

De acuerdo a David Hernández de ComputerHoy, la publicación referida en el apunte, las tendencias de lenguajes de programación que se espera para el 2020 son:

2020	
Lenguaje	Principales características
Swift	Muestra gran estabilidad y madurez, lo que hace que quizás sea el momento de aprenderlo. Desarrollado por Apple para programar en IOS. Posee muchas herramientas y es de código abierto
Kotlin	Es "el" lenguaje de programación para Android según Google, totalmente operable con Java y EXTREMADAMENTE demandado en estos últimos meses
Python	Es un lenguaje de programación general, uno de los de más rápido crecimiento. Un todo-terreno y sencillo de aprender, muy popular y favorito de los científicos de datos y analistas académicos. Permite agregar comentarios que el intérprete ignora.
Dart	Desarrollado por Google, un lenguaje de código abierto pensado para trabajar desde el cliente, utilizado para la app Flutter (código multiplataforma para crear aplicaciones). Pretendía originalmente sustituir a JavaScript por ofrecer mejores resultados a algunos problemas que éste presenta.
TypeScript	Desarrollado por Microsoft es de programación libre y código abierto, derivado de JavaScript aunque está superándolo.
Rust	Si bien es desarrollado y patrocinado por Mozilla y Samsung, es un proyecto comunitario de creciente popularidad por ser totalmente abierto y amplia variedad de uso.
Java	Aún vigente, pero no lo recomiendan como primera opción. Orientado a objetos, de código abierto, robusto, independiente de la plataforma y seguro

En 2020 algunos lenguajes mantienen la tendencia que se vislumbraba en 2019 aunque algunos han bajado en la consideración de la publicación consultada (ComputerHoy)

2019	
Lenguaje	Características
Kotlin	Mencionado en segundo lugar en la lista 2020
Java	También en la lista 2020 pero con una consideración bastante menor.

Javascript	Un lenguaje muy popular, presente en la mayoría de las páginas web Front End y Back End. Fue reemplazado en la lista de 2020 por Typescript
C y C++	muy utilizados en videojuegos, eficiente y flexible. alto rendimiento y fiabilidad, pero no mencionado en la tendencia 2020
PHP	también muy frecuente en las páginas web. Sitios web dinámicos y estáticos. Tampoco aparecen como tendencia en 2020
Rust	sigue vigente en 2020
C# o C Sharp	Es de Microsoft y forma parte de la plataforma .net - Orientado a objetos y fácil de aprender pero descartado en la consideración de CH en 2020
Python	Mejor posicionado en la tendencia 2020
Go	Es un lenguaje de programación desarrollado por Google, multiplataforma, compilado, concurrente, imperativo, estructurado, orientado a objetos y con recolector de basura, de código abierto. No mencionado en la tendencia 2020
Swift	Es uno de los mejor considerados en la tendencia de 2020

Cuestionario TIC 1

1 ¿Qué son las TIC's según la Ley?

R *Según la Ley 27078, las Tecnologías de la información y las comunicaciones (TIC's) son el conjunto de recursos, herramientas, equipos, programas informáticos, aplicaciones, redes y medios que permitan la compilación, procesamiento, almacenamiento y transmisión de información, como por ejemplo, voz, texto, audios, imágenes y videos entre otros.*

2 ¿Qué es el SBT?

R *El servicio básico telefónico (SBT) es el servicio que provee telefonía nacional e internacional de voz a través de redes locales, independientemente de la tecnología utilizada para su transmisión, siempre que permita la comunicación de sus usuarios entre sí.*

3 ¿Qué es el Servicio de Telecomunicación?

R *Es el servicio de transmisión, emisión o recepción de escritos, signos, señales, imágenes, sonidos o información de cualquier naturaleza, por hilo, radioelectricidad, medios ópticos u otros sistemas electromagnéticos a través de redes de telecomunicaciones.*

4 ¿Qué es el Servicio de vídeo a pedido o a demanda?

