Seguridad y Protección de Sistemas Informáticos

Fco. Javier Lobillo Borrero

Departamento de Álgebra, Universidad de Granada

Curso 2017/2018

Índice

Índice

Introducción

- El comercio en Internet ha llegado a depender casi exclusivamente de las instituciones financieras como terceros de confianza en el proceso de los pagos electrónicos.
- Las instituciones financieras no pueden evitar mediar en las disputas.
- El coste de esta mediación incrementa los costes de transacción, limitando su tamaño mínimo útil y eliminando la posibilidad de realizar pequeñas transacciones ocasionales, y hay un coste mayor al perderse la posibilidad de hacer transacciones irreversibles para servicios irreversibles.
- Se acepta como inevitable un cierto porcentaje de fraude.
- Esos costes y la incertidumbre en los pagos se pueden evitar cuando se usa dinero físico en persona, pero no existe mecanismo que permita realizar pagos a través de un canal de comunicación sin la participación de un tercero de confianza.
- Es necesario, por tanto, un sistema de pago electrónico basado en prueba criptográfica en lugar de confianza.
- Si las transacciones son computacionalmente imposibles de revertir, protegerán a los vendedores del fraude, y cualquier mecanismo de depósito de garantía se puede implementar fácilmente para proteger al comprador.

Herramientas: Funciones hash I

Necesitamos una función Hash H de n bits que tenga las propiedades:

- Dado $H(\texttt{nonce}||\texttt{msg})$ es computacionalmente imposible encontrar $\texttt{msg}$.
- Es computacionalmente imposible encontrar dos parejas $(\texttt{nonce}||\texttt{msg})$ y $(\texttt{nonce}'||\texttt{msg}')$ tales que $\texttt{msg} \neq \texttt{msg}'$ y $H(\texttt{nonce}||\texttt{msg}) = H(\texttt{nonce}'||\texttt{msg}')$.
- Para cualquier posible salida $y$de n bits, si elegimos k conteniendo una cadena dentro de una distribución uniforme de n bits, es computacionalmente imposible encontrar x tal que $H(k||x) = y$ en tiempo significativamente menor que 2^n.

Herramientas: Funciones hash II

Un puzle de búsqueda consiste en

- una función Hash H de n bits,
- un valor $\mathtt{id}$ conteniendo una cadena dentro de una distribución uniforme de n bits,
- un conjunto objetivo $Y \subseteq \mathbb{B}^n$.

Una solución del puzle es un valor x tal que

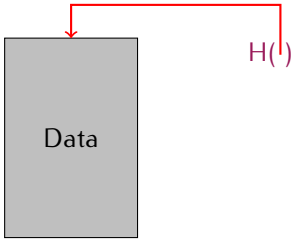
$$\mathrm{H}(\mathtt{id}||x) \in Y.$$

Bitcoin utiliza como función Hash `SHA256` basada en la construcción de Merkle-Damgard.

Herramientas: Blockchain I

Un puntero Hash

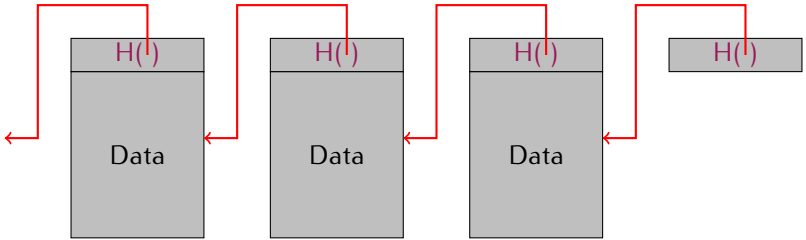
Es un puntero a unos datos junto con el Hash de esos datos.



Herramientas: Blockchain II

Blockchain

Es una lista enlazada mediante punteros Hash.

