su personal pueden haberse hecho pasar como el usuario.

De esta manera existe una necesidad de una aproximación que permita a un terminal de cliente recuperar datos privados del usuario con la ayuda de servidores a la vez que permanecen resistentes a ataques en los servidores. De manera más general, el problema de recuperar datos privados en un cliente que no conserva un estado constante entre transacciones se puede ver reducido al problema de generar y regenerar datos secretos fuertes para un usuario a partir de los datos secretos débiles de usuario, tales como una contraseña. El secreto fuerte se puede usar como una clave de encriptado en un criptosistema simétrico para encriptar y recuperar cualquier cantidad de datos privados que se podrían conservar en formato encriptado en un lugar de almacenamiento ampliamente accesible.

Existe también una necesidad de aproximaciones que permitan a un usuario autenticarse en un servidor de aplicaciones desde un terminal de cliente que no conserva un estado constante entre transacciones en base a una contraseña presentada. Aproximaciones actuales, tales como el procedimiento de autenticación Kerberos (por ejemplo, véase el documento de J. T. Kohl y B. C. Neuman, *El servicio de autenticación de red Kerberos (V5)*, Petición de comentarios (RFC) 1510, Internet Activities Board, 1993), implica al usuario para que primero se autentique en un servidor de autenticación y posteriormente en un servidor de aplicaciones usando un "billete" criptográfico del servidor de autenticación. Sin embargo, estas aproximaciones sufren de las deficiencias de que o el servidor de aplicaciones o el servidor de autenticación conservan datos que, si son expuestos a un atacante (ya sea interno o externo a la organización que haga funcionar el servidor), permite que el atacante adivine de manera exhaustiva las contraseñas y probablemente determine la contraseña del usuario. Estos problemas se pueden prevenir si la autenticación de usuario se basa en que un usuario presente una prueba de conocimiento de datos secretos fuertes en lugar de datos secretos débiles al servidor de aplicaciones o al servidor de autenticación.

También existe una necesidad de aproximaciones que permitan a un usuario crear una firma digital desde un terminal de cliente que no conserva un estado constante entre transacciones dentro del que se introduce una contraseña. Una aproximación para satisfacer este requisito es recuperar la clave privada de usuario que está dentro del terminal como se ha esbozado con anterioridad y calcular la firma digital en el terminal de cliente. Otra aproximación para satisfacer el requisito, que no requiere que la clave privada sea juntada en un lugar, implica las comunicaciones con múltiples servidores cada uno de los cuales conserva una parte independiente de la clave privada de firma del usuario. Dichos servidores generan cada uno de ellos una parte de la firma digital y las partes se combinan en el terminal

de cliente para dar la firma digital completa. Mientras que otro trabajo pertinente basado en dichos procedimientos consigue el objetivo de no juntar la clave privada en un lugar, dicho trabajo sufre la debilidad de que uno o más servidores que participan en el proceso de señalización conservan datos que permitan que la contraseña del usuario se vea atacada de manera exhaustiva. Por consiguiente, existe un riesgo de que cualquiera que comprometa cualquiera de dichos servidores pueda determinar la contraseña del usuario por medio de la adivinación exhaustiva, lo que, en caso de tener éxito, permite que, el atacante falsifique firmas digitales dando a entender que son de ese usuario. El problema se puede prevenir por medio de la autenticación a los servidores que generan partes de la firma digital demostrando el conocimiento de un secreto de usuario fuerte, que se pueden haber regenerado en base a la presentación de un secreto de usuario débil, en lugar de por medio de la autenticación directamente en base al propio secreto de usuario débil.

De esta manera, existe una necesidad de una aproximación que permita a un terminal de cliente regenerar datos secretos fuertes de usuario a partir de datos secretos débiles con la ayuda de servidores a la vez que permanecen resistentes a ataques en los servidores.

En las reivindicaciones anejas se declaran aspectos de la presente invención.

De acuerdo con una realización, un procedimiento para establecer (200, 500) datos secretos fuertes de usuario (100) para permitir la posterior recuperación (400, 600) de datos secretos fuertes, incluye los siguientes pasos. Se determinan (320) para el usuario (110) datos secretos débiles, por ejemplo una contraseña. El usuario se autentica (310) en los servidores (130), que incluyen a servidores que guardan secretos (de manera preferible al menos dos servidores que guardan secretos). Cada servidor que conserva secretos tiene correspondientes datos de secreto del servidor. Un cliente generador (120) posiblemente ayudado por los servidores que guardan secretos (130) calcula (330, 350) los datos secretos fuertes de usuario. Los datos secretos fuertes son una función de los datos secretos débiles del usuario y de los datos secretos del servidor. En una realización preferida, se calculan los componentes secretos (534) para cada uno de los servidores que guardan secretos. Cada componente secreto es una función de los datos secretos débiles y de los datos secretos fuertes para ese servidor que conserva secretos. Los componentes secretos se combinan (536) para generar los datos secretos fuertes. El cliente generador (120) también determina (350) datos de verificador para los servidores de verificación (130), preferiblemente al menos dos. Los datos de verificador hacen posible a un servidor de verificación (130) verificar (402, 602) si un dispositivo (220) ha recuperado con posterioridad de manera exitosa (400, 600) los datos secretos fuertes. Sin embargo, es desde el punto de vista computacional no factible para el servidor (130) determinar los datos secretos débiles en base solamente al acceso a sus datos de verificador. Los servidores de verificación (130) pueden almacenar (355) los datos de verificador para su uso posterior. El cliente generador (120) puede adicionalmente usar los datos secretos fuertes como una clave criptográfica en un criptosistema simétrico para encriptar (370) otros datos privados para el usuario (110), tales como la clave privada de usuario.

En una realización preferida, los datos secretos fuertes se calculan (330, 350) de la siguiente manera. El cliente generador (120) calcula los datos de petición del servidor para al menos uno de los siguientes servidores que guardan secretos (130). Los datos se petición del servidor son una función de los datos secretos débiles y de un secreto de cliente efímero, pero los datos de petición del servidor no revelan información acerca de los datos secretos débiles sin el conocimiento del secreto de cliente efímero. Como resultado de esto, el cliente generador (120) puede transmitir los datos de petición del servidor al servidor que conserva secretos (130) sin comprometer los datos secretos débiles. El servidor que conserva secretos (130) calcula los datos de respuesta del servidor que son una función de los datos secretos del servidor para el servidor que conserva secretos y de los datos de petición de servidor recibidos. Los datos de respuesta del servidor no revelan información acerca de los datos secretos del servidor sin el conocimiento de los datos secretos débiles y el secreto de cliente efímero. Como resultado de esto, el servidor que conserva secretos (130) puede transmitir los datos de respuesta del servidor al cliente generador (120) sin comprometer sus datos secretos del servidor. El cliente generador (120) calcula un componente secreto para el servidor que conserva secretos como una función de los datos de respuesta del servidor recibidos desde el servidor que conserva secretos y del secreto de cliente efímero. El componente secreto es una función de los datos secretos débiles y de los datos secretos del servidor pero es independiente del secreto de cliente efímero. El cliente generador (120) calcula entonces los datos secretos fuertes de usuario como una función de los componentes secretos.

En un refinado adicional, los datos secretos débiles son una contraseña *PWD* y los datos secretos del servidor son enteros aleatorios *b(i)*, donde i es un índice para los servidores (130). El cliente generador (120) calcula (534) los datos de petición del servidor que incluyen el valor $M = w^a$. Aquí, w = f(datos secretos débiles), donde f es una función que genera un elemento de un grupo *G*, y el secreto de cliente efímero incluye el entero aleatorio a para el que existe un correspondiente entero a' tal que $x^{aa'} = x$ para todos los x del grupo *G*. El grupo *G* es un grupo finito en el que la exponenciación es eficiente pero el problema del logaritmo discreto es no factible desde el punto de vista computacional, por ejemplo el grupo multiplicativo del conjunto de enteros módulo a prima *p* o un grupo de puntos sobre una curva elíptica sobre un campo finito. Todas las exponenciaciones se calculan en el grupo *G*. El servidor que conserva secretos (130) calcula (534) los datos de respuesta del servidor que incluyen el valor $c(i) = M^{b(i)}$. El cliente generador (120) calcula después (534) componentes secretos de acuerdo con $K(i) = h(c(i)^{a'})$ donde h es una función. Los datos secretos fuertes de usuario se calculan después (536) como una función de todos los componentes secretos K(i), por ejemplo como la operación O-exclusiva de estos componentes.

En otro aspecto de la invención, los datos secretos fuertes se recuperan (400, 600) a partir de los datos secretos débiles de la siguiente manera. Un cliente de recuperación (220) recibe (410) los datos secretos débiles del usuario. El cliente de recuperación (220) calcula entonces (440, 460) los datos secretos fuertes del usuario, que son una función

de los datos secretos débiles del usuario y de los datos secretos del servidor para al menos dos servidores que guardan secretos (130). En una realización preferida, el cálculo se basa en los componentes secretos, como se ha descrito anteriormente. El cliente de recuperación (220) determina también (450, 650) datos de prueba para proporcionar (460, 660) que los datos secretos fuertes se calcularon de manera exitosa (401, 601) y transmite (455, 655) los datos de prueba a los servidores de verificación (130) que pueden o no ser también servidores que guardan secretos. Mediante la validación de los datos de prueba usando los correspondientes datos de verificador, los servidores de verificación (130) pueden determinar (460, 660) si los datos secretos fuertes se recuperaron con éxito (401, 601) y tomar las acciones apropiadas (680). Por ejemplo, parece probable que una entidad (220) que no tenga acceso a los datos secretos débiles esté intentando regenerar (401, 601) los datos secretos fuertes, entonces el servidor de verificación podría dar órdenes a los servidores que guardan secretos (130) para que cesen de participar en cualquier intento adicional de recuperación (400, 600). De manera -alternativa, el servidor de verificación podría ser responsable de generar parte de una firma digital en nombre del usuario (110). En este caso, los datos de prueba podrían estar acompañados por un resumen de mensaje de un mensaje que vaya a ser firmado y el servidor de verificación (130) generará su parte de la firma digital solamente cuando se presente de manera simultánea con los datos de prueba adecuados. Si se encriptaron (370) datos privados adicionales usando los datos secretos fuertes como una clave criptográfica, entonces el cliente de recuperación (220) puede de manera adicional desencriptar (470) los datos privados.

En una realización preferida, los datos secretos fuertes se calculan (440, 640) de la siguiente manera. El cliente de recuperación (220) calcula (420, 620) los datos de petición del servidor para al menos uno y de manera preferible más de un servidor que conserva secretos. Los datos de petición del servidor son una función de los datos secretos débiles de un secreto de cliente efímero, pero no revelan información significativa acerca de los datos secretos débiles sin el conocimiento del secreto de cliente efímero. Los datos de petición del servidor se transmiten (425, 625) al servidor que conserva secretos, que calcula (430, 630) los datos de respuesta del servidor en base a los datos de petición del servidor y a sus datos secretos del servidor. Los datos de respuesta de servidor no revelan información significativa acerca de los datos secretos de servidor sin el conocimiento de los datos de petición de servidor. Los datos de respuesta del servidor se transmiten (435, 635) al cliente de recuperación (220) que recupera (440, 640) los datos secretos fuertes del usuario usando los datos de respuesta del servidor.