R *El servicio de vídeo a pedido o a demanda, es el servicio ofrecido por un prestador de servicios de TIC para el acceso a programas en el momento elegido y a petición propia, sobre la base de un catálogo.*

Cuestionario TIC 2

Preguntas

1. ¿Qué son los servicios de TIC's?
 - Son los servicios que utilizan las redes de telecomunicaciones para cumplir su objetivo de transporte y distribución de señales o datos como voz, texto, video e imágenes desde y hacia terceros usuarios.
2. ¿Qué es la Telecomunicación?
 - Es toda transmisión, ya sea recepción o emisión de signos, señales, sonidos, escritos, imágenes o información de cualquier tipo a través de hilos, radioelectricidad, medios ópticos u otros medios electromagnéticos.
3. ¿Qué se requiere para la prestación de Servicios de TIC?
 - Para la prestación de servicios TIC se requerirá la previa obtención de la licencia habilitante y registrar cada servicio de acuerdo lo solicite la autoridad de aplicación.

Documento Ciberseguridad 1

¿Qué es la **ciberseguridad**?

Es la seguridad de la tecnología de la información, **engloba un gran número de técnicas y métodos para proteger nuestro sistema**, así como otros dispositivos o las redes.

¿Qué hace **un especialista** en ciberseguridad?

El especialista en ciberseguridad es el encargado de la privacidad y protección de datos de las empresas y las organizaciones para hacer frente a los ciberataques.

¿Qué hace un **Consultor** de Ciberseguridad?

Un **Consultor de Seguridad** es el encargado de asesorar y supervisar las medidas de seguridad necesarias para proteger de manera efectiva los bienes de una empresa o de un cliente, en tal sentido, hacen uso de su conocimiento y experticia para evaluar las potenciales amenazas de seguridad y violaciones para prevenirlas y desarrollar protocolos y planes de contingencia en caso de cualquier incidencia.

1 ¿Cuáles son las fases de la ciberseguridad?

Prevención: Es el primer paso que reducirá en gran medida el margen de riesgo. **Actuar de forma temprana e informarnos**, determinar las posibles amenazas y cuáles serán las medidas de prevención y reacción en caso de vernos afectados. **Los integrantes de la organización deben tener conocimientos básicos** sobre ciberseguridad.

Localización, en caso de haber sufrido algún tipo de problema, localizar dónde radica el problema. Para ello disponer de un antivirus potente. **Gestionar las vulnerabilidades de nuestro sistema y llevar a cabo una monitorización** de forma continua.

Reacción: Una vez localizada la amenaza, dar respuesta técnica sobre la misma **desconectando los equipos de la red**, instalar un antivirus o actualizar el que ya teníamos. Despues, **análisis sobre el sistema y cambios de todas las contraseñas**. **Para terminar, realizar una limpieza** a fondo del sistema. En el caso de que nos hayan **robado datos** o información confidencial, también deberemos proceder de la manera pertinente para **comunicarlo a los usuarios afectados y elevar lo ocurrido a una situación de delito informático**.

2

3

¿En qué época se empezó a hablar de la ciberseguridad?

En la década del 80 comenzó ha tratarse la ciberseguridad, coincidente con la aparición de los primeros malwares y la ingeniería social.

¿En qué se diferencian un hacker y un ciberdelincuente?

Un **hacker** es una persona que investiga los sistemas para detectar fallos y mejorarlos, por el contrario un **ciberdelincuente** (cracker) busca esos mismos fallos para explotarlos y obtener un beneficio.

¿Cuáles son las principales amenazas de ciberseguridad a las que debemos estar atentos?

Robo de datos de establecimientos minoristas

Seguridad móvil y amenazas que aprovechan las vulnerabilidades de los teléfonos

Ataques de phishing e ingeniería social

Robo de identidad

Pirateo de datos médicos

Depredadores sexuales que acosan a los niños

Ataques a bancos

Documento Ciberseguridad 2

1. ¿Qué es el ping de la red?