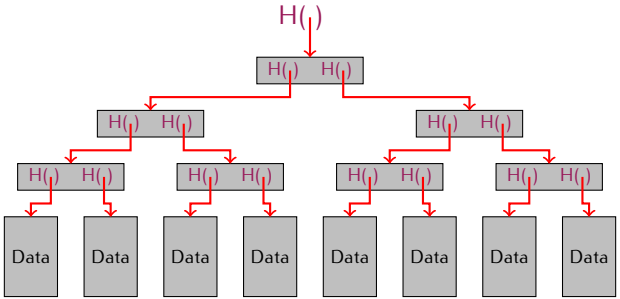


Utilidades

Registro inviolable: Alterar los datos en un bloque requiere recalcular los valores Hash de toda la cadena a partir de ese bloque, incluyendo el último puntero Hash que debe mantenerse guardado.

Herramientas: Árbol de Merkle I

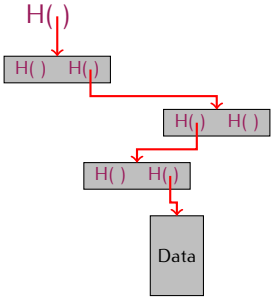
Supongamos que disponemos de un cierto número de bloques que contienen datos. Estos bloques serán las hojas de nuestro árbol.
Agrupamos las hojas en parejas de dos, y para cada pareja construimos una estructura de datos consistente en dos punteros hash, uno a cada uno de los bloques. Repetimos la operación hasta alcanzar un único puntero Hash, la raíz del árbol.



Herramientas: Árbol de Merkle II

Prueba de pertenencia

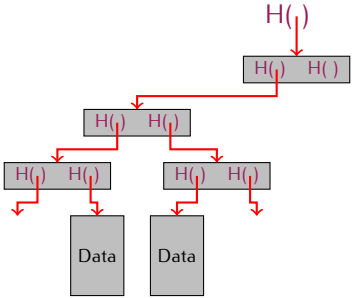
Para certificar que un bloque está en un árbol de Merkle, es suficiente con proporcionar la cadena de punteros Hash que conduce desde la raíz hasta el bloque en cuestión.



Herramientas: Árbol de Merkle III

Prueba de no pertenencia

Si los bloques de datos (las hojas) están ordenados, podemos probar que un bloque no está en el listado dando los caminos al anterior y al posterior.



Herramientas: ECDSA

secp256k1

```
Field Type: prime-field
Prime:
  00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
  ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:
  ff:fc:2f
A: 0
B: 7 (0x7)
Generator (uncompressed):
  04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:
  0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:
  f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:
  0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:
  8f:fb:10:d4:b8
Order:
  00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
  ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
  36:41:41
Cofactor: 1 (0x1)
```

Consenso distribuido

Protocolo

Partimos de N nodos, cada uno con un valor de entrada. Algunos de esos nodos son maliciosos o descuidados. Un protocolo de consenso distribuido tiene las siguientes propiedades:

- debe finalizar con todos los nodos honestos de acuerdo en un valor común,
- el valor debe haber sido generado por un nodo honesto.

Algoritmo de consenso de Bitcoin (simplificado)

La simplificación se encuentra en que se asume la posibilidad de seleccionar un nodo arbitrario de forma no vulnerable al ataque de Sybil.

1. Las nuevas transacciones se envían a todos los nodos.
2. Cada nodo recolecta las nuevas transacciones en un nodo.
3. En cada ronda un nodo consigue transmitir a los demás su bloque.
4. Los demás nodos aceptan el bloque solo si todas las transacciones son válidas.
5. Los nodos aceptan el bloque incluyendo su hash en el siguiente bloque que generan

Transacciones I

- Una moneda electrónica es una cadena de firmas digitales.
- Cada propietario transfiere la moneda al siguiente propietario firmando digitalmente un hash de la transacción previa y la clave pública del siguiente propietario, y añadiendo ambos al final de la moneda.
- El beneficiario puede verificar las firmas para verificar la cadena de propiedad.
- El beneficiario no puede verificar que uno de los propietarios no haya gastado dos veces la misma moneda.
- La solución habitual es introducir una autoridad central de confianza, o casa de la moneda, que comprueba cada transacción para que eso no se produzca.
- Tras cada transacción, la moneda debe regresar a la casa de la moneda para distribuir una nueva moneda, y solo las monedas emitidas directamente desde ella están libres de la sospecha de doble gasto.
- El destino de todo el sistema de dinero depende de la compañía que gestiona la casa de la moneda, por la cual pasa cada transacción.