En una realización preferida correspondiente a la descrita con anterioridad, el cliente de recuperación (220) genera de manera aleatoria (624) un secreto de cliente efímero a, que es un entero para el que existe un entero correspondiente a' tal que $x^{aa'} = x$ para todo x del grupo G. Calcula (626) los datos de petición del servidor incluyendo el valor $M = w^a$, donde $w = f(PWD)$ corno antes, y transmite (625) estos datos a al menos un servidor que conserva secretos (130). Cada servidor que conserva secretos (130) calcula (630) los datos de respuesta del servidor incluyendo el valor $c(i) = M^{b(i)}$ y transmite (635) esto de vuelta al cliente de recuperación (220). El cliente de recuperación (220) calcula (644) los componentes secretos $K(i) = h(c(i)^{a'})$, donde h es la misma función que antes. Los componentes secretos se combinan después (646), como en el cliente generador (120), para recuperar los datos secretos fuertes.

Estos procedimientos (300, 400, 500, 600) son ventajosos porque permiten a un usuario (110) recuperar (400, 600) datos secretos fuertes a partir de datos secretos débiles en muchos clientes de recuperación (220). Estos procedimientos son en general resistentes a ataques, incluyendo ataques sobre los servidores (130) o que comprometan a los mismos. De acuerdo además con la presente invención, software y/o hardware (100, 200) implementa los procedimientos anteriores.

Breve descripción de los dibujos

Éstos y otros objetos y características más detallados y específicos de la presente invención se describen más completamente en la siguiente especificación, con referencia a los dibujos acompañantes, en los que:

La figura 1 es un diagrama de bloques de un sistema de inicialización (100) de acuerdo con la presente invención;

La figura 2 es un diagrama de bloques de un sistema de recuperación (200) de acuerdo con la presente invención;

La figura 3 es una traza de eventos que ilustra un procedimiento de ejemplo (300) para inicializar el sistema 100, de acuerdo con la presente invención;

La figura 4 es una traza de eventos que ilustra un procedimiento de ejemplo (400) para recuperar datos secretos fuertes que ha sido inicializado usando el procedimiento 300;

La figura 5 es una traza de eventos que ilustra un procedimiento preferido (500) para inicializar el sistema 100 usando exponenciación en un grupo G, de acuerdo con la presente invención;

La figura 6 es una traza de eventos que ilustra un procedimiento preferido (600) para recuperar los datos secretos fuertes que ha sido inicializado usando el procedimiento 500.

Descripción detallada de las realizaciones preferidas

Las figuras 1 y 2 son diagramas de bloques que ilustran los sistemas 100 y 200 de acuerdo con la presente invención. Por razones que serán aparentes con posterioridad, se hará referencia al sistema 100 como un "sistema de inicialización" y al sistema 200 como un "sistema de recuperación".

El sistema de inicialización 100 incluye un usuario 110, un terminal de cliente 120, un número de servidores 130A-130N (de manera colectiva, los servidores 130) y opcionalmente también un medio de almacenamiento 140. El usuario 110 puede ser una persona, un grupo de personas, una entidad legal tal como una corporación, un ordenador, etc. El terminal de cliente 120, al que se hará referencia como el "cliente generador" 120, es típicamente algún tipo de dispositivo basado en ordenador. Ejemplos incluyen ordenadores personales, estaciones de trabajo de ordenador y teléfonos sin hilos digitales. Los servidores 130 de manera típica también son dispositivos basados en ordenador. En esta descripción, se hace referencia a los mismos como "servidores" debido al papel que desempeñan, pero esta terminología no significa que implique que sean necesariamente ordenadores de la clase servidor. Al menos un servidor y posiblemente todos los servidores 130 son servidores que guardan secretos. El papel desempeñado por los servidores que guardan secretos se describirá más completamente con posterioridad. En ciertas realizaciones, puede haber un único servidor 130. Las realizaciones alternativas prefieren dos o más servidores que guardan secretos 130. En realizaciones que utilicen múltiples servidores que guardan secretos 130, los servidores que guardan secretos de manera preferible están controlados por diferentes entidades de forma que ninguna entidad individual tenga acceso a todos los servidores que guardan secretos 130, por razones que se tratan más adelante. Ejemplos de medio de almacenamiento 140 incluyen una unidad de disco de red, un servicio de directorio o un servidor de ficheros. El usuario 110, los servidores 130 y el medio de almacenamiento 140 están cada uno de ellos acoplado al cliente generador 120. Las conexiones se pueden hacer por cualquier número de medios, incluyendo sobre redes de ordenadores tales como Internet o por medio de conexiones sin hilos. Las conexiones necesitan no ser permanentes o constantes. De hecho, como se describe con mayor detalle con posterioridad, el cliente generador 120 realiza una tarea particular y una vez que se ha completado la tarea, no hay necesidad de que los otros componentes comuniquen además con el cliente generador 120.

El sistema de recuperación 200 es similar al sistema de inicialización 100, excepto que el cliente generador 120 se sustituye por otro terminal de cliente 220, al que se hará referencia como el cliente de recuperación 220. El cliente de recuperación 220 puede o no ser el mismo dispositivo físico que el cliente generador 120. Ejemplos de clientes de recuperación 220 incluyen ordenadores personales, kioscos digitales, teléfonos digitales sin hilos u otros dispositivos sin hilos, y tarjetas inteligentes.

Los sistemas 100 y 200 implementan la siguiente funcionalidad. El usuario 110 tiene datos secretos fuertes que le gustaría ser capaz de usar desde el cliente de recuperación 220, donde "fuertes" implica datos que no se pueden deducir de manera factible por medio de la adivinación exhaustiva y "secretos" significa datos que nadie que no sea el poseedor secreto (por ejemplo, el usuario en el caso de los datos secretos fuertes de usuario) debería ser capaz de determinar de manera factible. Sin embargo, el cliente de recuperación 220 es un terminal de cliente que no pueda o que no tenga acceso a priori a los datos secretos fuertes del usuario 110. Además, el usuario 110 no sabe directamente sus datos secretos fuertes así, por ejemplo, el usuario 110 no puede simplemente introducir sus datos secretos fuertes dentro del cliente de recuperación 220. Así, el cliente de recuperación 220 debe de alguna manera regenerar o recuperar los datos secretos fuertes de usuario 110 y debe hacerlo así de una manera segura con el fin de mantener el fuerte secretismo de los datos. El usuario 110 conoce ciertos datos secretos débiles, donde "débiles" implica que los datos se pueden adivinar correctamente con un número moderado de intentos. Por ejemplo, puede ser una contraseña especificada por el usuario. El usuario 110 introduce sus datos secretos débiles dentro del cliente de recuperación 220, y en base a los datos secretos débiles del usuario 110, el sistema 200 recupera de manera segura después los datos secretos fuertes, permitiendo de esa manera al usuario 110 usar los datos secretos fuertes desde el cliente de recuperación 220. El sistema 100 y el cliente generador 120 realizan varios pasos de inicialización en base a los datos secretos fuertes y a los datos secretos débiles, de forma que el sistema 200 y el cliente de recuperación 220 puedan recuperar con posterioridad de manera segura los datos secretos fuertes a partir de los datos secretos débiles del usuario 110. Los servidores 130 ayudan en estos procesos.

En una realización preferida, los datos secretos fuertes son una clave criptográfica que se usa en un criptosistema simétrico para encriptar los datos privados de usuario 110 que podrían incluir una clave privada, y los datos secretos débiles son una contraseña. Los datos privados encriptados se almacenen en un medio de almacenamiento 140. Cuando el usuario 110 desee usar sus datos privados desde el cliente de recuperación 220, él suministra su contraseña al cliente de recuperación 220. El cliente de recuperación 220 recupera entonces la clave criptográfica que se usa para desencriptar los datos privados encriptados de usuario. El usuario 110 puede entonces usar sus datos privados como desee, por ejemplo, mediante el uso de su clave privada para firmar digitalmente mensajes dentro del cliente de recuperación 220.

Las figuras 3 y 4 son trazas de eventos que ilustran procedimientos de ejemplo 300 y 400 para realizar la inicialización del sistema 100 y para recuperar los datos secretos fuertes a partir de los datos secretos débiles del usuario 110, respectivamente. El procedimiento 300 usa el sistema 100 y el procedimiento 400 usa el sistema 200. Cada uno de los recuadros en línea discontinua 110, 120, 130, 140 y 220 representa uno de los componentes del sistema 100 o del sistema 200. Los recuadros en línea continua representan varios pasos en los dos procedimientos 300 y 400. La localización de un recuadro en línea continua dentro de un recuadro en línea discontinua generalmente indica que

el paso específico es realizado por ese componente específico. Por ejemplo, en la figura 3, el paso 310 está localizado dentro del recuadro con línea discontinua para el cliente generador 120. Esto indica por lo general que el cliente generador 120 realiza el paso 310. Los procedimientos de manera preferible se implementan por medio de software que se ejecuta en los distintos componentes dentro de cada sistema, posiblemente con la ayuda de módulos hardware, pero los módulos de procesado representados en las figuras 3 y 4 también se pueden implementar en hardware y/o firmware.

Con referencia primero a la figura 3, el procedimiento de inicialización 300 se puede descomponer en tres etapas generales 301-303. En la etapa de generación 301, el sistema 100 determina los datos secretos fuertes de usuario, a los que se hará referencia en esta realización como *K*. Esta etapa 301 incluye los pasos que permiten al sistema 200 regenerar de manera segura con posterioridad el secreto fuerte *K*. En la etapa de configuración de verificador 302, el sistema 100 ejecuta los pasos que permiten al sistema 200 verificar la posterior recuperación exitosa del secreto fuerte *K*. En la etapa de almacenamiento 303, el sistema 100 usa el secreto fuerte *K* para encriptar los datos privados para el usuario (por ejemplo, la clave privada de usuario o el certificado digital de usuario), para una recuperación posterior por parte del cliente 220 de recuperación. No todas las implementaciones utilizarán todas las etapas 301-303, pero éstas se incluyen en este ejemplo para ilustrar varios aspectos de la invención.

En la etapa de generación 301, el cliente generador 120 comienza por autenticar 310 al usuario 110 como un usuario legítimo para una cuenta de usuario *U*. Esto podría implicar las comunicaciones con otros sistemas, tales como un servidor de autenticación.

El usuario 110 y/o el cliente generador 120 determina 320 los datos secretos débiles del usuario 110, a los que se hace referencia en este documento como *PWD*. El secreto débil es típicamente una contraseña de usuario en esta realización, y esta descripción la caracterizará como tal. Sin embargo, el secreto débil podría tomar otros formatos y/o ser completamente o parcialmente de otra fuente, tal como datos de firma biométrica almacenados en una tarjeta física o un dispositivo asociado con el usuario 110. En la figura 3, la generación 320 del secreto débil se muestra cubriendo tanto al usuario 110 como al cliente generador 120. Esto es porque cualquiera puede participar en varios grados, dependiendo de la implementación específica. Por ejemplo, en una aproximación, el usuario 110 propone una contraseña *PWD* y el cliente generador 120 o acepta o rechaza la contraseña, dependiendo de si cumple ciertos criterios de anti-adivinación. En una aproximación diferente, el cliente generador 120 genera la contraseña *PWD* (por ejemplo, usando un proceso aleatorio) y después la asigna al usuario 110. En cualquier caso, al final del paso 320, tanto el usuario 110 como el cliente generador 120 tienen acceso a la contraseña *PWD*.