- Es un comando o herramienta diagnóstica que permite verificar el estado de una conexión de red o con un host local.

2. ¿Qué tipo de ataque es Ping de la Muerte?

- Un ping de la muerte es un tipo de ataque enviado a una computadora que consiste en mandar numerosos paquetes ICMP muy grandes (mayores a 65.535 bytes) con el fin de colapsar el sistema atacado.
- Es un tipo de ataque que fue corregido por los sistemas operativos posteriores a 1997.

3. ¿Qué es un análisis de vulnerabilidad y cuáles son los pasos para realizarlo?

- Es un proceso por el cual la organización determina el nivel de exposición y predisposición a perder elementos frente a una amenaza determinada intencional o no.
- Pasos para realizarlo
 - Identificación de los tipos (tecnológicas, naturales y sociales)
 - Calificación: Posible, probable e inminente.
 - Ámbito de análisis: personas, recursos, sistemas y procesos.
 - Interpretación del nivel de riesgo: diamante de riesgo.

4. ¿Cuál es la IP para hacer ping a tu propio servidor local?

- En el símbolo del sistema se escribe: ping 127.0.0.1 y se presiona Enter, se realiza un ping al propio servidor local,

5. ¿Cómo saber si el ping es bueno?

- Si el ping es menor de 60 milisegundos, se considera aceptable, aunque lo ideal sería menor a 20 milisegundos.

6. ¿Qué es un ciberataque y cuántos tipos hay?

- Un ciberataque es un conjunto de acciones ofensivas contra sistemas de información como bases de datos, redes computacionales, etc. hechas para dañar, alterar o destruir instituciones, personas o empresas.
- Existen cuatro grandes grupos:
 - Cibercrimen: robo de identidad, fraudes con fines económicos.
 - Hacktivismo: ataque a grandes empresas o gobiernos por motivos ideológicos.
 - Ciberespionaje: robo de información con motivos económicos.
 - Ciberterrorismo: dirigidos contra gobiernos, sistemas de defensa o estructuras de salud.

7. ¿Qué es el análisis de riesgos informáticos?

- El análisis de riesgos informáticos es la evaluación de los peligros que existen a nivel informático, los posibles eventos críticos, los daños que pueden producirse y las situaciones que amenazan tu entorno de negocio.

Máquina virtual y virtualización

1. ¿cuántos tipos de máquinas virtuales existen?
 - Existen dos tipos de máquinas virtuales:
 1. De sistema: emula una pc completa
 2. De procesos: ejecuta un solo proceso o aplicación dentro de su entorno de ejecución.
2. ¿Una máquina virtual puede acceder al resto de datos de la máquina anfitrión?
 - No puede acceder al resto de los datos de la maquina anfitrión, aunque algunas herramientas permiten copiar archivos de una máquina a otra.
3. ¿cómo suele llamarse a la PC que contiene a la máquina virtual?
 - Suele llamarse hipervisor, host o anfitrión.
4. ¿A qué se refiere cuando hablamos de guest?
 - A la maquina virtual con su sistema operativo “invitado”, se encuentra dentro de la maquina anfitrión.
5. ¿Qué son los contenedores y cómo operan?
 - Un contenedor es un tipo de virtualización que permite crear un entorno aislado para la ejecución una aplicación, compartiendo binarios y librerías con otros contenedores. Utilizan menos recursos que una máquina virtual. También se los conoce como “máquinas virtuales ligeras”
6. ¿cuál es la diferencia entre Cloud computing y virtualización?
 - Cloud computing: es el servicio que resulta de la manipulación del hardware por parte del software (virtualización), consiste en la entrega de recursos informáticos compartidos a través de software o datos y que son entregados como un servicio de demanda a través de Internet.
 - Virtualización: es la simulación que permite compartir recursos de hardware. Es la tecnología que permite la cloud computing. Las máquinas virtuales y los contenedores son tipos de virtualización.