Transacciones II

- Necesitamos una forma de que el beneficiario sepa que los propietarios previos no han firmado transacciones anteriores. La única manera de confirmar la ausencia de una transacción es tener conocimiento de todas las transacciones. En el modelo de la casa de la moneda, ésta tiene conocimiento de todas las transacciones y decide cuáles llegaron primero.
- Para lograr esto sin la participación de una parte de confianza, las transacciones han de ser anunciadas públicamente, y necesitamos un sistema para que los participantes estén de acuerdo en un único historial del orden en que fueron recibidas.
- El beneficiario necesita prueba de que en el momento de la transacción la mayor parte de los nodos estaban de acuerdo en que esa fue la primera que se recibió.

Sellado de tiempo

- Un servidor de sellado de tiempo trabaja tomando el hash de un bloque de ítems para sellarlos en el tiempo y notificar públicamente su *hash*.
- El sellado de tiempo prueba que los datos han existido en el tiempo, obviamente, para poder entrar en el hash.
- Cada sellado de tiempo incluye el sellado de tiempo previo en su hash, formando una cadena, con cada sellado de tiempo adicional reforzando al que estaba antes.

Proof-of-work I

- La proof-of-work consiste en buscar un valor cuyo hash comience con un número de bits cero.
- El trabajo medio que hace falta es exponencial en el número de cero bits requeridos y puede verificarse ejecutando un único hash.
- Para nuestra red de sellado de tiempo, implementamos la proof-of-work incrementando un nonce en el bloque hasta que se encuentre un valor que otorgue al hash del bloque los cero bits requeridos.
- Una vez que se ha agotado el esfuerzo de CPU para satisfacer la proof-of-work, el bloque no se puede cambiar sin rehacer el trabajo.
- A medida que bloques posteriores se encadenen tras él, el trabajo para cambiar un bloque incluiría rehacer todos los bloques anteriores.
- La proof-of-work también resuelve el problema de determinar la representación en la toma de decisiones mayoritarias.
- La proof-of-work es esencialmente un voto por CPU.
- La decisión de la mayoría está representada por la cadena más larga, en la cual se ha invertido el mayor esfuerzo de proof-of-work.

Proof-of-work II

- Si la mayoría de la potencia CPU está controlada por nodos honestos, la cadena honesta crecerá más rápido y dejará atrás cualquier cadena que compita.
- Para modificar un bloque pasado, un atacante tendría que rehacer la proof-of-work del bloque y de todos los bloques posteriores, y entonces alcanzar el trabajo de los nodos honestos.
- Para compensar el aumento en la velocidad del hardware y el interés variable de los nodos activos a lo largo del tiempo, la dificultad de la proof-of-work está determinada por una media móvil que apunta a un número medio de bloques por hora. Si se generan muy deprisa, la dificultad aumenta.

Red I

- Los pasos para ejecutar la red son:
  1. Las transacciones nuevas se transmiten a todos los nodos.
  2. Cada nodo recoge todas las transacciones en un bloque.
  3. Cada nodo trabaja en resolver una proof-of-work compleja para su bloque.
  4. Cuando un nodo resuelve una proof-of-work, transmite el bloque a todos los nodos.
  5. Los nodos aceptan el bloque si todas las transacciones en él son válidas y no se han gastado con anterioridad.
  6. Los nodos expresan su aceptación del bloque al trabajar en crear el siguiente bloque en la cadena, usando el hash del bloque aceptado como hash previo.
- Los nodos siempre consideran correcta a la cadena más larga y se mantendrán trabajando para extenderla. Si dos nodos transmiten simultáneamente diferentes versiones del siguiente bloque, algunos nodos recibirán una antes que la otra. En ese caso, trabajarán sobre la primera que hayan recibido, pero guardarán la otra ramificación por si acaso se convierte en la más larga.
- El empate se romperá cuando se encuentre la siguiente proof-of-work y una ramificación se convierta en la más larga; los nodos que trabajaban en la otra ramificación cambiarán automáticamente a la más larga.