Los datos secretos de servidor *b(i)* para el usuario 110 se establecen 340 para algunos y quizá para todos los servidores *S(i)*, donde *i* es un índice para los servidores. Se hace referencia a los servidores 130 para los que se establecen los datos secretos de servidor como servidores de conservan secretos. De manera análoga al paso 320, se muestra el paso 340 cubriendo el cliente generador 120 y los servidores que guardan secretos *S(i)* porque cada uno de ellos puede participar en distinto grado, dependiendo de la implementación específica, como se describirá además más adelante. Si una implementación específica pide al cliente generador 120 y a los servidores que guardan secretos *S(i)* intercambiar mensajes con el fin de establecer 340 los datos secretos de servidor, es importante que esos mensajes estén protegidos en su integridad y la fuente de cada mensaje esté autenticada con el fin de mantener la seguridad del protocolo completo. Por ejemplo, el cliente generador y los servidores que guardan secretos *S(i)* podrían intercambiar mensajes sobre un canal seguro. Al final del paso 340, cada servidor que conserva secretos *S(i)* tiene acceso a sus correspondientes datos secretos de servidor *b(i)* y de manera típica serán almacenados de manera segura 345 para su uso futuro en la recuperación de los datos secretos fuertes de usuario. El cliente generador 120 puede tener acceso también a los datos secretos de servidor *b(i)*, dependiendo de la implementación específica. Sin embargo, en este caso, de manera típica usaría los datos secretos de servidor *b(i)* solamente como parte de la inicialización 300 y después los borraría. El cliente generador 120 no retiene los datos secretos de, servidor *b(i)* para uso futuro.

El cliente generador 120 calcula 330 los datos secretos fuertes de usuario *K*. Los datos secretos fuertes *K* son una función del secreto débil de usuario *PWD* y de los datos secretos de servidor *b(i)*. De nuevo, el paso 330 cubre tanto el cliente generador 120 como los servidores *S(i)* porque, dependiendo de la implementación, cada uno de ellos contribuye al cálculo requerido para convertir el secreto débil de usuario *PWD* y los datos secretos del servidor *b(i)* en el secreto fuerte *K*. Sin embargo, aunque los servidores que guardan secretos *S(i)* pueden calcular valores intermedios, solamente el cliente generador 120 tiene acceso al secreto fuerte final *K*.

En la etapa de configuración de verificador 302, se determinan 350 los datos de verificador *v(i)* para algunos y quizá para todos los servidores *S(i)* y de manera preferible también son almacenados 355 por cada uno de los servidores *S(i)*. Se hará referencia a los servidores para los que se establecen los datos de verificador como los servidores de verificación 130. Los datos de verificador *v(i)* habilitan a los servidores de verificación *S(i)* para verificar si un cliente de recuperación 220 ha recuperado con éxito los datos secretos fuertes del usuario. En una realización preferida, los servidores que guardan secretos son también los servidores de verificación, aunque éste no es necesariamente el caso. De manera alternativa, los servidores de verificación pueden ser físicamente distintos a los servidores que guardan secretos, pero puede que haya un servidor de verificación correspondiente a cada uno de los servidores que guardan secretos. Por propósitos de redundancia, es preferible tener al menos dos servidores de verificación. Los datos de verificador se seleccionan de forma que sea no factible desde el punto de vista computacional que el servidor de verificación determine los datos secretos débiles en base solamente al acceso a sus datos de verificador. En una aproximación, el cliente generador 120 determina 350 los datos de verificador que se transmiten después al servidor

130. En una aproximación alternativa, cada servidor 130 determina sus propios datos de verificador. De manera análoga al paso 340, si una implementación específica pide al cliente generador 120 y al servidor 130 intercambiar mensajes con el fin de determinar 350 los datos de verificador, es importante que estos mensajes estén protegidos en su integridad y la fuente de cada mensaje esté autenticada.

En la etapa de almacenamiento 303, el cliente generador 120 encripta de manera adicional 370 otros datos para el usuario, a los que se hará referencia como los datos privados del usuario. En una realización preferida, los datos secretos fuertes *K* se usan como una clave criptográfica en un criptosistema simétrico. Por ejemplo, los datos privados podrían ser la clave privada del usuario, un secreto compartido por el usuario y por un servidor de aplicación, los números de cuenta de la tarjeta de crédito del usuario u otros datos privados o datos secretos que el usuario le gustaría usar desde el cliente de recuperación 220. Los datos privados encriptados *EPD* se almacenan 375 en el medio de almacenamiento 140 para una posterior recuperación. El medio de almacenamiento 140 de manera típica es ampliamente accesible; los datos privados de usuario están asegurados porque se almacenan en formato encriptado. En realizaciones alternativas, los datos secretos fuertes *K* se pueden usar de otras maneras para almacenar de manera segura los datos privados de usuario.

Haciendo referencia ahora a la figura 4, el procedimiento de recuperación 400 también se puede descomponer en tres etapas generales 401-403 correspondientes a las etapas 301-303 del procedimiento 300, no todas las cuales se requieren en todas las implementaciones. En la etapa 401, el cliente de recuperación 220 recupera el secreto fuerte de usuario *K*, con la ayuda de los servidores que guardan secretos 130. En la etapa 402, uno o más servidores de verificación 130 determinan si el cliente de recuperación 220 ha recuperado con éxito los datos secretos fuertes *K*. En la etapa 403, el cliente de recuperación 220 recupera también los datos privados de usuario almacenados en el medio de almacenamiento 140. Una vez más, se selecciona el procedimiento de ejemplo 400 con el fin de ilustrar varios aspectos de la invención, no todos los cuales se requieren para poner en práctica la invención.

El cliente de recuperación 220 recupera 401 los datos secretos fuertes de usuario 110 de la siguiente manera. El usuario 110 introduce 410 su identificador de cuenta de usuario *U* y los datos secretos débiles *PWD* en el cliente de recuperación 220. El cliente de recuperación 220 calcula 420 los datos de petición del servidor para cada uno, de los servidores que guardan secretos *S(i)* y transmite 425 los datos de petición del servidor a los servidores que guardan secretos. Los datos de petición del servidor son una función de los datos secretos débiles *PWD* y un secreto de cliente efímero *a*, tal como la salida de la función no revela información acerca del secreto débil a una parte que no conozca el secreto de cliente efímero *a*. En respuesta a los datos de petición de servidor recibidos, cada servidor que conserva secretos *S(i)* calcula 430 los datos de respuesta del servidor, que son una función de los datos de petición del servidor y de los datos secretos del servidor *b(i)*, y transmite 435 los datos de respuesta del servidor de vuelta al cliente de recuperación 220. El cliente de recuperación 220 calcula después 440 los datos secretos fuertes del usuario *K* como una función de los datos de respuesta del servidor recibidos. Como se ha descrito con anterioridad, los datos secretos fuertes son una función de los datos secretos débiles del usuario y de los datos secretos del servidor. El cliente de recuperación 220 tiene acceso directo a los datos secretos débiles de usuario pero no tiene acceso directo a los datos secretos del servidor. Sin embargo, los datos de respuesta del servidor contienen una dependencia de los datos secretos del servidor; de esta manera, el cliente de recuperación 220 tiene un acceso indirecto a los datos secretos de servidor y puede recuperar los datos secretos fuertes del usuario sin requerir el acceso directo a los datos secretos del servidor.

En la etapa de recuperación 403, el cliente de recuperación 220 recupera 475 los datos privados encriptados de usuario *EPD* y los desencripta 470 usando los datos secretos fuertes recuperados *K*. De esta manera, el cliente de recuperación 220 también recupera los datos privados de usuario, tales como la clave privada de usuario.

Al final del período de uso legítimo del secreto fuerte *K* y cualquier otro dato privado recuperado (por ejemplo, al final de la sesión en línea del usuario usando el cliente de recuperación 220), la copia de *K* y otros datos privados recuperados en el cliente de recuperación 220 son de manera preferible destruidos.

En la etapa de verificación 402, el cliente de recuperación 220 determina 450 los datos de prueba *d(i)* para probar al menos un servidor de verificación (de manera preferible a al menos dos servidores de verificación) que los datos secretos fuertes se recuperaron con éxito por parte del cliente de recuperación 220. Los datos de prueba *d(i)* se transmiten 455 a cada uno de los servidores de verificación. Cada servidor de verificación puede entonces verificar 460 si esta abstracción particular del proceso de regeneración 400 recuperó con éxito los datos secretos fuertes del usuario *K* y puede tomar entonces acciones apropiadas. Por ejemplo, en una realización, un servidor de verificación 130 podría ser responsable de una parte del proceso de generación de una firma digital en nombre del usuario 110. En este caso, los datos de prueba se podrían acompañar por un resumen de mensaje del mensaje orientado a usuario para que sea firmado digitalmente. Al producirse la verificación de los datos de prueba, el servidor de verificación 130 genera su componente de la firma digital y transmite este componente de vuelta al cliente. El cliente determina la firma digital completa en base a los componentes que recibe.

Como otro ejemplo, el servidor de verificación también puede ser un servidor que conserva secretos. Este servidor podría determinar, en base a la historia de pasados intentos sin éxito, que una entidad que no conoce los datos secretos débiles está intentando regenerar los datos secretos fuertes. De acuerdo con esto, el servidor que conserva secretos puede rechazar el participar en intentos de recuperación adicionales de recuperación o tomar otras acciones. En una aproximación, los servidores que guardan secretos conservan un seguimiento de todos los intentos para regenerar los datos secretos fuertes *K* para cada cuenta de usuario, y en el caso de excesivos intentos fallidos para cualquier cuenta,

estrangulará y/o bloqueará intentos adicionales de regeneración sobre esa cuenta hasta que se cambie la contraseña o el administrador local determine que los intentos fallidos no representan una amenaza para la cuenta.

Se pueden usar diferentes tipos de verificación. En una aproximación que use datos de verificador estáticos, los datos de verificador *v(i)* son una función unidireccional de uno o más elementos de datos que incluyen los datos secretos fuertes *K*. Los datos de verificador son calculados por el cliente generador 120 o alguna otra parte de confianza y se envían y se almacenan en cada servidor de verificación como parte del proceso de inicialización 300. De manera preferible, se calculan datos de verificador diferentes para cada servidor de verificación 130, por ejemplo, mediante la aplicación de una función de cálculo de clave a la cadena de datos que comprende los datos secretos fuertes *K* concatenada con un único pero no secreto identificador para el servidor particular 130. En la etapa de verificación 402, el cliente de recuperación 220 calcula 450 los datos de prueba mediante el recálculo de la misma función unidireccional de los datos secretos fuertes regenerados. Si los datos de prueba calculados por el cliente de recuperación 220 coinciden con los datos del verificador almacenados por el servidor, entonces se verifica la recuperación de los datos secretos fuertes.