Unidades de almacenamiento

1. ¿Cuál unidad ofrece el **menor costo** por gigabyte de almacenamiento?
 1. Las Unidades **HDD** son las mas económicas por GB de almacenamiento.
2. ¿Cuál unidad ofrece un **consumo de corriente más eficiente**?
 1. La unidad **SSD** son mas eficientes en el consumo de energía.
3. ¿Cuáles unidades ofrecen un **máximo rendimiento para el arranque y alto rendimiento de lectura/escritura**?
 1. Las unidades **SSD** son las de máximo rendimiento para el arranque y alto rendimiento de lectura / escritura.
4. ¿Cuáles ofrecen un **amplio rendimiento** para la mayoría de las plataformas de PC?
 1. Las unidades **HDD** generalmente ofrecen un amplio rendimiento para la mayoría de las plataformas de PC que se distribuyen hoy día.
5. ¿Cuáles unidades son **las más delgadas** de todas las opciones?
 1. Los productos **SSD** son los más delgados de todas las opciones de almacenamiento disponibles.
6. En principio ¿Cuáles unidades serían **más durables** debido a su diseño?
 1. En principio las unidades **SSD** serían las más duraderas debido a su diseño. Sin partes móviles, pueden soportar extremos mayores de impacto, caídas y temperaturas.
7. ¿Cuáles de todas las opciones pueden brindar **la mejor relación rendimiento/presupuesto**?
 1. Las opciones híbridas, **las SSHD**, serían quienes pueden brindar la mejor relación rendimiento/presupuesto.

Documento Ransomware

1) ¿De qué se trata en líneas generales el ransomware?

a) En líneas generales, un ransomware es un ataque en el cual el ciberdelincuente se apropia de los sistemas encriptando los archivos de la víctima a la cual le piden un rescate en bitcoins. La gran mayoría de las veces NO se devuelven los archivos luego del pago, por lo que NUNCA debe pagarse por un ransom.

2) ¿Cuáles son sus fases? Enumerarlas y describirlas en pocas líneas (con sus palabras)

a) Proceso inicial, en el cual el ciberdelincuente hace contacto con la víctima, generalmente a través de un correo electrónico de tipo phishing.

b) Proceso de ejecución y escalada de privilegios. El primer objetivo del atacante es conseguir permisos de administrador, luego de lo cual la víctima queda a merced del atacante.

c) Evasión de defensas y credenciales de acceso, una vez con permisos de administrador el delincuente modifica los ajustes de seguridad a su medida. Generalmente se utilizan herramientas muy difundidas.

d) Discovery (descubrimiento de la red), En esta fase se accede a la estructura del sistema, en su aspecto lógico, y se obtiene un "mapa" de todo el sistema, tipo de almacenamiento de datos, copias de respaldo, etc.

e) Movimiento lateral e impacto, se llega a esta fase a través de protocolos como RDP (de escritorio remoto). Finalmente ocurre la ejecución del ransomware, para lo cual, también se utilizan herramientas no muy difíciles de conseguir.

3) ¿Cuáles son los errores frecuentes que nos hacen víctimas del ransomware?

a) No prestar atención a la protección del acceso a nuestros sistemas, debemos cuidar de no dejar abiertos innecesariamente puertos de acceso.

b) Contraseñas débiles sin métodos de autenticación adicionales.

c) No prestar atención a los logs del sistema, estos ataques se toman su tiempo de preparación, lo cual nos permitiría detectar anomalías.

d) Ignorar alertas, alarmas que se disparan, etc. NO abrir correos electrónicos sospechosos, sobre todo si tienen archivos adjuntos.

e) No mantener actualizados los sistemas.