Red II

- La transmisión de nuevas transacciones no precisa alcanzar todos los nodos, con alcanzar a la mayoría de los nodos, entrarán en un bloque en poco tiempo.
- Las transmisiones de nodos también toleran mensajes perdidos. Si un nodo no recibe un bloque, lo reclamará cuando reciba el siguiente bloque y se dé cuenta de que falta uno.

Incentivo I

- Por convenio, la primera transacción en un bloque es una transacción especial con la que comienza una moneda nueva, propiedad del creador del bloque.
- Esto añade un incentivo a los nodos para soportar la red, y proporciona una forma de poner las monedas en circulación, dado que no hay autoridad central que las distribuya.
- La inserción estable de una cantidad constante de monedas nuevas es análoga at trabajo de mineros de oro, que consumen recursos para añadir oro a la circulación. En nuestro caso, es tiempo de CPU y electricidad lo que se gasta.
- El incentivo también se basa en las comisiones por transacción. Si el valor de salida de una transacción es menor que el valor de entrada, la diferencia es una comisión por transacción que se añade al valor de incentivo del bloque que contiene la transacción. Una vez que un número predeterminado de monedas ha entrado en circulación, el incentivo puede evolucionar hacia comisiones de transacción y estar completamente libre de inflación.

Incentivo II

- El incentivo puede ayudar a que los nodos permanezcan honestos. Si un atacante codicioso fuera capaz de reunir más potencia CPU que la de todos los nodos honestos, tendría que escoger entre usarla para defraudar a la gente robándoles los pagos recibidos, o usarla para generar nuevas monedas. Debe encontrar más rentable respetar las reglas, esas reglas que le favorecen entregándole más monedas nuevas que a todos los demás en conjunto, que socavar el sistema y la validez de su propia riqueza.

Espacio de disco

- Cuando la última transacción de una moneda está enterrada bajo suficientes bloques, las transacciones que se han gastado antes que ella se pueden descartar para ahorrar espacio de disco.
- Para facilitar esto sin romper el hash del bloque, los valores hash de las transacciones almacenados en un Árbol de Merkle, incluyendo solo la raíz en el hash del bloque. Los bloques viejos pueden compactarse podando ramas del árbol. Los hashes interiores no necesitan ser guardados.
- Una cabecera de bloque sin transacciones pesaría unos $80B$. Si suponemos que los bloques se generan cada 10 minutos, $80B \times 6 \times 24 \times 365 = 4{,}2MB$ por año. Siendo habitual la venta de ordenadores con 2GB de RAM en 2008, y con la Ley de Moore prediciendo un crecimiento de $1{,}2GB$ anual, el almacenamiento no debería suponer un problema incluso si hubiera que conservar en la memoria las cabeceras de bloque.

Verificación de pagos simplificada I

- Es posible verificar pagos sin ejecutar un nodo plenamente en red. El usuario solo necesita tener una copia de las cabeceras de bloque de la cadena más larga de proof-of-work, que puede conseguir solicitándola a los nodos de red hasta estar convencido de que tiene la cadena más larga, y obtener la rama Merkle que enlaza la transacción con el bloque en que está sellado en el tiempo.
- El usuario no puede comprobar la transacción por sí mismo pero, al enlazarla a un lugar en la cadena, puede ver que un nodo de la red la ha aceptado, y los bloques añadidos posteriormente confirman además que la red la ha aceptado.
- la verificación es fiable en tanto que los nodos honestos controlen la red, pero es más vulnerable si un atacante domina la red.
- Mientras que los nodos de red pueden verificar las transacciones por sí mismos, el método simplificado puede ser engañado por transacciones fabricadas por un atacante, en tanto el atacante pueda continuar dominando la red.

Verificación de pagos simplificada II

- Una estrategia para protegerse contra esto podría ser aceptar alertas de los nodos de red cuando detecten un bloque no válido, sugiriendo al software del usuario que descargue el bloque entero y las transacciones con aviso para confirmar la inconsistencia.
- Los negocios que reciban pagos con frecuencia seguramente preferirán tener sus propios nodos ejecutándose para tener más seguridad independiente y verificación más rápida.