En una variante de esta aproximación, el cliente de recuperación 220 recalcula primero datos como una función unidireccional del secreto fuerte *K* y después calcula los datos de prueba como una función unidireccional de uno o más elementos de datos que incluyen los datos de verificador recalculados y un número aleatorio de un solo uso no secreto, tal como una consigna de hora o número aleatorio de un solo uso enviado previamente desde el servidor al cliente de recuperación o junto con los datos de respuesta del servidor (por ejemplo, en el caso de un servidor que conserve secretos) o en un mensaje separado. El servidor de verificación 130 calcula sus propios datos de prueba aplicando la misma función unidireccional a su copia de los datos de verificador y al número aleatorio de un solo uso. Si los datos de prueba calculados por el cliente de recuperación 220 coinciden con los datos de prueba calculados por el servidor de verificación 130, entonces se verifica la recuperación de los datos secretos fuertes. El número aleatorio de un solo uso permite al servidor confirmar que los datos de prueba son recientes y no son una repetición de datos de prueba anteriores para el mismo usuario. En otras palabras, el número aleatorio de un solo uso distingue los datos de prueba actuales de otros casos de datos de prueba para el mismo usuario.

En una aproximación de verificación diferente, asúmase que los datos privados de usuario que se almacenan en un medio de almacenamiento 140 incluyen datos privados (por ejemplo, una clave privada) de un sistema de prueba asimétrico, para los que existen correspondientes datos públicos. El sistema de prueba asimétrico tiene la propiedad de que una entidad que tenga acceso a los datos privados puede comprobar que accede a los datos privados a una segunda entidad con acceso a los datos públicos sin revelar los datos privados a la segunda entidad. El sistema asimétrico de prueba podría ser un sistema de firma digital, tal como RSA o DSA, un sistema de prueba de conocimiento cero en el que la posesión de información se puede verificar sin revelar ninguna parte de esa información, o cualquier otro tipo de sistema asimétrico de prueba. Para los propósitos de describir la invención, se supondrá un sistema de firma digital. En este caso, el cliente de recuperación 220 puede generar datos de prueba que dependan de los datos privados desencriptados, proporcionando de esta manera la recuperación con éxito de los datos privados y, así, también la recuperación con éxito de los datos secretos fuertes (ya que los datos secretos fuertes se necesitan para desencriptar con éxito los datos privados).

Por ejemplo, si los datos privados son la clave de la firma digital privada del usuario, los datos de prueba podrían comprender un mensaje, de manera preferible conteniendo un número aleatorio de un solo uso para permitir la detección de repeticiones, que fueron firmadas de manera digital usando la clave privada. Si los datos de verificador son la correspondiente clave pública, se podrían usar para verificar después con éxito la recuperación de la clave privada. La verificación 460 de la recuperación con éxito de los datos secretos fuertes incluiría entonces la verificación de que se usó la clave privada para firmar digitalmente el mensaje. Si se usa un número aleatorio de un solo uso, entonces la verificación de lo recientes que son los datos de prueba implicaría la verificación de que el mensaje firmado digitalmente contiene el número aleatorio de un solo uso. Como una alternativa, los datos de prueba pueden ser una función de otros datos de usuario que el servidor de verificación pueda autenticar como con origen en el usuario. Otras variaciones del paso de verificación se pueden basar en otras aproximaciones, por ejemplo, en el uso de pruebas de conocimiento cero (por ejemplo, véase J. Nechvatal, "Criptografía de Clave Pública", en G. J. Simmons (Ed.), *Criptología contemporánea: La Ciencia de la Integridad de la Información* (Nueva York: IEEE Press, 1992), páginas 107 a 126).

Otras variaciones de los procedimientos: 300 y 400 serán aparentes. Sin embargo, con el fin de mantener la resistencia contra ataques, incluyendo el comprometer a los servidores, cualquier protocolo con éxito preferiblemente debería incluir los siguientes atributos. En primer lugar, la observación de cualquiera o de todos los mensajes por parte de un oyente oculto porta suficiente información para el oyente oculto para deducir de una manera factible los datos secretos débiles *PWD* o los datos secretos fuertes *K*. En segundo lugar, el conocimiento de cualquier cosa que sea menos que todos los datos secretos del servidor a partir de un número predeterminado de servidores que guardan secretos no permitirá a ninguna parte deducir de una manera factible ni los datos secretos débiles *PWD* ni los datos secretos fuertes *K*. "Cualquier parte" incluye a los propios servidores. Esto es, un servidor no puede deducir de una manera factible ni los datos secretos débiles *PWD* ni los datos secretos fuertes *K* a menos que un número predeterminado de servidores que guardan secretos interactúen mediante la revelación de sus datos secretos de servidor o fallando en el estrangulamiento o en el bloqueo de la cuenta en el caso de excesivos fallos de intento en la ejecución del protocolo. Como resultado de esto, los procedimientos 300 y 400 son ventajosos porque son resistentes a muchos tipos de ataques, incluyendo el poner en compromiso a un servidor, cuando los datos secretos fuertes dependen de los datos secretos del servidor de al menos dos servidores que guardan secretos.

Las figuras 5 y 6 son trazas de eventos que ilustran realizaciones preferidas 500 y 600 de los procedimientos 300 y 400 respectivamente. Las realizaciones preferidas se basan en cálculos dentro de un grupo finito G en el que la exponenciación es eficiente pero el problema del logaritmo discreto es no factible desde el punto de vista computacional. Un grupo adecuado G es el grupo multiplicativo del conjunto de enteros módulo a prima *p*, donde *p* es una prima grande con propiedades que la hacen adecuada como un módulo Diffie-Hellman. En particular, en un mínimo *p-1* debe tener un factor prima grande. Una mejor restricción sobre *p* es que será una prima segura $p = 2q + 1$, donde *q* es prima. También son adecuados otros grupos cíclicos incluyendo un grupo de puntos sobre una curva elíptica sobre un campo finito. Los procedimientos se ilustrarán en el contexto en el que se usarán con posterioridad los datos secretos fuertes *K* como una clave criptográfica usada para encriptar la clave privada de usuario pero, como se ha tratado con anterioridad con los procedimientos 300 y 400, los procedimientos preferidos 500 y 600 no están limitados tampoco a esta aplicación particular. En este ejemplo en particular, cada uno de los servidores 130 funciona tanto como un servidor que conserva secretos como un servidor de verificación.

Con referencia en primer lugar a la figura 5, el procedimiento de inicialización 500 se puede descomponer en tres etapas generales 501-503, análogas a las etapas 301-303 del procedimiento 300. La etapa de generación 501 comienza lo mismo como la etapa de generación 301. El cliente de generador 120 autentica 310 al usuario como un usuario legítimo para la cuenta *U*. A continuación, se determinan 320 los datos secretos débiles del usuario *PWD*.

Los datos secretos de servidor *b(i)* para el usuario 110 se establecen 340 para cada servidor que conserva secretos *S(i)*, donde *i* es un índice para los servidores 130. En esta realización, los datos secretos de servidor son un entero aleatorio *b(i)*, de manera preferible un número par para evitar contra ataques de subgrupo pequeño que son bien conocidos en los procedimientos Diffie-Hellman. Al final del paso 340, cada servidor que conserva secretos tiene acceso a sus datos secretos de servidor *b(i)*; el cliente generador 120 puede o no tener acceso a los datos secretos de servidor *b(i)*, dependiendo de la implementación específica. En una aproximación en la que tanto un servidor que conserva secretos *S(i)* como el cliente generador 120 tengan acceso a los datos secretos de servidor de ese servidor *b(i)*, los datos secretos de servidor *b(i)* son generados o por el cliente generador 120 o por el servidor *S(i)* y comunicados a la otra parte en un formato encriptado. De manera alternativa, los datos secretos de servidor se calculan por medio de la combinación de valores aleatorios provenientes tanto del cliente generador 120 como del servidor que conserva secretos *S(i)*, por ejemplo, por medio del intercambio de la clave Diffie-Hellman. Las comunicaciones están protegidas de manera preferible de forma que ninguna otra parte pueda aprender el *b(i)*. Por otra parte, si el cliente generador 120 no tiene acceso a los datos secretos de servidor *b(i)*, entonces cada servidor que conserva secretos *S(i)* podría generar sus datos secretos de servidor *b(i)* pero no compartirlos con el cliente generador 120 o con los otros servidores *S(i)*. Cualquiera que sea el procedimiento de generación, cada servidor típicamente almacenará de manera segura 345 sus datos secretos de servidor para su uso futuro en la regeneración de los datos secretos fuertes de usuario.

El cliente generador 120 calcula 530 los datos secretos fuertes *K* de la siguiente manera. En primer lugar, el cliente generador 120 calcula 532 $w = f(PWD)$, donde *f* es una función unidireccional que genera un elemento de grupo G. A continuación, se calcula el componente secreto *K(i)* 534 para cada servidor que conserva secretos de acuerdo con la relación $K(i) = h(w^{b(i)})$. En esta expresión, *h* es una función unidireccional, tal como una función de cálculo de clave criptográfica, que genera un valor con bits adecuadamente no distorsionados y sin correlar, como podría ser adecuado para el uso con una clave de encriptado simétrica. La exponenciación $w^{b(i)}$ se calcula en el grupo *G* (es decir, enteros módulo *p* en este ejemplo). Dependiendo de la implementación, los servidores que guardan secretos pueden o no participar en este cálculo. Nótese que cada componente secreto es una función tanto de los datos secretos débiles de usuario como de los datos secretos fuertes de usuario para el correspondiente servidor que conserva secretos.

Por ejemplo, si el cliente generador 120 tiene un conocimiento directo de los datos secretos del servidor *b(i)*, entonces puede calcular directamente 534 los componentes secretos $K(i) = h(w^{b(i)})$. De manera alternativa, si el cliente generador 120 no tiene acceso a los datos secretos del servidor *b(i)*, entonces se puede usar el siguiente procedimiento de transacción protegida. El cliente generador 120 selecciona un secreto efímero de cliente que es un entero aleatorio a para el que existe un correspondiente entero *a'* de forma que $x^{aa'} = x$ para todo *x* del grupo *G*. Por ejemplo, si *G* es el grupo de enteros módulo *p, a* se puede seleccionar para que sea un elemento del grupo multiplicativo de enteros módulo *p-1* y *a'* sería entonces su inverso multiplicativo. El cliente generador calcula y envía un mensaje $M_1 = w^a$ a cada servidor que conserva secretos *S(i)*. El servidor que conserva secretos *S(i)* calcula $c(i) = M_1^{b(i)} = w^{ab(i)}$ y envía *c(i)* al cliente generador 120. El cliente generador 120 calcula el valor *a'* correspondiente a a y después calcula 534 el componente secreto $K(i) = h(c(i)^{a'}) = h(w^{b(i)})$. Esta aproximación asegura que no se exponen datos secretos a un oyente oculto, haciendo uso de las propiedades de encriptado de la exponenciación discreta.

Una vez que el cliente generador 120 haya calculado 534 los componentes secretos *K(i)*, calcula después 538 el secreto fuerte *K* como una función de los componentes secretos *K(i)* de los servidores que guardan secretos participantes *S(i)*. En este ejemplo, el secreto fuerte *K* se calcula 536 de acuerdo con una operación O- exclusiva a nivel de bit, $K = K(1) \oplus K(2) \oplus \ldots \oplus K(N)$ donde O denota la operación O- exclusiva. El procedimiento de suma binaria y los procedimientos de compartir secreto umbral *t* de *N* son otros dos procedimientos para combinar los componentes secretos *K(i)*. En un procedimiento de compartir secreto umbral t de N, existen N servidores que guardan secretos pero los datos secretos fuertes se pueden calcular a partir de la recuperación de datos de solamente t de ellos, donde t es menor que N. Otros procedimientos serán obvios.