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

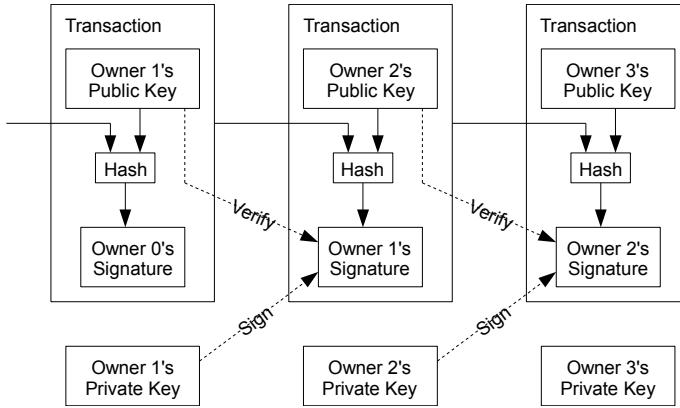
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

2. Transactions

We define **an electronic coin as a chain of digital signatures**. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. **A payee can verify the signatures to verify the chain of ownership.**

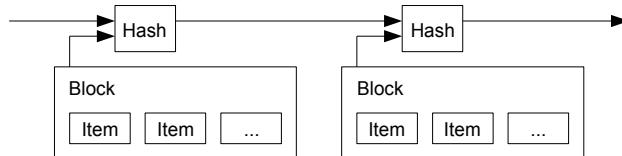


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, **the earliest transaction is the one that counts, so we don't care about later attempts to double-spend**. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

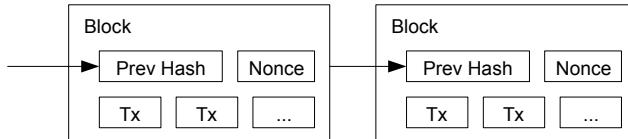
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

5. Network

The steps to run the network are as follows:

- 1) New transactions are broadcast to all nodes.
- 2) Each node collects new transactions into a block.
- 3) Each node works on finding a difficult proof-of-work for its block.
- 4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
- 5) Nodes accept the block only if all transactions in it are valid and not already spent.
- 6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

6. Incentive

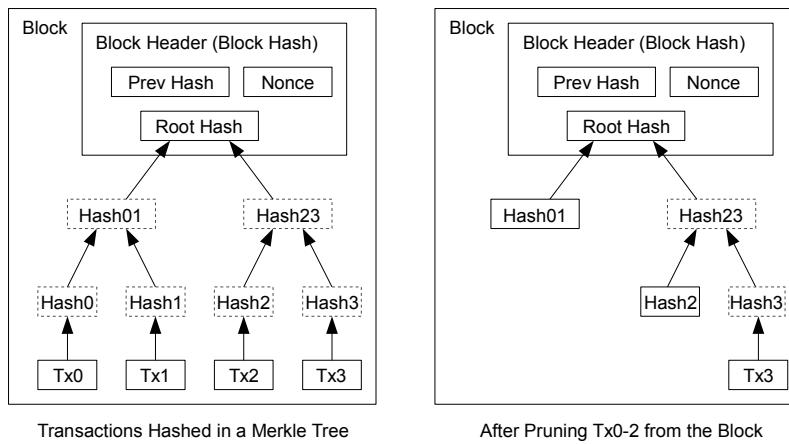
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant amount of new coins is **analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended**.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