Combinando y dividiendo valor

- Aunque sería posible manipular monedas individualmente, no sería manejable hacer una transacción para cada céntimo en una transferencia.
- Para permitir que el valor se divida y combine, las transacciones contienen múltiples entradas y salidas.
- Normalmente habrá, o bien una entrada simple de una transacción anterior mayor, o bien múltiples entradas combinando pequeñas cantidades, y como máximo dos salidas: una para el pago y otra devolviendo el cambio, si lo hubiera, al emisor.
- Cabe señalar que la diseminación de control, donde una transacción depende de varias transacciones, y esas transacciones dependen de muchas más, no supone aquí un problema. No existe la necesidad de extraer una copia completa e independiente del historial de una transacción.

Privacidad

- El modelo tradicional de banca consigue un nivel de privacidad limitando el acceso a la información a las partes implicadas y el tercero de confianza.
- La necesidad de anunciar públicamente todas las transacciones excluye este método, pero aún así se puede mantener la privacidad rompiendo el flujo de información en otro punto: manteniendo las claves públicas anónimas.
- El público puede ver que alguien está enviando una cantidad a otro alguien, pero sin que haya información vinculando la transacción con nadie. Esto es similar al nivel de información que comunican las bolsas de valores, donde el tiempo y tamaño de las operaciones individuales, la "cinta", son hechos públicos, pero sin decir quiénes fueron las partes.
- Como cortafuegos adicional, debería usarse un nuevo par de claves en cada transacción para evitar que se relacionen con un propietario común.
- Con las transacciones multientrada será inevitable algún tipo de vinculación, pues revelan necesariamente que sus entradas pertenecieron al mismo propietario.
- El riesgo es que si se revela la identidad del propietario de una clave, la vinculación podría revelar otras transacciones que pertenecieron al mismo propietario.

Por qué funciona. I

- Consideramos el escenario de un atacante intentando generar una cadena alternativa más rápido que la cadena honesta.
- Incluso si se consigue, el sistema no queda abierto a cambios arbitrarios como crear valor de la nada o tomar dinero que nunca perteneció al atacante. Los nodos no van a aceptar una transacción inválida como pago y los nodos honestos nunca aceptarán un bloque que las contenga.
- Un atacante solo puede tratar de cambiar una de sus propias transacciones para recuperar dinero que ha gastado recientemente.
- La carrera entre la cadena honesta y la cadena de un atacante puede verse como un camino aleatorio binomial. El suceso que prospera es la cadena honesta añadiendo un bloque, aumentando su liderato por $+1$, y el suceso que fracasa es la cadena del atacante añadiendo un bloque, reduciendo la brecha en -1.
- La probabilidad de que un ataque alcance la cadena honesta desde un déficit dado es análoga al problema de la ruina del jugador.

Por qué funciona. II

- Supongamos que un jugador con crédito ilimitado comienza con un déficit y juega en potencia un número infinito de intentos para alcanzar un punto de equilibrio. Podemos calcular la probabilidad de que alcance ese punto, o de que un ataque alcance alguna vez a la cadena honesta

$$
\begin{aligned}
p &= \text{probabilidad de que un nodo honesto encuentre el siguiente bloque} \\
q &= \text{probabilidad de que el atacante encuentre el siguiente bloque} \\
q_z &= \text{probabilidad de que el atacante alcance la cadena honesta desde } z \text{ bloques atrás}
\end{aligned}
$$

$$
q_z = \begin{cases} 1 & \text{si } p \leq q \\ (q/p)^z & \text{si } p > q \end{cases}
$$

- Asumiendo que $p > q$, la probabilidad cae de forma exponencial a medida que aumenta el número de bloques que el atacante tiene que alcanzar. Con las probabilidades en su contra, si no tiene un golpe de suerte que lo haga avanzar desde el principio, sus oportunidades se irán desvaneciendo a medida que se va quedando atrás.