En la etapa de almacenamiento 503, los datos secretos fuertes *K* se usan como una clave criptográfica en un criptosistema simétrico para encriptar 370 unos datos privados de usuario incluyendo la clave privada $Priv_U$. Los datos

privados encriptados, denotados por *EPD*, que incluyen a la clave privada encriptada, denotada por $E_K(Priv_U)$, donde E_K significa encriptado con clave *K*, se almacenan 375 en el medio de almacenamiento 140.

En la etapa de configuración del verificador 502, la clave pública, Pub_U, correspondiente a la clave privada de usuario, $Priv_U$ se podría usar como los datos de verificador *v(i)*. En esta realización, cada servidor que conserva secretos también desempeña el papel de un servidor de verificación *S(i)* y almacena 355 sus datos de verificador $v(i) = Pub_U$ o al menos tiene acceso a la clave pública. En una realización alternativa, los datos de verificador $v(i) = h(K, ld(i))$, donde *ld(i)* es un único identificador pero un identificador públicamente conocido para el servidor *S(i)* y *h* es una función unidireccional tal como una función de cálculo de clave.

Con referencia ahora a la figura 6, el proceso de recuperación 600 incluye también tres etapas 601-603. En la etapa 601, el cliente de recuperación 220 recupera el secreto fuerte de usuario *K* en base a su secreto débil *PWD*, con la ayuda de los servidores que guardan secretos. En la etapa de verificación 602, el cliente de recuperación 220 demuestra a los servidores de verificación que ha recuperado con éxito el secreto fuerte *K*. En la etapa 603, el cliente de recuperación 220 recupera la clave privada de usuario, $Priv_U$.

La recuperación 601 del secreto fuerte K comienza con el cliente de recuperación 220 que recibe 410 el identificador de cuenta de usuario *U* y una contraseña *PWD* del usuario 110. El cliente de recuperación 220 genera entonces los componentes secretos necesarios K(i) usando el procedimiento de transacción protegido anteriormente descrito. En particular, el cliente de recuperación 220 calcula 622 $w = f(PWD)$, donde *f* es la misma función unidireccional usada en la etapa de generación 500. El cliente de recuperación 220 selecciona 624 un secreto de cliente efímero que es un entero aleatorio a para el que existe un correspondiente entero *a'* tal que $x^{aa'} = x$ para todo *x* del grupo *G*. Por ejemplo, si *G* es el grupo de entero módulo *p*, a se puede seleccionar para que sea un elemento del grupo multiplicativo de enteros módulo *p-1* y *a'* sería entonces su inverso multiplicativo. El cliente de recuperación calcula después 626 los datos de petición de servidor $M_1 = w^a$ y transmite 625 estos datos de petición de servidor al servidor *S(i)*. Nótese que los datos de petición de servidor M_1 son una función tanto de los datos secretos débiles *PWD* como del secreto de cliente efímero *a*. Sin embargo, los datos de petición de servidor M_1 no revelan información acerca de los datos secretos débiles *PWD* sin el conocimiento del secreto de cliente efímero *a*.

El servidor *S(i)* recibe los datos de petición de servidor M_1. El servidor incrementa un contador de intentos de recuperación no verificada para la cuenta de usuario *U* y la contraseña actual *PWD* y determina 680 si es probable que una parte sin acceso a la contraseña esté intentando regenerar los datos secretos fuertes. En esta realización, lo hará de esta manera mediante la determinación de si el número de intentos de recuperación no verificada sobrepasa un umbral. En el caso de que lo sobrepase, entonces el servidor inhabilita la cuenta de usuario *U* y aborta el proceso de recuperación 600. Dependiendo de las propiedades del grupo *G*, el servidor puede verificar también que los datos de petición del servidor M_1 satisfacen las propiedades de fuerza necesarias. Si los datos de petición del servidor M_1 no tienen las propiedades de fuerza requisito, entonces el servidor aborta el proceso de recuperación. Si el proceso de recuperación no se ha abortado, el servidor calcula 630 los datos de respuesta del servidor $c(i) = {M_1}^{b(i)} = w^{ab(i)}$ y envía 635 *c(i)* al cliente de recuperación 220. El servidor genera también 690 un único índice *n(i)*, o número aleatorio de un solo uso, para la ejemplificación del proceso de recuperación y transmite el número aleatorio de un solo uso al cliente de recuperación 220. El servidor fija un estado variable indicando la verificación pendiente de número aleatorio de un solo uso *n(i)*. En una aproximación preferida, el servidor transmite al cliente de recuperación 220 un único mensaje que se basa tanto en los datos de respuesta del servidor *c(i)* como en el número aleatorio de un solo uso *n(i)*. De manera similar a los datos de petición del servidor M_1, los datos de respuesta del servidor *c(i)* son una función de los datos secretos del servidor *b(i)* para el servidor que conserva secretos y de los datos de petición del servidor M_1 recibidos. Sin embargo, los datos de respuesta del servidor *c(i)* no revelan información acerca de los datos secretos del servidor *b(i)* sin el conocimiento de los datos de petición del servidor M_1, o algo equivalentemente, de los datos secretos débiles *PWD* y del secreto de cliente efímero *a*.

Al recibirse el mensaje desde el servidor, el cliente de recuperación 220 puede abortar el proceso de recuperación 600 si el mensaje recibido no tiene una fuerza de requisito. En caso contrario, el cliente de recuperación 220 calcula 642 el valor *a'* que corresponde con *a*. Después calcula 644 el componente secreto $K(i) = h(c(i)^{a'}) = h(w^{b(i)})$. Nótese que el uso de un secreto de cliente efímero a hace que las comunicaciones entre el cliente de recuperación 220 y el servidor que conserva secretos 130 sean resistente a ataques destinados a reducir los datos secretos débiles *PWD* o los datos secretos del servidor *b(i)*. Sin embargo, el componente secreto *K(i)* es una función tanto de los datos secretos débiles *PWD* como de los datos secretos de servidor *b(i)*, pero son independientes del secreto de cliente efímero *a*. Finalmente, el cliente de recuperación 220 calcula 646 los datos secretos fuertes $K = K(1) \oplus K(2) \oplus ... \oplus K(N)$. El cliente de recuperación 220 puede recuperar entonces la clave privada de usuario $Priv_U$ mediante la recuperación 475 y el desencriptado 470 *EPD* usando la clave criptográfica recuperada *K*.

Como se ha mencionado con anterioridad, se pueden usar diferentes aproximaciones de verificación. Por ejemplo, supóngase que se usa la clave pública de usuario Pub_U como los datos de verificador *v(i)*. Entonces, el cliente de recuperación 220 puede regenerar 650 datos de prueba mediante la firma digital de un mensaje que contenga los distintos números aleatorios de un solo uso *n(i)* usando la clave privada recuperada de usuario $Priv_U$. Cada servidor de verificación verifica 660 la recuperación con éxito de los datos secretos fuertes *K* mediante la verificación de la firma digital usando la clave pública de usuario Pub_U y después verificando que el número aleatorio de un solo uso correcto *n(i)* está incluido en el mensaje. Por otra parte, supóngase que los datos de verificador $v(i) = h(K, ld(i))$. Entonces, los datos de prueba se pueden calcular de acuerdo con la expresión $g(v(i), n(i))$, donde *g* es una función unidireccional

tal como una función de cálculo de clave criptográfica. El servidor de verificación verifica 660 los datos de prueba mediante el cálculo de su propio valor desde su propio conocimiento de *v(i)* y *n(i)*, y comparando el resultado con el valor recibido.

Al producirse la recepción de los datos de prueba, cada servidor determina si la variable de estado indica la verificación pendiente del número aleatorio de un solo uso *n(i)*. Si la verificación está pendiente, entonces el servidor verifica que los datos de prueba recibidos demuestran con éxito el conocimiento del secreto fuerte *K* y de la originalidad unida al número aleatorio de un solo uso *n(i)*. Si se verifican ambos, entonces se decrementan el contador de intentos de recuperación no verificada para la cuenta de usuario *U* y la contraseña *PWD*. En caso contrario, el proceso de recuperación se considera que no ha tenido éxito.

Los procedimientos 300, 400, 500 y 600 son particularmente ventajosos porque son resistentes a muchos tipos de ataques, incluyendo ataques por los servidores o sobre los servidores *S(i)*. Lo siguiente son algunos tipos de ataques y contramedidas de ejemplo.

Un atacante puede intentar adivinar o en cualquier otro caso comprometer la contraseña *PWD* de usuario 110. Esto se puede combatir poniendo requisitos acerca de la elección de contraseña *PWD* con el fin de reducir la oportunidad de que un atacante adivinará la contraseña. Por ejemplo, se puede requerir que la contraseña tenga una longitud mínima o se cambie de manera periódica. En otra contramedida, se selecciona un umbral de intentos que limite el número de intentos de adivinación en la contraseña *PWD*. De esta manera, un atacante solamente tiene un número limitado de intentos para adivinar la contraseña *PWD*. Si se sobrepasa un umbral de intentos, el proceso de generación 500 de manera preferible se vuelve a ejecutar, requiriendo una nueva contraseña *PWD* y la generación de un nuevo secreto fuerte *K*, frustrando de esta manera al atacante. El umbral de intentos se debería fijar de manera preferible de forma que, dados los requisitos establecidos acerca de la contraseña, la probabilidad de un ataque de adivinación con éxito sea aceptablemente pequeña a la vez que aún se equilibra la realidad de que algunos intentos no exitosos pueden ser legítimos más que derivados de un ataque. Por ejemplo, un intento sin éxito puede ser registrado de manera legítima en ausencia de cualquier ataque, si el usuario 110 teclea de manera incorrecta su contraseña *PWD* y/o como resultado de un fallo de la comunicación o fallo del sistema.

De manera alternativa, un atacante podría intentar poner en compromiso el protocolo calculando *w*, *a* y/o *b(i)* a partir de los mensajes transmitidos entre varios componentes. Sin embargo, este ataque puede ser frustrado mediante la selección del grupo *G* para que sea resistente a ataques logaritmo discretos. Por ejemplo, cuando *G* en el conjunto de enteros módulo a prima *p*, entonces *p* se selecciona para que sea un módulo Diffie-Hellman lo suficientemente fuerte (por ejemplo, una prima segura) de acuerdo con reglas bien conocidas. Entonces, dichos ataques no serán factibles debido al problema de logaritmo discreto. Esto también es cierto para un servidor atacante. No es factible para un servidor el calcular *w, a,* y/o cualquier *b(i)* distintos a los suyos propios. La selección de una prima fuerte *p*' también da como resultado un secreto fuerte *K*, haciendo así que no sea factible atacar directamente texto cifrado encriptado bajo el secreto fuerte *K*.

En otro tipo de ataque, el atacante podría hacerse pasar como un cliente de recuperación y enviar datos de petición de servidor débiles M_1 al servidor *S(i)*, con la intención de obtener alguna información acerca de los datos secretos de servidor *b(i)* cuando el servidor devuelva los datos de respuesta de servidor $c(i) = M_1^{b(i)}$. El principal ataque conocido de este tipo se refiere al ataque de subgrupo pequeño sobre criptosistemas Diffie-Hellman. Esto se puede advertir por varios medios, incluyendo la selección inicial de prima *p* con propiedades adecuadas, tales como una prima segura de la forma $p = 2q + 1$, donde *q* es una prima, y haciendo *b(i)* siempre un número par. Dependiendo de las propiedades del tipo particular de grupo *G*, el servidor *S(i)* puede comprobar también la fuerza de los M_1 recibidos y rechazar responder si los reconoce como un valor débil.