7. Reclaiming Disk Space

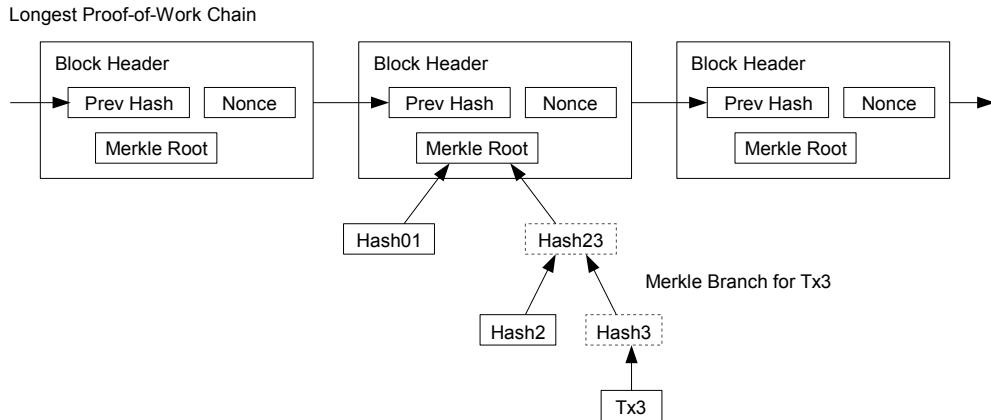
Once the latest transaction in a coin is buried under enough blocks, the **spent transactions** before it **can be discarded to save disk space**. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, $80 \text{ bytes} * 6 * 24 * 365 = 4.2\text{MB}$ per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

8. Simplified Payment Verification

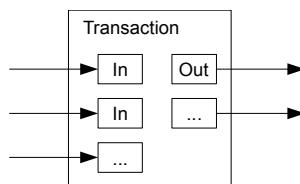
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

9. Combining and Splitting Value

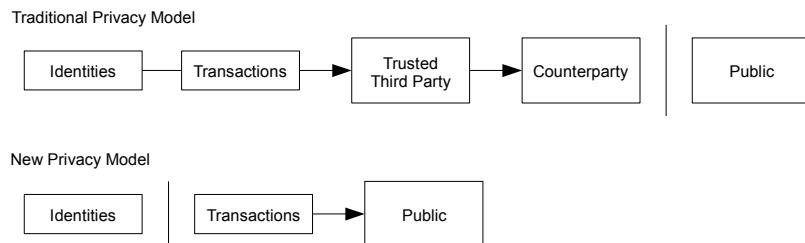
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

10. Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to **announce all transactions** publicly precludes this method, but **privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous**. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

11. Calculations

We consider **the scenario of an attacker trying to generate an alternate chain** faster than the honest chain. Even if this is accomplished, **it does not throw the system open to arbitrary changes**, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. **An attacker can only try to change one of his own transactions to take back money he recently spent**.

The race between the honest chain and an attacker chain can be characterized as **a Binomial Random Walk**. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:

p = probability an honest node finds the next block

q = probability the attacker finds the next block

q_z = probability the attacker will ever catch up from z blocks behind

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

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z \frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

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

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^z \frac{\lambda^k e^{-\lambda}}{k!} (1 - (q/p)^{(z-k)})$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Running some results, we can see the probability drop off exponentially with z.

```
q=0.1
z=0    P=1.000000
z=1    P=0.2045873
z=2    P=0.0509779
z=3    P=0.0131722
z=4    P=0.0034552
z=5    P=0.0009137
z=6    P=0.0002428
z=7    P=0.0000647
z=8    P=0.0000173
z=9    P=0.0000046
z=10   P=0.0000012

q=0.3
z=0    P=1.000000
z=5    P=0.1773523
z=10   P=0.0416605
z=15   P=0.0101008
z=20   P=0.0024804
z=25   P=0.0006132
z=30   P=0.0001522
z=35   P=0.0000379
z=40   P=0.0000095
z=45   P=0.0000024
z=50   P=0.0000006
```

Solving for P less than 0.1%...

```
P < 0.001
q=0.10  z=5
q=0.15  z=8
q=0.20  z=11
q=0.25  z=15
q=0.30  z=24
q=0.35  z=41
q=0.40  z=89
q=0.45  z=340
```

12. Conclusion

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

References

- [1] W. Dai, "b-money," <http://www.weidai.com/bmoney.txt>, 1998.
- [2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.
- [3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.
- [4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.
- [5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.
- [6] A. Back, "Hashcash - a denial of service counter-measure," <http://www.hashcash.org/papers/hashcash.pdf>, 2002.
- [7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.
- [8] W. Feller, "An introduction to probability theory and its applications," 1957.