Por qué funciona. III

- Cuánto tiempo necesita esperar el receptor de una transacción para tener la suficiente seguridad de que el emisor no puede cambiarla. Asumimos que el emisor es un atacante que quiere que el receptor crea durante un tiempo que le ha pagado; entonces cambiará el pago para devolvérselo a sí mismo un tiempo después. El receptor recibirá un aviso cuando esto suceda, pero el emisor espera que ya sea demasiado tarde.
- El receptor genera un nuevo par de claves y da la clave pública al emisor poco antes de firmar. Esto impide que el emisor pueda preparar una cadena de bloques previa trabajando de continuo en ella hasta tener la suerte suficiente como para ponerse a la cabeza y, entonces, ejecutar la transacción. Una vez que la transacción se ha emitido, el emisor deshonesto comienza a trabajar en secreto en una cadena paralela que contiene una versión alternativa de su transacción.
- El receptor espera hasta que la transacción se ha añadido al bloque y z bloques se han enlazado tras él. No sabe la cantidad de progreso que ha realizado el atacante, pero asumiendo que los bloques honestos han tomado la media de tiempo esperada por bloque, el potencial de progreso del atacante será una distribución de Poisson9 con un valor esperado $\lambda = z\frac{q}{p}$.

Por qué funciona. IV

- Para obtener la probabilidad de que el atacante pueda ponerse al día incluso ahora, multiplicamos la densidad de Poisson para cada aumento en el progreso que podría haber realizado, por la probabilidad que podría haber alcanzado desde ese punto:

$$\sum_{k\geq 0} \frac{\lambda^k e^{-k}}{k!} \cdot \begin{cases} (q/p)^{z-k} & \text{si } k \leq z \\ 1 & \text{si } k > z \end{cases} = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-k}}{k!} \left(1 - (q/p)^{z-k}\right)$$

Bibliografía I


C. Adams, P. Cain, D. Pinkas, and R. Zuccherato.
Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).
IETF, August 2001.


Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker.
Digital Watermarking and Steganography.
The Morgan Kaufmann Series in Multimedia Information and Systems. Elsevier, second edition edition, 2008.


D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk.
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
IETF, May 2008.


Quynh H. Dang.
The Keyed-Hash Message Authentication Code (HMAC).
National Institute of Standards and Technology (NIST), July 2008.

Bibliografía II


Kresimir Delac and Mislav Grgic.
A survey of biometric recognition methods.
In *46th International Symposium Electronics in Marine, ELMAR-2004*, pages 184–193, 2004.


Hans Delfs and Helmut Knebl.
Introduction to Cryptography. Principles and Applications.
Information Security and Cryptography. Springer, third edition, 2015.


T. Dierks and E. Rescorla.
The Transport Layer Security (TLS) Protocol Version 1.2.
IETF, August 2008.


Educause.
7 things you should know about Federared Identity Management, September 2009.

Peter Gutmann.
Everything you Never Wanted to Know about PKI but were Forced to Find Out.
Technical report, University of Auckland.

Bibliografía III


ISO/IEC.
Information technology — Security techniques — A framework for identity management —, December 2011.


Joseph Migga Kizza.
Guide to Computer Network Security.
Computer Communications and Networks. Springer, 3rd. edition, 2015.


S. Kent and K. Seo.
Security Architecture for the Internet Protocol.
IETF, December 2005.


Andre Karamanian, Srinivas Tenneti, and Francois Dessart.
PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks.
Cisco Press, 2011.

Bibliografía IV


Carlos Munuera.
Steganography from a Coding Theory Point of View, pages 83–128.
WORLD SCIENTIFIC, 2013.


Satoshi Nakamoto.
Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer.
Technical report, bitcoin.org.
Traducido por @breathingdog.


National Institute of Standards and Technology (NIST).
DATA ENCRYPTION STANDARD (DES), October 1999.


National Institute of Standards and Technology (NIST).
ADVANCED ENCRYPTION STANDARD (AES), November 2001.


National Institute of Standards and Technology (NIST).
SECURE HASH STANDARD, August 2002.

Bibliografía V


National Institute of Standards and Technology (NIST).
Digital Signature Standard (DSS), July 2013.


Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry.
Fundamentals of Computer Security.
Springer, 2003.