Un atacante podría intentar cerrar definitivamente o forzar un cambio de contraseña en una cuenta de usuario o en múltiples cuentas soportadas por un servidor *S(i)* enviando datos de petición de servidor falsos repetidos M_1, o interrumpiendo protocolos, por ejemplo, mediante la interceptación y el descarte de mensajes que lleven datos de respuesta del servidor o datos de prueba. El objetivo de este ataque es provocar que el servidor inhabilite una o más cuentas de usuario. El impacto de este ataque se puede reducir teniendo servidores *S(i)* que "estrangulen" el procesamiento de peticiones M_1 repetidas para una cuenta de usuario. Esto es, el servidor *S(i)* suspende o retrasa el procesado para una cuenta de usuario en vista de múltiples intentos no exitosos durante algún período de tiempo.

De manera alternativa, un atacante podría intentar sobrecargar un servidor *S(i)* mediante el envío de números masivos de M_1 falsos al servidor, provocando de esta manera que realice números masivos de exponenciaciones inútiles pero intensivas desde el punto de vista computacional. Hasta esta extensión, dichos intentos son para la misma cuenta de usuario, el estrangulamiento (como se ha tratado con anterioridad) combatirá de manera significativa este ataque. De esta manera, el ataque es viable principalmente si se usan números grandes de diferentes cuentas de usuarios. Así, puede resistir haciendo difícil adivinar un identificador de cuenta de usuario válido *U*, por ejemplo, evitando que identificadores de cuenta de usuario sean sacados de un único número de secuencia.

Como un ejemplo final, un fallo de comunicaciones o un fallo del sistema en un servidor podría causar múltiples fallos de protocolo de recuperación en los otros servidores, dando como resultado la inhabilitación innecesaria y el cambio forzado de contraseña de una o más cuentas de usuario. Para reducir este problema, los servidores de manera preferible deberían estar diseñados para una alta disponibilidad, con redundancia apropiada de sistemas y de caminos

de las comunicaciones. Además, si un servidor falla de manera inevitable, los controles de gestión pueden suspender de manera temporal el funcionamiento de otros servidores. El estrangulamiento (como se ha tratado con anterioridad) también reducirá la incidencia de la inhabilitación de la cuenta.

La descripción anterior está incluida para ilustrar el funcionamiento de las realizaciones preferidas y no es significa que limite el alcance de la invención. A partir de la anterior discusión, muchas variaciones serán aparentes para alguien que sea experto en la técnica que serían abarcadas por el alcance de la presente invención. Por ejemplo, se puede requerir la presencia de una tarjeta hardware con el fin de completar el proceso de recuperación 400. En una aproximación, el cliente de recuperación 220 determina si la tarjeta hardware del usuario está presente, y si es así, el cliente de recuperación 220 transmite entonces datos que dan fe o que prueban este hecho a cada uno de los servidores que guardan secretos. Los servidores, a su vez, participan en el proceso de regeneración solamente si reciben estos datos desde el cliente de recuperación 220. Cuando se usen junto con una tarjeta hardware, tal como una tarjeta de respuesta a un reto o un generador de contraseña sincronizado en el tiempo una vez, de tal manera que el cliente de recuperación 220 pruebe adicionalmente la posesión de la tarjeta a uno o a más servidores, la combinación resultante de procedimientos puede servir de manera efectiva en lugar de un procedimiento de tarjeta inteligente en el que se recupere un secreto fuerte de la tarjeta inteligente en respuesta a la presentación de un secreto débil de usuario tal como un PIN. La ventaja de la aproximación primera sobre la aproximación de tarjeta inteligente es una posible reducción mayor en el despliegue y en los costes de mantenimiento ya que los tipos anteriormente mencionados de tarjetas no requieren interfaces hardware especiales, como requieren las tarjetas inteligentes. De esta forma, el alcance de la invención está limitado solamente por las siguientes reivindicaciones.

REIVINDICACIONES

1. Un procedimiento para habilitar dispositivos para regenerar de manera segura unos datos secretos fuertes de usuario a partir de datos secretos débiles para el usuario, comprendiendo el procedimiento:

la determinación (320) de los datos secretos débiles de usuario;

el cálculo (340) de los datos de petición del servidor para al menos un servidor que conserva secretos (130), en el que los datos de petición de servidor son una función de los datos secretos débiles y de un secreto de cliente efímero y los datos de petición de servidor no revelan información acerca de los datos secretos débiles sin el conocimiento del secreto de cliente efímero;

la recepción (340) de los datos de respuesta del servidor desde el servidor que conserva secretos, en el que los datos de respuesta de servidor son una función de los datos secretos de servidor para el servidor que conserva secretos y de los datos de petición de servidor, y los datos de respuesta del servidor no revelan información acerca de los datos secretos de servidor sin el conocimiento de los datos secretos débiles y del secreto de cliente efímero;

el cálculo (330) de un componente secreto para el servidor que conserva secretos como una función de los datos de respuesta de servidor recibidos desde el servidor que conserva secretos y del secreto de cliente efímero, en el que el componente secreto es una función de los datos secretos débiles y del secreto de servidor pero es independiente del secreto efímero;

el cálculo (330) de los datos secretos fuertes de usuario como una función del componente secreto; y

la determinación (350) de los datos de verificador para al menos un servidor de verificación (130J - 130N), en el que los datos de verificador habilitan al servidor de verificación para verificar que un dispositivo ha recuperado con éxito los datos secretos fuertes pero no es factible desde el punto de vista computacional para el servidor de verificación el determinar los datos secretos débiles solamente en base al acceso a sus datos de verificador.

2. El procedimiento de la reivindicación 1, en el que:

los datos secretos de servidor incluyen un entero aleatorio b;

el componente secreto es un valor $K = h(w^b)$, en el que h es una función;

w = f(datos secretos débiles), en el que f es una función que genera un elemento de un grupo finito G en el que la exponenciación es eficiente pero el problema del logaritmo discreto no es factible desde el punto de vista computacional; y

la exponenciación w^b se calcula en el grupo G;

el secreto de cliente efímero es un entero aleatorio a para el que existe un correspondiente entero a' de forma que $x^{aa'} = x$ para todo x en el grupo G;

los datos de petición de servidor se calculan como el valor $M = w^a$ en el que la exponenciación se calcula en el grupo G;

los datos de respuesta del servidor se calculan como el valor $c = M^b$, en los que la exponenciación se calcula en el grupo G; y

el componente secreto se calcula como el valor $K = h(c^{a'})$ en el que la exponenciación se calcula en el grupo G.

3. El procedimiento de la reivindicación 2 en el que el grupo G se selecciona de:

un grupo multiplicativo del conjunto de enteros módulo p, donde p es una prima grande adecuada como módulo Diffie-Hellman; y

un grupo de puntos sobre una curva elíptica sobre un fichero finito.

4. El procedimiento de la reivindicación 1 comprendiendo de manera adicional:

el encriptado de los datos privados para el usuario usando los datos secretos fuertes como una clave criptográfica en un criptosistema simétrico; en el que

el paso de determinar los datos de verificador incluye la determinación de datos públicos que corresponden a los datos privados de usuario, en el que una primera entidad con acceso a los datos privados puede probar el mencionado acceso a una segunda entidad con acceso a los datos públicos sin revelar los datos privados a la segunda entidad.

5. El procedimiento de la reivindicación 1 en el que el paso de determinar los datos de verificador comprende el cómputo de los datos de verificador como una función unidireccional de los datos secretos fuertes.

6. Un sistema (100) para habilitar dispositivos para regenerar de manera segura datos secretos fuertes de un usuario a partir de datos secretos débiles para el usuario, comprendiendo el sistema:

un cliente generador (120) y al menos un servidor que conserva secretos (130) acoplado al cliente generador, dicho cliente y dicho servidor comprendiendo un medio para ejecutar los siguientes pasos:

el cliente generador determinando los datos secretos débiles de usuario;

el cliente generador calculando y transmitiendo datos de petición de servidor a un servidor que conserva secretos, en el que los datos de petición del servidor son una función de los datos secretos débiles y de un secreto de cliente efímero, y los datos de petición de servidor no revelan información acerca de los datos secretos débiles sin conocimiento del secreto de cliente efímero;

el servidor que conserva secretos calculando y transmitiendo datos de respuesta de servidor al cliente generador, en el que los datos de respuesta del servidor son una función de los datos secretos del servidor para el servidor que conserva secretos y de los datos de petición del servidor, y los datos de respuesta del servidor no revelan información acerca de los datos secretos de servidor sin el conocimiento de los datos secretos débiles y del secreto de cliente efímero;

el cliente generador calculando un componente secreto para el servidor que conserva secretos como una función de los datos de respuesta del servidor recibidos desde el servidor que conserva secretos y del secreto de cliente efímero, en el que el componente secreto es una función de los datos secretos débiles y de los datos secretos de servidor pero es independiente del secreto de cliente efímero;

el cliente generador calculando los datos secretos fuertes de usuario como una función del componente secreto; y

el cliente generador determinando los datos de verificador para al menos un servidor de verificación (130J - 130N),

en el que los datos de verificador habilitan al servidor de verificación para verificar que un dispositivo ha recuperado con éxito los datos secretos fuertes pero no es factible desde el punto de vista computacional para el servidor de verificación el determinar los datos secretos débiles en base solamente al acceso a sus datos de verificador.

7. El sistema de la reivindicación 6 en el que:

los datos secretos de servidor incluyen un entero aleatorio b;

el componente secreto es un valor $K = h(w^b)$, en el que h es una función;

w = f(datos secretos débiles), en la que f es una función que genera un elemento de un grupo finito G en el que la exponenciación es eficiente pero el problema del logaritmo discreto no es factible desde el punto de vista computacional; y

la exponenciación w^b se calcula en el grupo G;

el secreto de cliente efímero es un entero aleatorio a para el que existe un correspondiente entero a' tal que $x^{aa'} = x$ para todos los x del grupo G;

los datos de petición del servidor se calculan como el valor $M = w^a$ en el que la exponenciación se calcula en el grupo G;

los datos de respuesta del servidor se calculan como el valor $c = M^b$ en el que la exponenciación se calcula en el grupo G; y

el componente secreto se calcula como el valor $K = h(c^{a'})$ en el que la exponenciación se calcula en el grupo G.

8. El sistema de la reivindicación 7 en el que el grupo G se selecciona de:

un grupo multiplicativo del conjunto de entero módulo p, donde p es una prima larga adecuada como un módulo Diffie-Hellman; y

un grupo de puntos sobre una curva elíptica sobre un campo finito.

9. El sistema de la reivindicación 6 en el que los pasos del procedimiento comprenden de manera adicional:

el cliente generador encriptando los datos privados para el usuario que usa los datos secretos fuertes como una clave criptográfica en un criptosístema simétrico; en el que

el paso de determinar los datos de verificador incluye la determinación de datos públicos que corresponden a los datos privados de usuario, en el que una primera entidad con acceso a los datos privados puede confirmar el mencionado acceso a una segunda entidad con acceso a los datos públicos sin revelar los datos privados a la segunda entidad.

10. El sistema de la reivindicación 6 en el que el paso de determinar los datos de verificador comprende el cálculo de los datos de verificador como una función unidireccional de los datos secretos fuertes.

11. Un producto de programa de ordenador que tiene instrucciones ejecutables por un ordenador de cliente generador para dar instrucciones al ordenador de cliente generador para que lleve a cabo todos los pasos de un procedimiento como se reivindica en cualquiera de las reivindicaciones 1 a la 5.

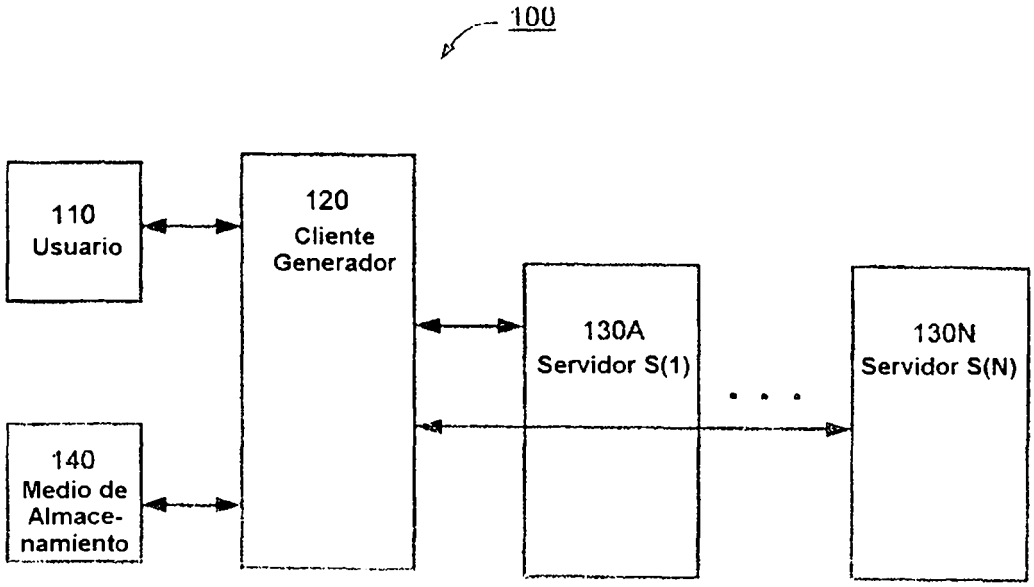
100
110
Usuario
120
Cliente
Generador
130A
Servidor S(1)
130N
Servidor S(N)
140
Medio de
Almace-
namiento

Figura 1

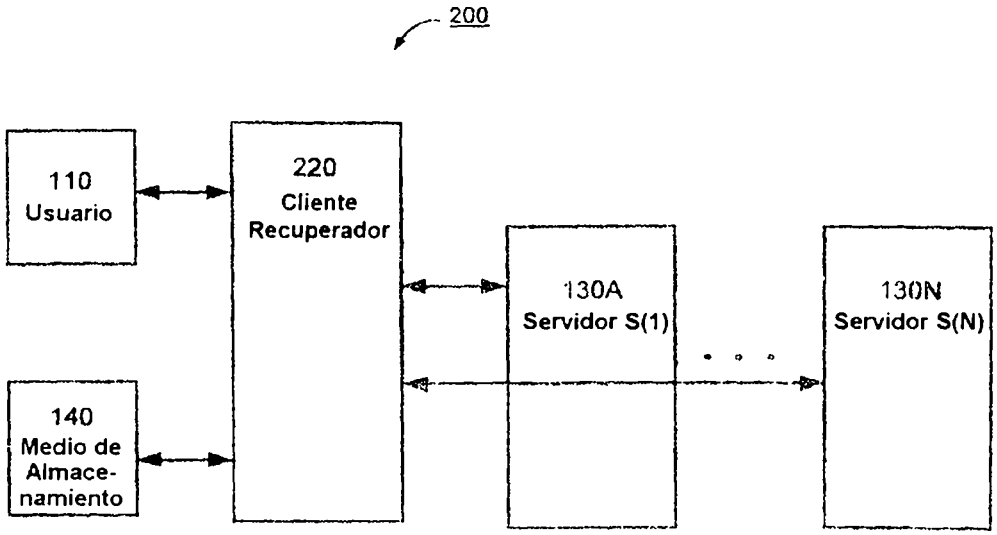
200
110
Usuario
220
Cliente
Recuperador
130A
Servidor S(1)
130N
Servidor S(N)
140
Medio de
Almace-
namiento

Figura 2

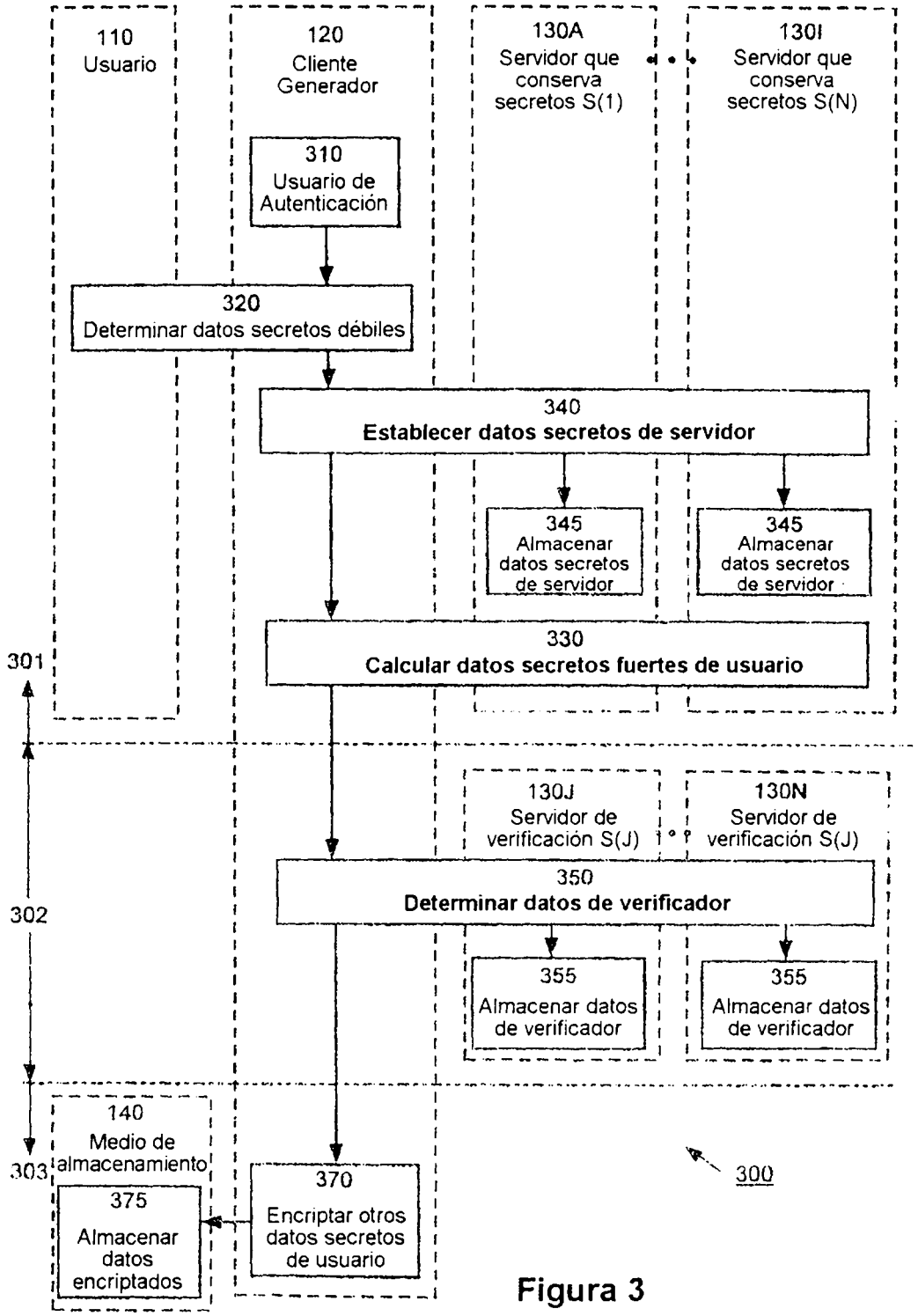
110
Usuario
120
Cliente
Generador
130A
Servidor que
conserva
secretos S(1)
130I
Servidor que
conserva
secretos S(N)
310
Usuario de
Autenticación
320
Determinar datos secretos débiles
340
Establecer datos secretos de servidor
345
Almacenar
datos secretos
de servidor
345
Almacenar
datos secretos
de servidor
330
Calcular datos secretos fuertes de usuario
301
130J
Servidor de
verificación S(J)
130N
Servidor de
verificación S(J)
350
Determinar datos de verificador
302
355
Almacenar datos
de verificador
355
Almacenar datos
de verificador
140
Medio de
almacenamiento
303
370
Encriptar otros
datos secretos
de usuario
375
Almacenar
datos
encriptados
300

Figura 3

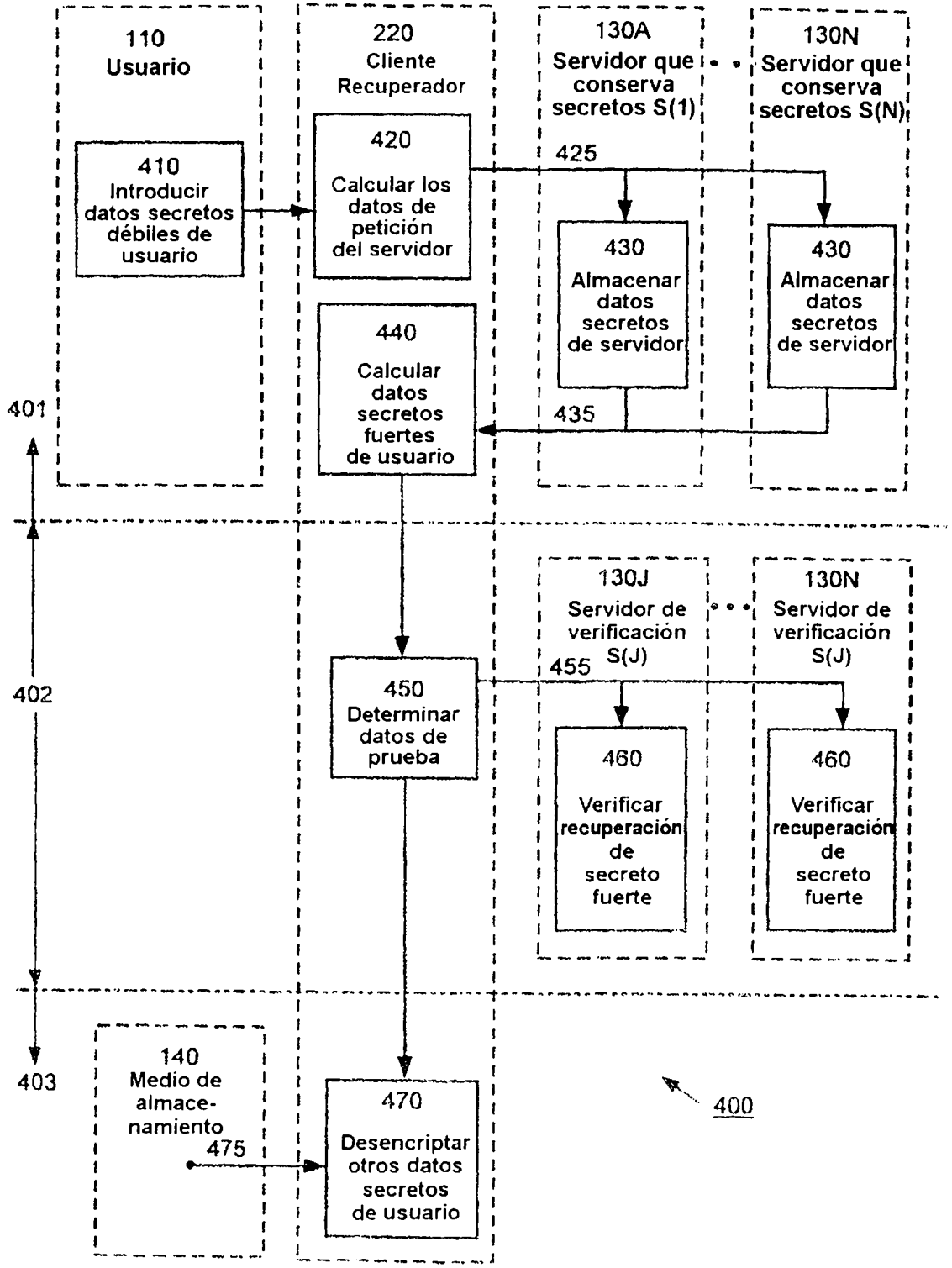
110
Usuario
410
Introducir datos secretos débiles de usuario
220
Cliente Recuperador
420
Calcular los datos de petición del servidor
440
Calcular datos secretos fuertes de usuario
130A
Servidor que conserva secretos S(1)
130N
Servidor que conserva secretos S(N)
425
430
Almacenar datos secretos de servidor
430
Almacenar datos secretos de servidor
435
401
402
403
130J
Servidor de verificación S(J)
130N
Servidor de verificación S(J)
455
450
Determinar datos de prueba
460
Verificar recuperación de secreto fuerte
460
Verificar recuperación de secreto fuerte
140
Medio de almacenamiento
475
470
Desencriptar otros datos secretos de usuario
400

Figura 4

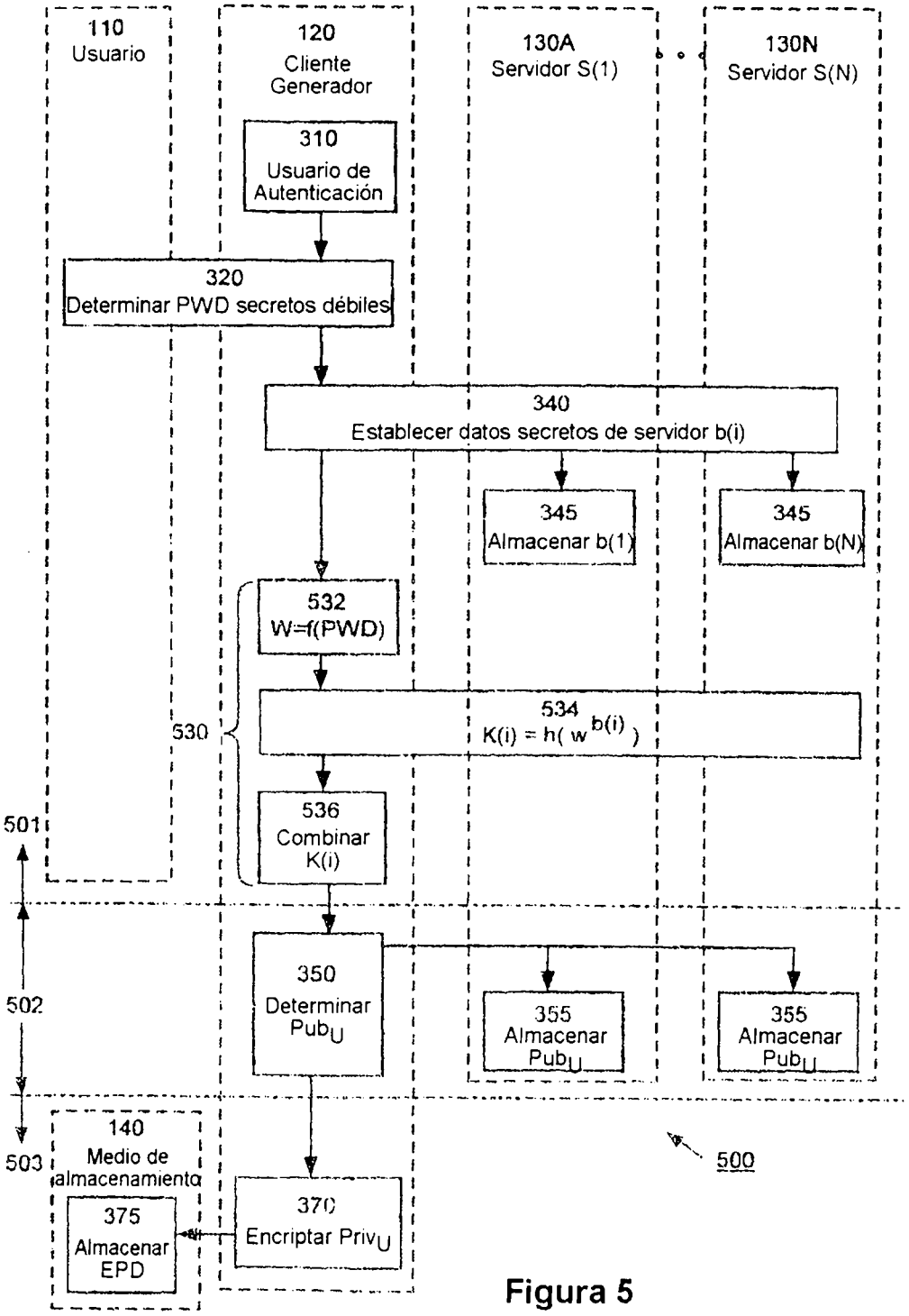
110
Usuario
120
Cliente
Generador
130A
Servidor S(1)
130N
Servidor S(N)
310
Usuario de
Autenticación
320
Determinar PWD secretos débiles
340
Establecer datos secretos de servidor b(i)
345
Almacenar b(1)
345
Almacenar b(N)
532
W=f(PWD)
534
K(i) = h(w b(i))
530
536
Combinar
K(i)
501
502
503
350
Determinar
Pub U
355
Almacenar
Pub U
355
Almacenar
Pub U
140
Medio de
almacenamiento
375
Almacenar
EPD
370
Encriptar Priv U
500

Figura 5

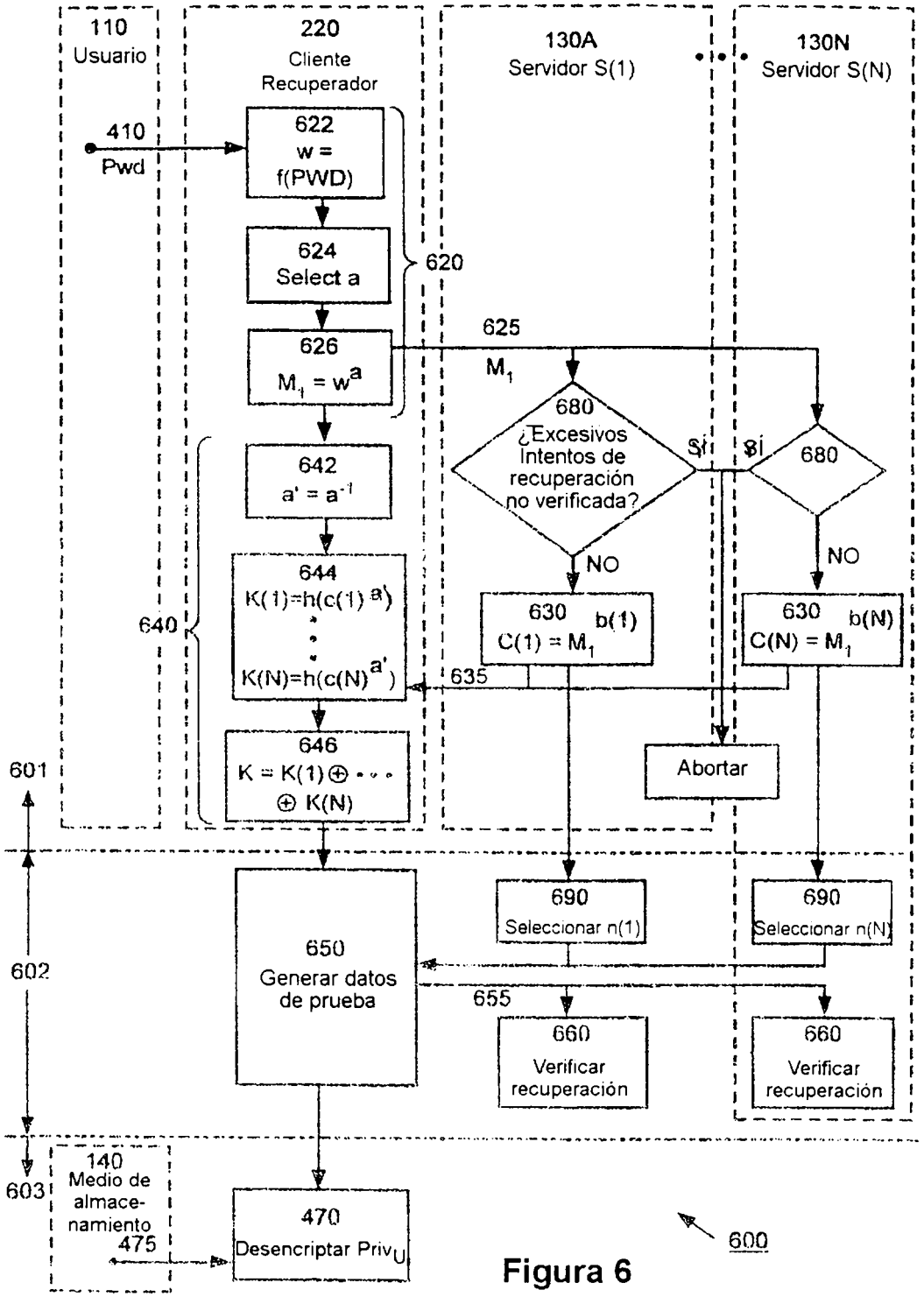
110
Usuario
220
Cliente
Recuperador
130A
Servidor S(1)
130N
Servidor S(N)
410
Pwd
622
w =
f(PWD)
624
Select a
620
626
$M_1 = w^a$
625
M_1
680
¿Excesivos
Intentos de
recuperación
no verificada?
Si
Si
680
NO
NO
642
$a' = a^{-1}$
644
$K(1)=h(c(1)^{a'})$
$K(N)=h(c(N)^{a'})$
640
630 b(1)
$C(1) = M_1$
630 b(N)
$C(N) = M_1$
635
646
$K = K(1) \oplus \cdots \oplus K(N)$
Abortar
601
602
650
Generar datos
de prueba
690
Seleccionar n(1)
690
Seleccionar n(N)
655
660
Verificar
recuperación
660
Verificar
recuperación
603
140
Medio de
almace-
namiento
475
470
Desencriptar $Priv_U$
600

Figura 6